Waterpixels et Leur Application à l'Apprentissage Statistique de la Segmentation

Vaïa Machairas

THÈSE DE DOCTORAT

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

Préparée à MINES ParisTech

WATERPIXELS AND THEIR APPLICATION TO IMAGE SEGMENTATION LEARNING

WATERPIXELS ET LEUR APPLICATION A L'APPRENTISSAGE STATISTIQUE DE LA SEGMENTATION

École doctorale n°432

SCIENCES DES MÉTIERS DE L'INGÉNIEUR

Spécialité MORPHOLOGIE MATHÉMATIQUE

Soutenue par **Vaïa MACHAIRAS**
le 16 Décembre 2016

Dirigée par **Etienne DECENCIÈRE**
et Thomas WALTER

COMPOSITION DU JURY :

Mr Nicolas PASSAT
Université de Reims Champagne-Ardenne
Président

Mme Corinne VACHIER
Université Paris Est Créteil
Rapporteur

Mme Thérèse BALDEWECK
L'Oréal Recherche et Innovation
Membre du jury

Mme Valery NARANJO
Universidad Politécnica de Valencia
Membre du jury

Mr Jean SERRA
ESIEE, Membre du jury

Mr Etienne DECENCIÈRE
MINES ParisTech, Membre du jury

Mr Thomas WALTER
MINES ParisTech, Membre du jury

Remerciements

Cette thèse a été réalisée au Centre de Morphologie Mathématique (CMM), sur le site de Fontainebleau de MINES ParisTech. Je tiens à remercier mes deux encadrants, Etienne Decencière (CMM) et Thomas Walter (Centre de Bio-Informatique de l'Ecole) pour leur aide et leur confiance tout au long de ces trois années. Leur implication, leur soutien et leurs conseils avisés m'ont permis de beaucoup progresser et d'avoir toujours envie de me dépasser.

Je souhaiterais remercier les membres de mon jury, Mr Nicolas Passat (président du jury et rapporteur), Mme Corinne Vachier-Lagorre (rapporteur), Mme Thérèse Baldeweck (examinatrice), Mme Valery Naranjo (examinatrice) et Mr Jean Serra (examinateur), pour leur intérêt pour les waterpixels et de m'avoir permis d'obtenir le titre de Docteur en Morphologie Mathématique.

Pour qu'un enfant grandisse, il faut tout un village. (Proverbe africain) Cette thèse n'aurait pas été si épanouissante, tant d'un point de vue scientifique que personnel, sans la présence d'un grand nombre de personnes. J'ai eu la chance de pouvoir vivre cette belle expérience entourée par la grande famille du CMM. Je tiens donc à remercier Fernand Meyer et Michel Bilodeau de m'avoir accueillie pour mon stage puis ma thèse respectivement, et tous les collègues que j'ai pu rencontrés pendant ces deux périodes: Dominique Jeulin, Serge Beucher, Jesus Angulo, Beatriz Marcotegui, Petr Dokladal, Petr Matula, François Willot, Matthieu Faessel, Serge Koudoro, Bruno Figliuzzi, Santiago Velasco-Forero, mais aussi Julie, Hellen, Vincent, Thibaud, Xiwei, Torben, El Hadji, André, John, Dario, Andres, Bassam, Enguerrand, Emmanuel, Amira, Luc, Nicolas, Jia-Xin, Joris, Jean-Charles, Gianni, Haisheng, Sebastien, Amin, Jean-Baptiste, Théodore, Robin, Antoine, Kaiwen et Albane. Je n'oublierai pas les bons moments passés ensemble et toutes les discussions que nous avons eu pour refaire le monde (avec ou sans morphologie mathématique). Mention spéciale à *mi hermanito*, qui a été un modèle pour moi tout au long de ma thèse, et à la team de printemps/été/automne/hiver qui a toujours été là pour moi et qui a été exceptionnelle en toutes circonstances. Last but not least, rien n'aurait pu se faire sans la participation de Catherine Moysan et Anne-Marie De Castro, deux secrétaires en or, d'une générosité sans bornes.

Je souhaiterais aussi remercier les collègues rencontrés en dehors du CMM, notamment Gaëlle, Pascale, Laura, Claudie, Jihane, Emilie, Marine, Ricardo, Dariouche, Pierre, Nelson, Angélique, Arezki, Julia, Aurélie et Lydia. Votre amitié a été très précieuse. Merci également à toute la team du mardi, qui m'a suivie dans ce merveilleux projet avec confiance et enthousiasme. Vos sourires et votre détermination ont été pour moi la plus belle des récompenses.

Je tiens à remercier tous nos collaborateurs sur ce projet : Peter Naylor du CBIO, Nicolas Passat, Jimmy Francky Randrianasoa, Andrés Troya-Galvis, Pierre Gançarski en Alsace, David Cardenas Pena en Colombie, Thérèse Baldeweck de L'Oréal Recherche et Innovation; ainsi que nos collaborateurs lors de mon stage : Marie-Claire Schanne-Klein, Stéphane Bancelin, Carole Aimé, Thibaud Coradin et Claire Albert.

Merci à Sabine Süsstrunk, Kawalina, Flavien, Kaem et Marine, Céline et Cain, Carolina et Mehdi Kerkouche de m'avoir inspirée au cours de ces trois années. Merci aussi à Mme El Mejri, Mme Rivière, Mme Adams, Mme Turin, Mohammed, Mr Baume et Mr Hébert pour leurs en-

seignements qui m'ont guidée jusqu'ici.

Je voudrais remercier tous mes amis pour leur soutien, en particulier Fabrice, Anthony, Catherine, Murielle, Ouarda, Severine, Sixtine, Guianie et Julien.

Enfin un immense merci à toute ma famille, en particulier à mes parents et ma soeur Lara qui ont toujours été là pour moi. Je vous aime.

Je terminerai avec deux citations utiles à tout doctorant préparant une thèse :

It always seems impossible until it's done. Nelson Mandela.

Quand tu arrives en haut de la montagne, continue de grimper. Proverbe tibétain.

Résumé

***

L'objectif de ces travaux est de fournir une méthode de segmentation sémantique qui soit générale et automatique, c'est-à-dire une méthode qui puisse s'adapter par elle-même à tout type de base d'images, afin d'être utilisée directement par les non-experts en traitement d'image, comme les biologistes.

Pour cela, nous proposons d'utiliser la classification de pixel, une approche classique d'appren -tissage supervisé, où l'objectif est d'attribuer à chaque pixel l'étiquette de l'objet auquel il appartient. Les descripteurs des pixels à classer sont souvent calculés sur des supports fixes, par exemple une fenêtre centrée sur chaque pixel, ce qui conduit à des erreurs de classification, notamment au niveau des contours d'objets. Nous nous intéressons donc à un autre support, plus large que le pixel et s'adaptant au contenu de l'image : le *superpixel*.

Les superpixels sont des régions homogènes et régulières, issues d'une segmentation de bas niveau. Nous proposons une nouvelle façon de les générer grâce à la ligne de partage des eaux, les *waterpixels*, méthode rapide, performante et facile à prendre en main par l'utilisateur. Ces superpixels sont ensuite utilisés dans la chaîne de classification, soit à la place des pixels à classer, soit comme support pertinent pour calculer les descripteurs, appelés SAF (*Superpixel-Adaptive Features*). Cette seconde approche constitue une méthode générale de segmentation dont la pertinence est évaluée qualitativement et quantitativement sur trois bases d'images provenant du milieu biomédical.

***

Abstract

***

In this work, we would like to provide a general method for automatic semantic segmentation, which could adapt itself to any image database in order to be directly used by non-experts in image analysis (such as biologists).

To address this problem, we first propose to use pixel classification, a classic approach based on supervised learning, where the aim is to assign to each pixel the label of the object it belongs to. Features describing each pixel properties, and which are used to determine the class label, are often computed on a fixed-shape support (such as a centered window), which leads, in particular, to misclassifications on object contours. Therefore, we consider another support which is wider than the pixel itself and adapts to the image content: the *superpixel*.

Superpixels are homogeneous and rather regular regions resulting from a low-level segmentation. We propose a new superpixel generation method based on the watershed, the *waterpixels*, which are efficient, fast to compute and easy to handle by the user. They are then inserted in the classification pipeline, either in replacement of pixels to be classified, or as relevant supports to compute the features, called *Superpixel-Adaptive Features* (SAF). This second approach constitutes a general segmentation method whose pertinence is qualitatively and quantitatively evaluated on three databases from the biological field.

***

List of the main abbreviations

***

CS	Computational support
GT	Groundtruth
ML	Machine learning
MOMA	Mathematical morphology
SAF	Superpixel-adaptive feature
SP	Superpixel
UC	Unit of classification
WP	Waterpixel

***

Contents

1 Introduction

Hope is a choice.
Mary Margaret, Once Upon a Time.

Résumé

L'objectif de ces travaux est de fournir aux non-experts en analyse d'image une méthode générale de segmentation qui puisse s'appliquer facilement à toute base d'images sans avoir besoin d'être adaptée à chaque fois par l'utilisateur. Ce chapitre présente le sujet de thèse ainsi que le plan du manuscrit.

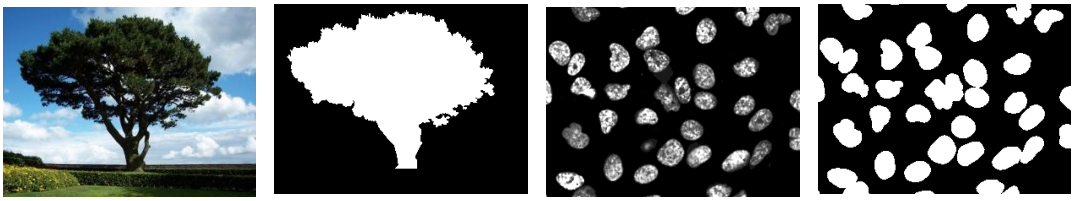
*****

1.1 Context and motivation

Object recognition is one of the most challenging tasks in image analysis. It is particularly useful in the biomedical field, for example, where progress in imaging techniques makes it possible to acquire more and more images with an increasing precision in order to help biologists in their analysis for instance. Processing this data (e.g. counting cells to compute the proportion of abnormal ones, tracking a specific structure through hundreds of slices of a 3D volume corresponding to the MRI of a brain, etc) turns out to be not only tedious (and a potential cause of errors) but also a waste of time in the elaboration of the diagnosis or in the scientific progress towards a better knowledge of biomedical phenomena. Therefore, images are more and more dealt with in an automatic way, in order to save time and ease biologists every day life. This is also true in many other fields, such as in security, urban scene understanding, remote sensing, or even social networks.

In this work, we focus on a special task of object recognition: *semantic segmentation*. Semantically segmenting an image consists in partitioning it into regions which have a meaning, *i.e.* which correspond to real objects in the scene. Each region of this resulting partition is defined by a set of pixels which have the same intensity value. This value, called *label*, corresponds to a unique class of objects (*e.g.* cars, cells, persons, ...).

Figure 1.1 shows some examples of images with their corresponding possible semantic segmentations. The case of image 1 illustrates the search for the main object in the foreground (here a tree) in opposition to the background. Thus, the final segmentation presents only two labels. Figure 1.1.d presents a segmentation of nuclei of image 2, which are assigned the same label as they belong to the same class "nucleus". Finally, the third case, image 3, emphasizes the fact that, for a given image, a segmentation is not unique as we may want to isolate different types of objects, leading each time to a different segmentation result. Figures 1.1.f and 1.1.g are two possible segmentations of image 3, targeting respectively all figures and only the eight. Therefore, the corresponding segmentation method must be designed to find the specified target. Moreover, due to the difficulty of creating such a method, commonly used segmentation tools are often optimized not only for a specific type of object but also to a specific type of images (imaging device, acquisition conditions, etc). Dealing with new images and new objects hence requires, most of the time, a different segmentation method to be designed, which takes time. It is worth noting that in most cases, images are to be dealt with not one by one but rather gathered into *databases*, *i.e.* sets of images sharing similar properties (same types of objects and/or same acquisition conditions etc). Unfortunately, due to the variety of objects and images to be analyzed, it also takes time to design for each database a specific segmentation method, as pointed out above.



(a) Image 1 (b) Segmentation of the main object in (a) (c) Image 2 (d) Segmentation of nuclei in (c)

(e) Image 3 (f) Segmentation of figures in (e) (g) Segmentation of eight in (e)

Figure 1.1: **Segmentations of different images.** Each segmentation presented here is a *groundtruth*, *i.e.* what we would like to obtain at the end of the segmentation procedure. Often performed manually, groundtruth is used as reference when evaluating quantitatively the performance of a segmentation method.

In this work, our aim is to **provide a general segmentation method which could be easily applied to any image database and by any user** regardless of the latter's potential (non-)knowledge of image analysis tools.

We will address this issue with a classification framework (supervised learning). Image pixels (or more precisely the vectors representing them) will constitute the data samples (at least in a first phase) and the classes will represent the different objects we would like to find in the image database. This approach requires that for each database, we already know the labels of some pixels to perform the training phase, *i.e.* we know to which objects some pixels belong. For this purpose,

we will use some images of the database which are already segmented by hand by the experts needing the segmentation of their database. The machine learning method we use will hence try to reproduce the best it can the way experts work when they visually analyze the content of their images.

Remark 1: Thanks to our collaborations, we will focus mainly on databases coming from the biomedical field (see Chap. 3). However, the proposed method is general and can be applied to any other databases.

Remark 2: The considered databases presented groundtruth with complete annotations (*i.e.* pixels were labeled), but only a set of examples for each object class would have been enough to perform the training phase.

1.2 Thesis outline

As we have seen, the aim of this work is to provide a general segmentation method achieving good performance on any image database without needing to be modified each time by the user (expert or not in image analysis), as far as the latter provides some examples already segmented by hand within the database to be segmented.

As previously announced, we first propose, inspired by the literature, to address this issue with *pixel classification*, a general approach based on supervised learning. Part I is dedicated to the set up and assessment of such a workflow. This method is indeed constituted of several steps, which are expounded in Chap. 2. We then evaluate qualitatively and quantitatively this pipeline on different databases to check if it achieves good segmentation results no matter what the asked segmentation task is (see Chap. 3). This leads us to consider the solutions which could improve this already promising method.

We subsequently focus on another image representation, *superpixels*, producing a special low-level segmentation of the image. As pointed out by an abundant literature, this structure seems to be a promising lead to improve segmentation performance when using the general framework of classification.

In Part II, we remind what superpixels are and which specific properties they should satisfy (see Chap. 4). With the help of another field of image analysis, namely Mathematical Morphology, we propose a new method to generate them based on the watershed transformation (see Chap. 5). We show that these superpixels, hence called *waterpixels*, offer good performance in terms of quality and computation time.

Once generated, these superpixels can be used to improve the classification pipeline performed to obtain the final segmentation. Part III presents two different uses of waterpixels in such a workflow: superpixel classification (as proposed in the literature) and a novel application: Superpixel-Adapative Features (SAF). They are expounded in Chap. 6 and Chap. 7 respectively.

Conclusion and prospects are then presented in Chap. 8.

While the aim of this work is to provide a general segmentation method, **the main contribution of this PhD is to study the relevance of the image representation embodied by superpixels in the proposed workflow**.

On another level, we also would like to show that Machine Learning and Mathematical Morphology are two powerful fields which can benefit from each other in order to go one step further towards efficient and automatic object recognition.

We have decided to make our codes available to the scientific community. They can be downloaded on our website: **http://cmm.ensmp.fr/∼machairas/**. Programming was done in Python, with the help of Morph-M (see Koudoro et al. [2012]), SMIL (see Faessel and Bilodeau [2013]), scikit image (van der Walt et al. [2014]), numpy (Dubois et al. [1996], Ascher et al. [1999] and Oliphant [2006]), mahotas (Coelho [2012]), scikit learn (Pedregosa et al. [2011a]) and Vigra libraries (Köthe [2000]).

Segmentation and Classification

The aim of this thesis is to provide a general method to segment every image database by supervised learning.

In this part, we propose to address this issue with a classic approach often used in the literature: ***pixel classification***. Indeed, it consists in assigning to each pixel the label of the object type it belongs to. At the end of the process, all pixels which have obtained the same label form one or more connected components which belong to the same type of structures (*e.g.* cells, cars, background, ...). The result constitutes the semantic segmentation of the image.

If it is appropriately designed, this pipeline can perform supervised learning on every database, requiring only, from the user, to provide each time some examples of images already segmented by hand. This is why it is used by general segmentation software dedicated, for instance, to the biological field (such as Ilastik by Sommer et al. [2011]).

In a first phase, we expound and set up such a workflow with general features (standard operators used by Ilastik, operators from mathematical morphology, textural features: Haralick and Gabor filters) and Random Forests (Breiman [2001]). We apply this pixel classification to three different databases coming from the biological field (L'Oréal, CAMELYON16, Coelho et al. [2009]) and analyse the segmentation performance of this approach.

We conclude by suggesting to insert *superpixels* in this pipeline, a solution which is more thoroughly presented and assessed in Parts 2 and 3.

2 Segmentation as a classification task

La philosophie est écrite dans ce grand livre qui s'étend chaque jour devant nos yeux : l'univers.
Mais on ne peut le comprendre si nous n'apprenons d'abord son langage
et si nous ne comprenons les symboles avec lesquels il est écrit.
Galilée

Résumé

Dans ce chapitre, nous décrivons une approche classique de segmentation d'image qui utilise l'apprentissage statistique pour s'adapter à tous types de bases d'images. En effet, le choix a été fait dans ces travaux de thèse de voir la segmentation comme une tâche de classification, où l'objectif est d'assigner à chaque pixel l'étiquette du type d'objet auquel il appartient. Nous exposons comment construire une chaîne de classification de pixels en utilisant des descripteurs de pixels généraux issus de la littérature ainsi qu'une méthode d'apprentissage robuste et efficace : les forêts aléatoires (Breiman [2001]). Cette chaîne sera évaluée qualitativement et quantitativement sur plusieurs bases d'images dans le chapitre suivant.

*****

In this work, we have decided to consider segmentation as a *classification task*, where the aim is to assign to each pixel the label of the object it belongs to. This chapter is dedicated to expound how pixel classification works (see Sec. 2.1), how we build this classic pipeline with elements from the literature (see Sec. 2.2 and Sec. 2.3) and why it enables to design a general segmentation method working on different databases.

2.1 General scheme for pixel classification

In this section, we will expound how pixel classification works on a given image database $\mathscr{D}$. Note that this procedure should be performed every time the image database changes.

Recall that database $\mathscr{D}$ is split into two subsets: $\mathscr{D}_{\mathscr{T}}$ (manual segmentations available) used for *training* phase and $\mathscr{D}_{\mathscr{P}}$ (images to be segmented) used for *prediction* phase.

Let $f : D \to V$ be an image of $\mathscr{D}$, where D is a rectangular subset of $\mathbb{Z}^2$, and V a set of values,

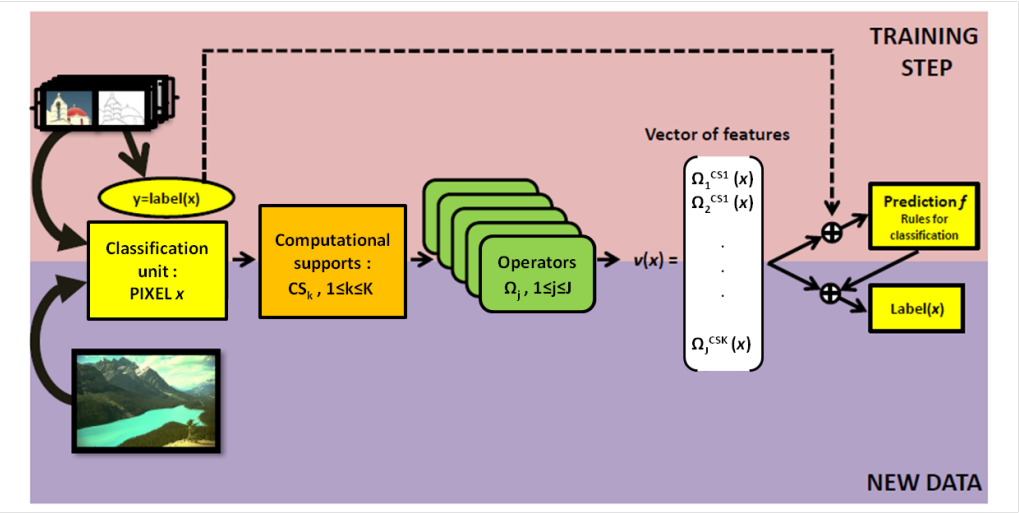


Figure 2.1: General scheme for pixel classification.

for example $\{0,\ldots,255\}$ when f is an 8 bit grey level image or $\{0,\ldots,255\}^3$ for color images. The aim is to assign to each pixel $x \in D$ a label $l(x) \in \mathscr{C}$, where $\mathscr{C} = \{c_1, c_2, \ldots, c_K\}$ is the set of labels corresponding to the K categories of objects to be found. The scope of this work will be limited to binary classification (*i.e.* $\mathscr{C} = \{0,1\}$) but can easily be extended to more than two classes. At this stage, the two classes could be "object (1) vs. background (0)" or "contour of an object (1) vs. not an object contour (0)". We will prefer here the region approach corresponding to the former. This also implies that the groundtruth should be provided in the same format.

The general scheme for pixel classification is illustrated in Fig. 2.1. Note that pixels constitute *units of classification* (**UC**), *i.e.* they are the elements to which we would like to assign a label at the end of the process. As a supervised learning technique, classification is comprised of two phases: a ***training*** step and a ***prediction*** step. The aim of training is to establish the classification rules assigning a label to each pixel. As explained in the previous chapter, this first phase is performed on the set of pixels of $\mathscr{D}_{\mathscr{T}}$ for which labels are already known since they are part of the images which have already been manually segmented by experts. The task of the learning method (notation: **ML method**) is to understand the link between this already known label and the pixel's properties which are subsumed into its *vector of features*. Then, in the second phase, called *prediction*, any new pixel from $\mathscr{D}_{\mathscr{P}}$ can be assigned the label of the object it belongs to by computing its vector of features and giving the latter as input to the learning method which will apply the learned *prediction function* and find the pixel's label.

Notation 2.1. *If p is the number of features computed for a pixel x, then each vector $v(x)$ is in $\mathbb{R}^p$. We denote by $X \in \mathbb{R}^{n \times p}$ the data matrix which is given as input to the ML method for training, where n is the number of pixels in $\mathscr{D}_{\mathscr{T}}$.*

In the following, we will get in more detail about the computation of the vector of features (see Sec. 2.2) and the machine learning method used (see Sec. 2.3).

2.2 Features

In this section, we will explain how to obtain the vector of features $v(x) \in \mathbb{R}^p$ of each pixel x used for training or prediction.

2.2.1 Definitions

First of all, what is a feature? A feature describes a pixel property. For instance, what is the intensity of the pixel? Or what is its position on the x-axis in the image? As an example, if the former is computed on a pixel of an 8-bit image, the answer will lie in the range $[0,255]$. Note that the result may not be a number, e.g. when asking yes/no questions (Boolean) (e.g. *is intensity greater than 140?*), or if we get an histogram. In this work, all features used will be adapted so as to obtain a single value in $\mathbb{R}$: *yes/no* can be converted to $\{0,1\}$, an histogram with B bins can be converted to B features, etc. Indeed, this conversion step, easy to perform, is necessary as it is compulsory for the learning method used afterwards.

What is a "good" feature? A good feature is a feature that helps discriminate the data (here the pixels) into the two classes defined in the ground truth, as illustrated in Fig. 2.2.a, where each feature is represented by one dimension of the space: we can see that f_1 is a discriminant feature (it is easy to find a threshold which can split perfectly the data into two pure groups), whereas f_2 is not. Note that, even if a feature is not discriminant on its own, it can still be useful combined with others, as shown in Fig. 2.2.b. where the separation between the two classes must take into account the two features f_1 and f_2 to be efficient. In our case, we have decided to compute a large number of general features (*i.e.* not specific to a given task) in order to ensure the generality of our method. Then, for each database, the power of discrimination of a given feature can vary, as well as for all their possible combinations. To ensure a good classification performance, we have chosen a machine learning method which is able, as far as possible, to discard "bad" features and use only "good" ones among the general set provided (see Sec. 2.3).

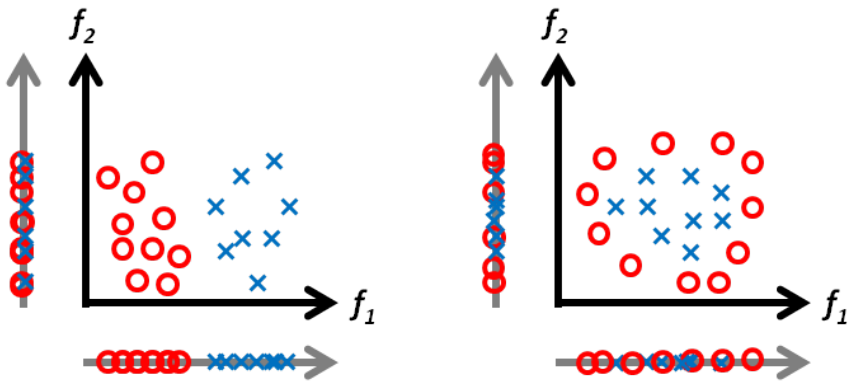


Figure 2.2: What is a good feature?

More formally, a feature is defined as follows:

Definition 2.2.1 — Feature. A feature F can be described by the set of following elements:

$$F = \{c, \Omega, GEO, CS, \Sigma\}$$

where:

1. Ω is the **operator** which is applied to the image (e.g. *Identity*, linear or non-linear filter-

ings, etc).

2. c is the **channel**, or the combination of channels, on which the operator is applied (e.g.: *Identity* on the red channel to obtain the red intensity; L, a and b channels to compute the Lab gradient, etc). In this work, all operators will be applied on a single channel.

3. CS is the **computational support**, *i.e.* the support on which the feature is computed. Indeed, when describing a pixel, looking only at this very pixel may be somewhat limited. On the contrary, widening the field of view may capture richer information. Therefore, one can use various *computational supports*, from the pixel itself to some fixed neighborhood or some fixed window centered on it.

4. Σ is the **integrator**. Indeed, when widening the computational support, the set of pixels included in the latter will give a set of corresponding values. If the computational support has the same shape and size for every pixel, all these V values can be kept, thus corresponding to V features, or they can be "integrated" into one to give a statistical representation of this richer support, using, for example, the mean, the max, the min or the standard deviation value.

5. GEO is a boolean corresponding to the way operators are applied on images (**geodesic** or **non-geodesic way**). It will be explained into more details in Chapt. 6. For the moment, we assume that each operator is applied to the whole image and that the integrated value is computed afterwards on the computational support.

Remark: Note that we call ***unit of classification*** (notation: **UC**) the unit to be classified, *i.e.* the unit to which we would like to assign a label at the end of the classification process. In this chapter, UCs are the pixels. It is different from the concept of ***computational support*** (**CS**) which is used as support to compute a feature in order to describe the properties of the chosen UC while computing the latter's vector of features. They can be the same (for example when we perform a pixel classification with features computed on this very pixel) or different (pixel classification with features computed on centered windows for instance).

In the following paragraph, we will review some operators and their combination with some typical computational supports used in the literature for pixel classification. They will serve in the next chapter to build a general pixel classification pipeline.

2.2.2 Operators Ω and computational supports CS

In addition to the Identity operator, all transformations and filtering techniques can be used as operators to compute features. We will consider in particular four families of operators:

1. **Standard operators** previously used for pixel classification as in Sommer et al. [2011]: Gaussian smoothing, Laplacian of Gaussian, Difference of Gaussian, Gaussian Gradient Magnitude, eigenvalues of structure tensor ($\times 2$), eigenvalues of Hessian of Gaussian ($\times 2$). Varying the parameter σ of the Gaussian leads to different filter responses.
2. **MOMA**, a set of non-linear transformations from the field of Mathematical Morphology: erosion ε, dilation δ, opening γ, closing ϕ, white top hat TH, black top hat THi and morphological gradient MG (see Soille [2003] for a review). The structuring element neighborhood and size are indicated each time as power and index respectively. Example: ε_3^{V6} denotes an erosion whose structuring element is defined by size 3 and 6-connectivity neighborhood.
3. **Gabor filters** (Kamarainen [2003]), textural features, parametrized by the *bandwidth* of the

Gaussian, and two parameters for the sinusoide: its *frequency* and its *direction*.
4. **Haralick features** (Haralick et al. [1973]), textural features, averaged over all directions.

We have used the implementations of vigra (Köthe [2000]), SMIL (Faessel and Bilodeau [2013]), scikit image (van der Walt et al. [2014]) and mahotas (Coelho [2012]) libraries for these four families respectively.

As far as computational supports are concerned, we focus on two usual *CS* to perform pixel classification: the pixel itself and the window centered on it. Figures 2.3 and 2.4 show examples of operators applied on the same image and integrated over these two *CS* respectively (Σ: mean).

Remark: Some of these operators highlight contour pixels vs. non contour pixels, instead of highlighting objects pixel vs. background pixels. If the *CS* is the pixel, these features will not help to discriminate objects vs. background as they will be considered as "bad" features discarded by the Random Forests. If the *CS* is wider, such as a sliding window, they will give information on contour statistics in this area; hence they can serve as textural features enabling to discriminate objects from one another.

2.3 A powerful machine learning method: Random Forests

Random Forests (**RF**), an efficient machine learning method, have been chosen to process the data. Proposed by Breiman [2001], a random forest relies on a set of decision trees which each outputs a probability, for each data sample, to belong to a given class, and realizes afterwards a consensus between their classification results to obtain the final labels. This section expounds more thoroughly how random forests work and thus why they are well suited for the design of a general segmentation paradigm using classification.

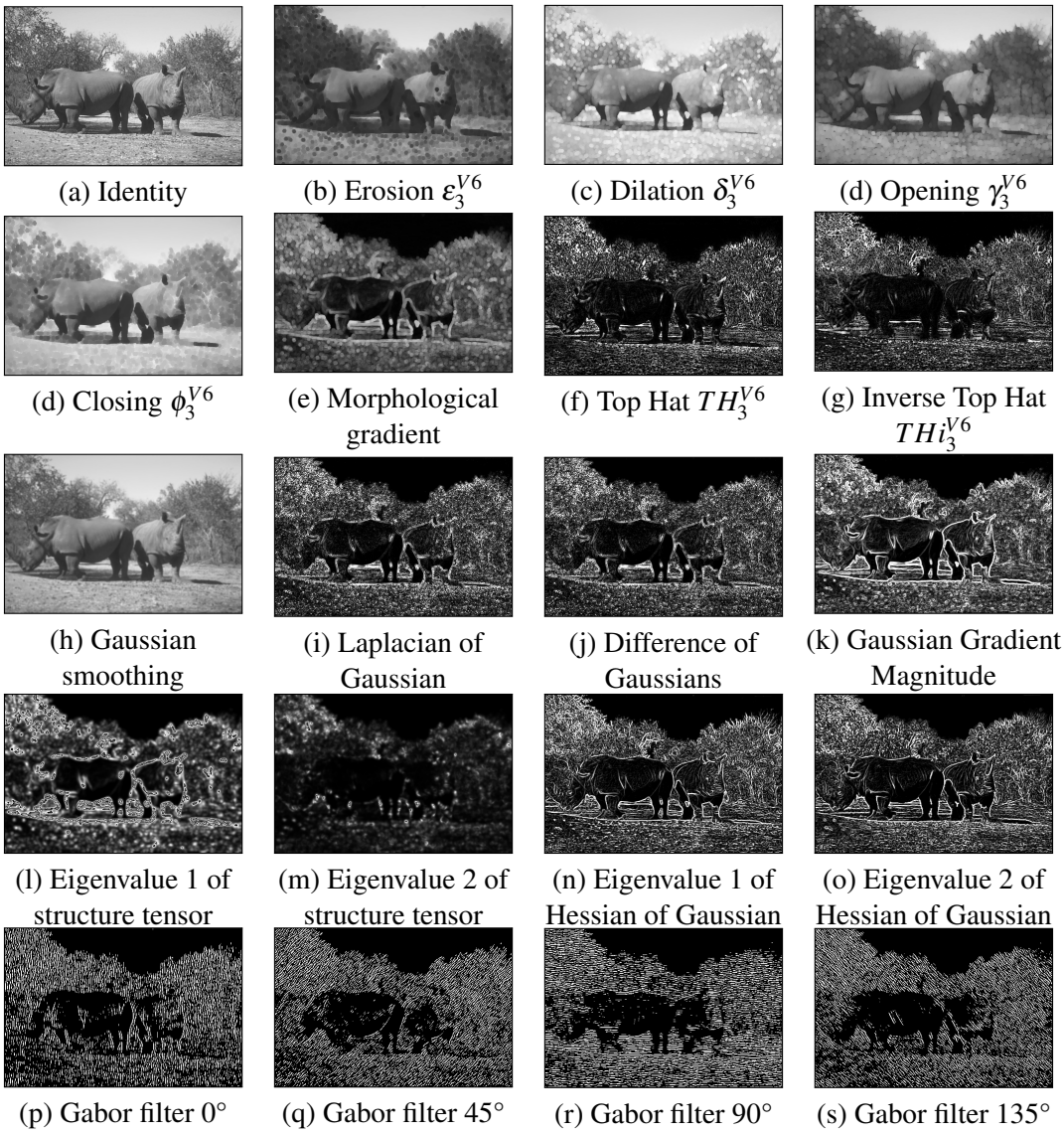
2.3.1 Principle and application to classification

Binary decision tree and classification

A random forest is a collection of B binary decision trees. In this paragraph, we will define what is a binary decision tree and show how to use it for the binary classification of data samples (*i.e.*, in our case, pixels of a given database $\mathscr{D}$). More information can be found in Breiman [2001] and Peter et al. [2015].

> **Definition 2.3.1 — Binary decision tree.** A binary decision tree b is a hierarchically organized set of nodes such that, starting from a *root* node, each node has exactly 0 or 2 child nodes. A node without children is called a *leaf* (or *terminal node*). Each non-terminal node contains a binary decision called *splitting function* designed to route instances towards the left or right child node. Each instance sent initially to the root recursively passes through the tree until it reaches a leaf.

It is possible to use a binary decision tree for classification: the idea is to route every data sample, sent initially to the root, through the nodes until it reaches a leaf which will contain information on its most probable class label. The construction of such a tree is performed during the training phase, during which all splitting functions are found, with the help of the already known labels of the training set. The aim of a splitting function is to split data seen by a given node into two subsets which are "purer" than the set seen by this very node, *i.e.* that each subset should then contain an increased majority of one data class label over the other one. How can we find the splitting function associated to a given node i? Let $\mathscr{F}$ be the set of features used to



(a) Identity (b) Erosion ε_3^{V6} (c) Dilation δ_3^{V6} (d) Opening γ_3^{V6}

(d) Closing ϕ_3^{V6} (e) Morphological gradient (f) Top Hat TH_3^{V6} (g) Inverse Top Hat THi_3^{V6}

(h) Gaussian smoothing (i) Laplacian of Gaussian (j) Difference of Gaussians (k) Gaussian Gradient Magnitude

(l) Eigenvalue 1 of structure tensor (m) Eigenvalue 2 of structure tensor (n) Eigenvalue 1 of Hessian of Gaussian (o) Eigenvalue 2 of Hessian of Gaussian

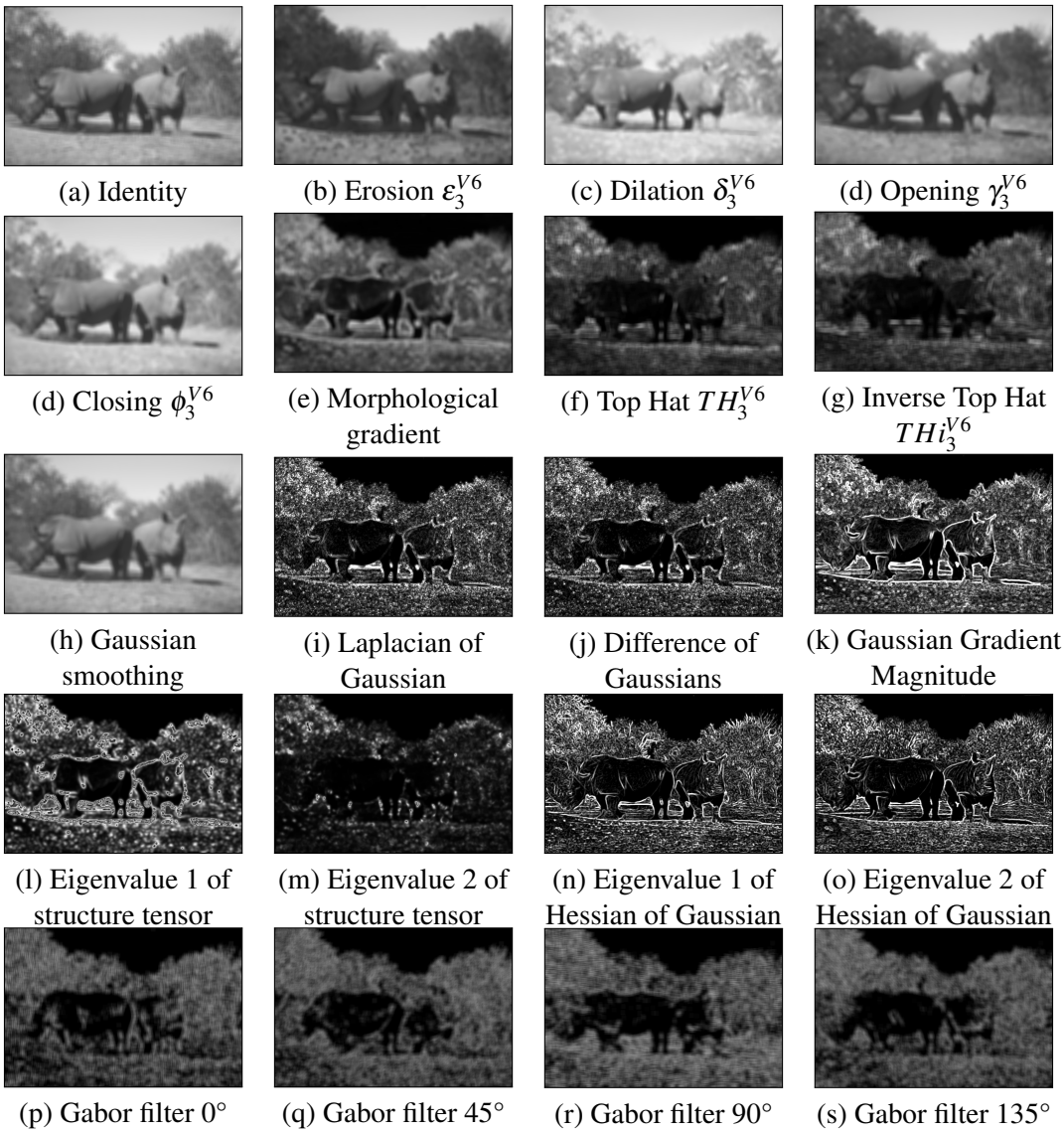
(p) Gabor filter 0° (q) Gabor filter 45° (r) Gabor filter 90° (s) Gabor filter 135°

Figure 2.3: Application of some operators (with typical parameter values) on an image from the Berkeley segmentation Database (Martin et al. [2001]). **CS: pixel.**

compute the vector of features of each pixel, with values in $\mathbb{R}$. A splitting function can be defined as a couple $(f,\theta) \in \mathscr{F} \times \mathbb{R}$ where f is a feature and θ is a threshold. For a given node i seeing a subset of data S_i, we define the two subsets $S_{i,L}^{f,\theta} = \{x \in S_i | f(x) \leq \theta\}$ and $S_{i,R}^{f,\theta} = \{x \in S_i | f(x) > \theta\}$ and the information gain generated by this split as:

$$IG(S_i,f,\theta) = G(S_i) - \frac{|S_{i,L}^{f,\theta}|}{|S_i|} \times G(S_{i,L}^{f,\theta}) - \frac{|S_{i,R}^{f,\theta}|}{|S_i|} \times G(S_{i,R}^{f,\theta}) \tag{2.1}$$

where $G(S_i)$ is a purity measure of the set S_i (the Gini index, Gini [1912], in our case). In practice, to create a split given a feature f and a set of samples S_i, we consider t thresholds $\theta_1, \theta_2, \ldots, \theta_t$ regularly distributed between the extreme values of $f(p)$ observed over all $p \in S_i$. The threshold providing the highest information gain is retained and defines the information gain $IG(S_i,f)$ of the feature f given S_i. Then, the retained splitting function $(\hat{f},\hat{\theta})$ is determined by keeping the feature $\hat{f}$ providing the highest information gain, together with its corresponding best threshold $\hat{\theta}$.



(a) Identity (b) Erosion ε_3^{V6} (c) Dilation δ_3^{V6} (d) Opening γ_3^{V6}

(d) Closing ϕ_3^{V6} (e) Morphological gradient (f) Top Hat TH_3^{V6} (g) Inverse Top Hat THi_3^{V6}

(h) Gaussian smoothing (i) Laplacian of Gaussian (j) Difference of Gaussians (k) Gaussian Gradient Magnitude

(l) Eigenvalue 1 of structure tensor (m) Eigenvalue 2 of structure tensor (n) Eigenvalue 1 of Hessian of Gaussian (o) Eigenvalue 2 of Hessian of Gaussian

(p) Gabor filter 0° (q) Gabor filter 45° (r) Gabor filter 90° (s) Gabor filter 135°

Figure 2.4: Application of some operators (with typical parameter values) on an image from the Berkeley segmentation Database (Martin et al. [2001]). **CS: window** of radius $r = 10$.

After splitting, $S_{i,L}^{\hat{f},\hat{\theta}}$ and $S_{i,R}^{\hat{f},\hat{\theta}}$ are respectively sent to the left and right child nodes. The process is recursively repeated until a maximum depth of the tree is reached or until the number of samples sent to child nodes is too low, in which case a leaf is created. The posterior probability stored at a leaf is defined as the class distribution over the arriving subset of labeled samples.

This way, after the training phase is performed, any unlabeled data sample (pixel from $\mathscr{D}_{\mathscr{P}}$) passes through the tree starting from the root and reaches a leaf. The label of the majority class in the posterior probability of this leaf (stored during training) is then assigned to this data.

Random forests and classification

If a binary decision tree is a good candidate for classification, it is nevertheless prone to over-fitting (*i.e.* fitting too well to the training data and losing the generality, *i.e.* the ability to adapt well to any new data). The idea of random forests is thus to combine the decision of a large number of

trees to limit over-fitting. For this to be successful, trees must be as decorrelated from one another as possible. Two sources of randomness are introduced to enforce the specialization of the trees:

1. Each tree sees a different subset of the training data. If $|\mathscr{D}_{\mathscr{T}}| = m$, then each tree sees m data samples which have been selected in $\mathscr{D}_{\mathscr{T}}$ by uniform random sampling with replacement.
2. At each node, only a uniform randomly sampled subset of features is considered when looking for the best candidate and hence best splitting function.

Note that, for prediction, the final label assigned to a data sample is the most represented label among the b labels output by the b trees.

2.3.2 Settings

Subsampling

To reduce the computation time of the training phase, a random and balanced sub-sampling has been applied to the training set $\mathscr{D}_{\mathscr{T}}$, which was reduced to 100 000 samples.

Missing values

Random forests are not able to deal with missing values in the data matrix X given as input. Yet, it may not be possible to compute some features in certain cases, leading to missing values. We can manage those by replacing them with consistent values. Our replacement strategy is the following. We remind that in the data matrix $X \in \mathbb{R}^{n \times p}$, the rows correspond to samples and the columns to features. For each row i ($i \in \{1,\dots,n\}$), we define by J_i the set of column numbers for which there is a missing value:

$$J_i = \{j \in [1,p] | X(i,j) \text{is a missing value}\} \tag{2.2}$$

Let us suppose that there is one or more missing values in row i_0. We search for the five other rows i_1, i_2, i_3, i_4 and i_5 (without missing values) which are the most similar to row i_0, *i.e.* which minimize the most the following cost:

$$\sum_{j \notin J_{i_0}} (X(i,j) - X(i_0,j))^2 \tag{2.3}$$

Once the $\{i_l\}_{l \in [1,5]}$ are found, each missing value in i_0 is replaced by the average of the other five values located in the same column:

$$\forall j \in J_i, X(i_0,j) = \frac{1}{5} \sum_{l=1}^{5} X(i_l,j) \tag{2.4}$$

Random Forest parameters

Random forests offer many parameters that must be tuned to achieve good classification performance. The main parameters are the number of trees, the purity criterion, the number of features to be considered when looking for the best split and the maximal depth of a tree. Parameter values have to be optimized based on the number of training samples, the number of features used and the database considered. In practice, we will perform a model selection procedure during training to find the best possible tuple of values (v values if there are v parameters). Let us suppose that the considered database $\mathscr{D}$ is split into 3 subsets called *train*, *validation* and *test* subsets ($\mathscr{D}_{train}$, $\mathscr{D}_{val}$ and $\mathscr{D}_{test}$ respectively). Let $\mathscr{M} = \{\mathscr{M}_l, \mathscr{M}_\infty, \dots, \mathscr{M}_{\mathscr{L}}\}$ be the set of model candidates (*i.e.* random forests with different sets of parameters values). The correct procedure should be as follows:

1. Each model $\mathscr{M}_l, l \in [0,L]$ is trained on $\mathscr{D}_{train}$, then applied on $\mathscr{D}_{val}$ for prediction and evaluation of the classification performances.
2. The model $\mathscr{M}_{\hat{l}}$ which achieves the best performances on $\mathscr{D}_{val}$ is chosen.

3. The selected model $\mathscr{M}_{\hat{f}}$ is applied for prediction on the test subset $\mathscr{D}_{test}$ to obtain the final classification performance estimation.

If the database is only constituted of a train and a test subsets $\mathscr{D}_{\mathscr{T}}$ and $\mathscr{D}_{\mathscr{P}}$, which is generally the case, the first subset should itself be split into *train* and *val* parts: $\mathscr{D}_{\mathscr{T}} = \{\mathscr{D}_{train}; \mathscr{D}_{val}\}$. But how can we be sure that this split does not impact the classification performance, *i.e.* that another split could not have led to different classification performance? To address this issue, we generally perform cross-validation to evaluate a model candidate. It consists in randomly splitting $\mathscr{D}_{\mathscr{T}}$ into k folds, performing the training on $k-1$ folds and prediction on the remaining fold, repeating this process k times (one for each of the k different folds). The k results are then averaged to give a fair evaluation of the candidate model. Note that two folds should not contain pixels of the same image in order to avoid the overestimation of classification performance.

How to choose the features?

As, in the most widely used version of RF, the split at each node of the tree is performed with a feature that is selected among some random subset of the original feature set, such that it optimizes the separation of training samples, there is an inherent feature selection. If the number of irrelevant features is not too high, the model will therefore tend to disregard the most irrelevant features by construction. Hence, no feature selection, and therefore no input is required from the user.

Implementation

In this work, we have used the random forests implemented in *scikit learn* (Pedregosa et al. [2011b]), a Python machine learning library. We have used typical values for RF parameters (see Breiman [2001]), except for two parameters which were automatically optimized for each database: the number of trees $n\_estimators$ (increasing this number aims at reducing over-fitting but also increases computation time) and $min\_samples\_leaf$ which is linked to the depth of the trees (the minimum number of data samples in a leaf for the latter to be kept while building the tree during training).

2.4 Conclusion

In conclusion, we have presented in this chapter the pipeline of pixel classification which will serve as general segmentation method to be applied on different databases. We have reviewed how to compute the features and how the machine learning method works. Next chapter is dedicated to the construction and assessment of such a workflow on various databases.

3 Tools for segmentation evaluation: Application to the pixel classification workflow

Great dancers are not great because of their technique,
they are great because of their passion.
Martha Graham

Résumé

L'objectif de ce chapitre est d'évaluer les performances de la segmentation par classification de pixels. Nous nous intéressons en particulier aux différences que l'on peut observer entre deux supports de calcul utilisés classiquement pour calculer les descripteurs associés au pixel : soit le pixel lui-même, soit la fenêtre glissante centrée chaque fois sur le pixel. Nous terminons en proposant d'utiliser un support plus adapté au calcul afin d'améliorer les performances de segmentation. Cette solution sera ensuite développée dans les parties 2 et 3 du manuscrit.

*****

The aim of this chapter is to assess the pixel classification pipeline on different databases in order to evaluate its segmentation performance and its general behavior. As previously said, general features are favored to compute the data matrix X which is given as input to the random forest. More specifically, we investigate two computational supports to classify a pixel: either the pixel itself, or a window of a given size centered on this very pixel. The evaluation procedure is described in the first section.

3.1 Evaluation procedure

This section expounds how pixel classification, as well as other future pipelines proposed in this manuscript, will be evaluated. To check if the method used is general, we choose three databases, called $\mathscr{D}_1$, $\mathscr{D}_2$ and $\mathscr{D}_3$, which exhibit different properties (size, types of objects, acquisition procedures, etc). They come from the biological field. Segmentation in all three cases plays a crucial role for quantification of experimental outcomes. The concrete applications however are very diverse and range from fundamental research in cell biology to industrial applications. They are presented in paragraphs 3.1.1 to 3.1.3. Paragraph 3.1.4 focuses on the quantitative criteria used to evaluate segmentation perfomances of the considered approach(es).

3.1.1 The L'Oréal database $\mathscr{D}_1$

The first database $\mathscr{D}_1$ is provided by L'Oréal Recherche et Innovation, thanks to the collaboration with Thérèse Baldeweck.

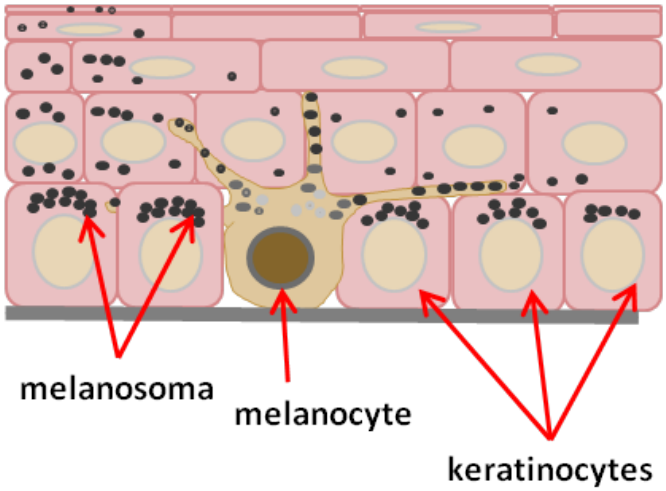
This database contains images of reconstructed skin, acquired by multiphoton microscopy. In these images, we can distinguish melanocytes (bright elongated cells) as well as keratinocytes (circular structures). The role of both types of cells is illustrated in Fig. 3.1.a. Melanocytes are melanin-producing cells located in the bottom layer (the stratum basale) of the skin's epidermis, the middle layer of the eye (the uvea), the inner ear, meninges, bones, and heart. Melanin is the pigment primarily responsible for skin color. Once synthesised, melanin is contained in a special organelle called a melanosome. Melanosoma are moved along the dendrites (arm-like structures) of the melanocytes, so as to reach the keratinocytes which will migrate to the surface of the epidermis, protecting themselves (*i.e.* their DNA) thanks to the melanin. An example image from the database can be seen in Fig. 3.1.b., with a zoom on a melanocyte in Fig. 3.1.c. and a zoom on a keratinocyte in Fig. 3.1.d. .

The database $\mathscr{D}_1$ contains 8 2D grey-level images of size 511x511 pixels. **The aim is to segment all melanocytes** in $\mathscr{D}_1$, *i.e.* to obtain the label 1 for all pixels belonging to a melanocyte and the label 0 for any other pixel (background, keratinocytes, etc). The corresponding groundtruth have been produced by an expert from L'Oréal Recherche et Innovation. The eight pairs of corresponding images are presented in Fig. 3.2 (with enhanced contrast).

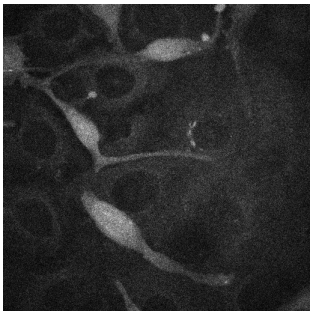
Specific segmentation methods exist for this database, proposed in Serna et al. [2014]. The authors provide a new segmentation method working well on elongated objects such as melanocytes. Images are represented as component trees using threshold decomposition. Elongation and area stability attributes are combined to define area-stable elongated regions. A qualitative and quantitative comparison is made with another segmentation method from the state-of-the-art: *maximally stable extremal regions* (MSER) by Matas et al. [2002]. This method is general but has to be specifically tuned in order to achieve good performance on the considered database. In the following, we will benchmark our pipeline(s) against these two approaches, denoted by *Serna et al.* and *MSER*.

3.1.2 The CAMELYON database $\mathscr{D}_2$

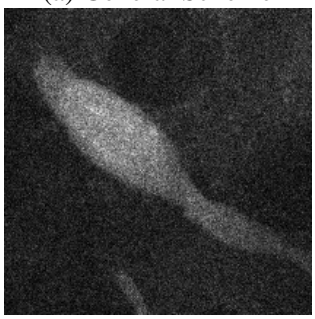
This database is extracted from the recent CAMELYON16 challenge database (ISBI16). This challenge aims at facilitating and improving breast cancer diagnosis by automatically detecting metastases in whole-slide digitized images of lymph nodes. Indeed, metastatic involvement of lymph nodes is a highly relevant variable for breast cancer prognosis; however, the diagnosis procedure for pathologists is "tedious, time-consuming and prone to misinterpretation" as pointed out by the organizers. Hence, an automatic and efficient detection of such structures becomes essential. Their database is constituted of 400 whole-slides images (270 for train, with GT, and 130 for test, without GT) collected in the Radboud University Medical Center as well as in the University Medical Center Utrecht from the Netherlands (see Fig. 3.3 and Fig. 3.4). These images are provided in a multi-resolution pyramid structure, which makes their processing even more challenging (e.g. one slide is too big to be stored in RAM for pertinent high resolutions). Therefore, we extract a simplified database, consisting in selected crops of size 500×500 pixels at resolution 2 of the train slides (for which GT are provided). These regions of interest are equally chosen



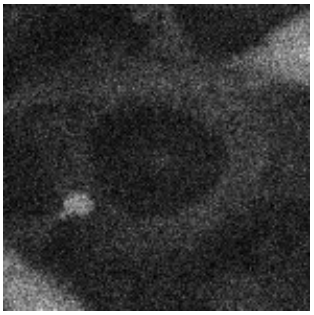
(a) General Scheme



(b) Image of $\mathscr{D}_1$

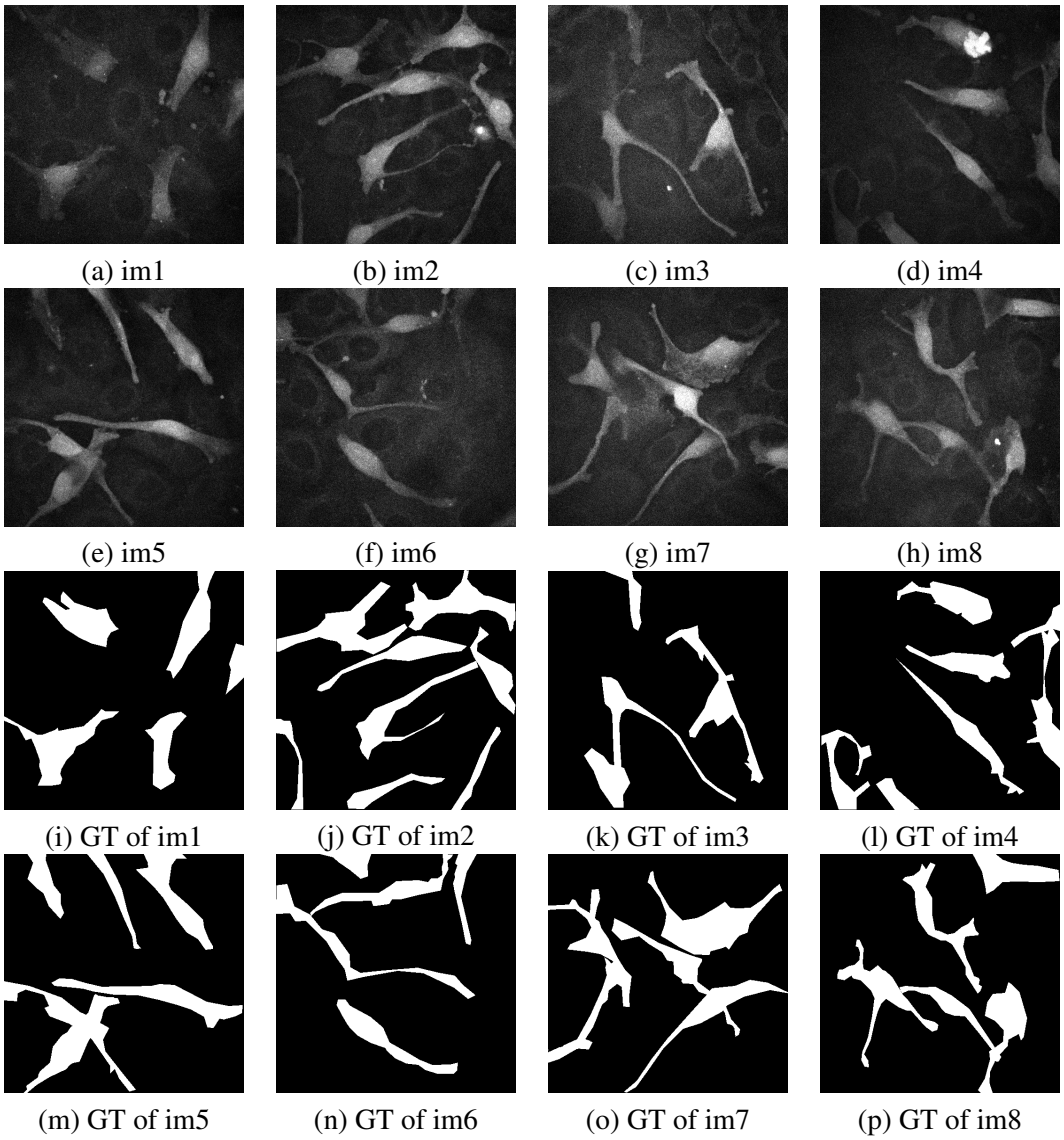


(c) Zoom on a melanocyte



(d) Zoom on a keratinocyte

Figure 3.1: Melanocytes and keratinocytes. *Scheme provided by L'Oréal.*



(a) im1 (b) im2 (c) im3 (d) im4

(e) im5 (f) im6 (g) im7 (h) im8

(i) GT of im1 (j) GT of im2 (k) GT of im3 (l) GT of im4

(m) GT of im5 (n) GT of im6 (o) GT of im7 (p) GT of im8

Figure 3.2: The L'Oréal database $\mathscr{D}_1$.

among four configurations[1] to be representative of the original database and ensure a pertinent new training. What we call the CAMELYON database $\mathscr{D}_2$ is then constituted of 317 images of size 500×500 pixels (217 for train and and 100 for test). Some examples are shown in Fig. 3.3 and Fig. 3.4. Note that two images from train and test subsets respectively must come from two different slides in order to ensure a pertinent evaluation of classification performance. The same remark can be made for cross-validation: two folds should not contain crops, or pixels, coming from the same slide.

Our aim is to segment (and not only detect) **all metastatic regions** in $\mathscr{D}_2$, *i.e.* assign the label 1 to all pixels belonging to a tumoral region and the label 0 otherwise. Groundtruth, performed by a pathologist, are also shown in Fig. 3.4.

As this challenge is recent (April, 2016), and to the best of our knowledge, only participating teams' solutions are available for this database. However, the challenge task consisted in detecting tumoral regions (only one point per region being required) rather than precisely segmenting them. Therefore, no specific method can be used to benchmark our general pipeline(s) in the following.

3.1.3 The Coelho database $\mathscr{D}_3$

This database has been made available by Coelho et al. [2009]. Originally created for a study of pattern unmixing algorithms, it is now often used for benchmarking segmentation methods. It contains 50 images (of size 1349×1030 pixels) of U2OS cell nuclei (human bone osteosarcoma epithelial cells nuclei), acquired by fluorescence microscopy. We split the database into a train (30 images) and test (18 = 20-2 images, two images being discarded as in Coelho et al. [2009]) subsets. For this database, **the aim is to segment the cell nuclei**, as shown in Fig. 3.5.

One specific segmentation method, designed by Thomas Walter, can be used to benchmark our general pipeline(s) in the following. It consists in a three-level Otsu thresholding, after toggle-mapping and median filterings.

Examples of segmentations performed by competing specific methods will be shown together with our general pipeline classification results in this very chapter as well as in the final chapter (Chapt. 7).

3.1.4 Evaluation criteria

Defining what is a good segmentation is often challenging in the field of image analysis. In our framework however, this task happens to be easy as the groundtruth of each image is provided (at least in the *train* and *val* sets of the database). Hence, the more a segmentation resembles its corresponding groundtruth, the higher its quality is. We have chosen five well-known classification evaluation criteria to measure this similarity, namely *precision* P, *recall* R, *F-score* F, *Jaccard index* J and overall pixel *accuracy* Acc. They are defined in the next paragraph.

Let $I : D \to V$ be an image of a given database $\mathscr{D}$, where D is a rectangular subset of $\mathbb{Z}^2$, and V a set of values. Let $I_{GT} : D \to \{0,1\}$ be a groundtruth of I and $I_{RES} : D \to \{0,1\}$ be the result of the classification procedure applied to I. A pixel $x \in D$ is said to be *detected* if $I_{RES}(x) = 1$, *i.e.* if it has obtained the label 1 at the end of the classification process. Then, we define *true positive*,

[1] 1) tumoral tissue only, 2) non-tumoral tissue only, 3) frontier between tumoral and non-tumoral tissues, 4) frontier between tissue and white background. Extraction implemented by Peter Naylor and performed by Thomas Walter, both from CBIO, MINES ParisTech.

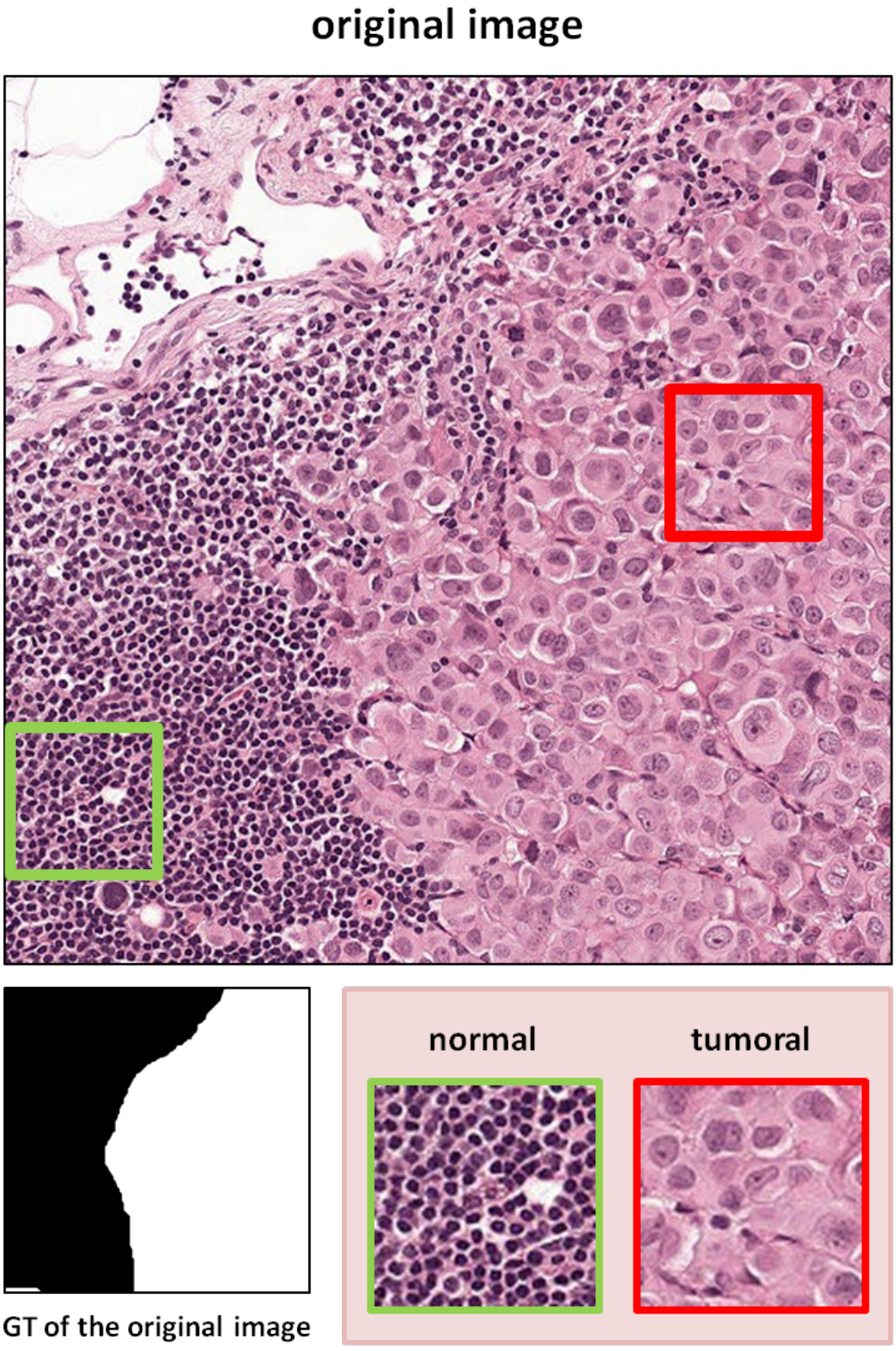
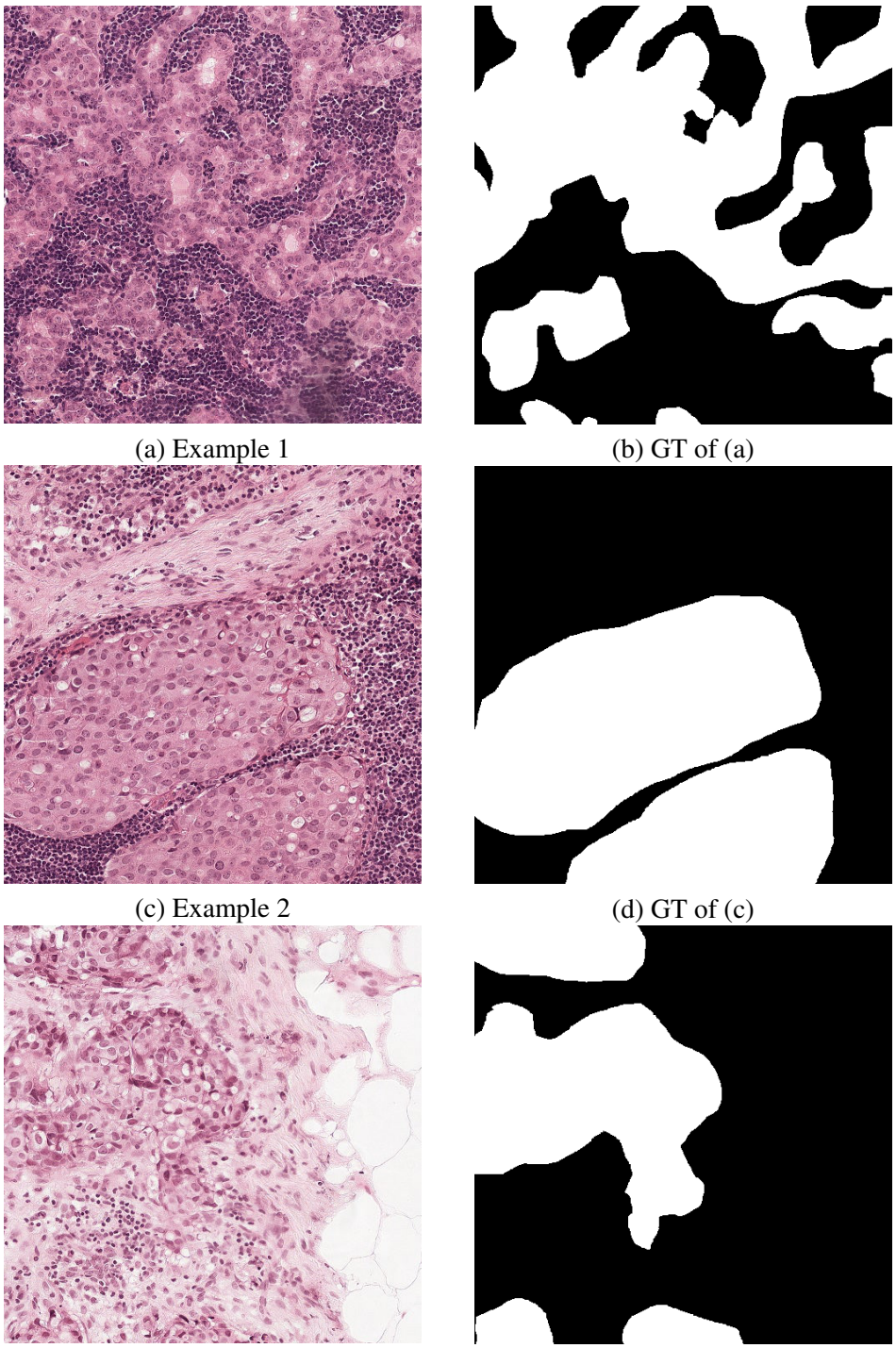


Figure 3.3: One image example from database $\mathscr{D}_2$.



(a) Example 1 (b) GT of (a)

(c) Example 2 (d) GT of (c)

(e) Example 3 (f) GT of (e)

Figure 3.4: Examples from the CAMELYON16 database $\mathscr{D}_2$.

true negative, *false positive* and false negative as follows:

Definition 3.1.1 — True positive (TP). A pixel $x \in D$ is said to be a *true positive* if and only if $I_{RES}(x) = 1$ and $I_{GT}(x) = 1$, *i.e.* it has been detected and it truly corresponds to an object pixel in the groundtruth.

Definition 3.1.2 — True negative (TN). A pixel $x \in D$ is said to be a *true negative* if and only if $I_{RES}(x) = 0$ and $I_{GT}(x) = 0$, *i.e.* it has not been detected and it corresponds to a background pixel in the groundtruth.

Definition 3.1.3 — False positive (FP). A pixel $x \in D$ is said to be a *false positive* if and only if $I_{RES}(x) = 1$ and $I_{GT}(x) = 0$, *i.e.* it has been detected but it does not correspond to an object pixel in the groundtruth.

Definition 3.1.4 — False negative (FN). A pixel $x \in D$ is said to be a *false negative* if and only if $I_{RES}(x) = 0$ and $I_{GT}(x) = 1$, *i.e.* it has not been detected but it corresponds to an object pixel in the groundtruth.

In the following, we call **TP** (respectively **TN**, **FP** and **FN**) the subset of pixels which are true positives (respectively true negatives, false positives and false negatives) among a set of pixels coming from one or more images.

Definition 3.1.5 — Evaluation criteria. Let $\mathscr{P}$ be a set of pixels.

- The precision P of $\mathscr{P}$ gives the proportion of detected pixels which are truly object pixels:

$$P = \frac{|TP|}{|TP| + |FP|} \tag{3.1}$$

- The recall R of $\mathscr{P}$ indicates the proportion of object pixels which have been detected:

$$R = \frac{|TP|}{|TP| + |FN|} \tag{3.2}$$

- The F-score F of $\mathscr{P}$ realizes a trade-off between P and R by taking their harmonic mean:

$$F = \frac{2 \times P \times R}{P + R} \tag{3.3}$$

- The Jaccard index J proposed in Jaccard [1901]: for each image, two sets are considered: the object pixels and the detected pixels. The cardinal of their union and the cardinal of their intersection are computed. The process is repeated and summed over all images constituting the set $\mathscr{P}$. The Jaccard index expresses the ratio between the total cardinal of the intersections over the total cardinal of the unions:

$$J = \frac{vol(intersection)}{vol(union)} \tag{3.4}$$

- The overall pixel accuracy Acc is the percentage of pixels which have been correctly classified:

$$Acc = \frac{1}{|P|} \sum_{p \in P} Ind(p) \tag{3.5}$$

where Ind is a function which equals 1 in p if $I_{res}(p) = I_{DET}(p)$ and 0 otherwise.

To these five criteria evaluating the *classification* performance at the pixel level, we also add one criterion specific to segmentation: the **average number of connected components** Nb_cc. This measure highlights the parcelling out of the detected objects. Indeed, let Nb be the number of objects to be found in an image. We would like Nb_cc to be as close as possible to Nb. The higher Nb_cc is compared to Nb, the more objects are split into different parts, which makes their future analysis more difficult, or even meaningless. This measure is thus important to evaluate the quality of a segmentation.

These six measures are to be applied on a binary image. Note that Random Forests output, for each image to be segmented, a probability map giving for each pixel of the image the probability to belong to class of label 1 (the object class in our case). Thresholding this probability map with different thresholds will give different resulting binary images. For preliminary studies (see Sec. 3.2.2, 6.2.2 and 7.3), we will use an intermediate probability threshold of 0.5 and display examples obtained with this very threshold. For final results, presented in 3.2.3 and 7.4, the six criteria will be expressed as a function of this probability threshold, and thresholds used for illustration images will be specified every time. Last but not least, evaluating performance with varying thresholds allows us to use another usual curve: the receiver operator characteristics (**ROC curve**). This curve is created by plotting the *true positive rate* against the *false positive rate*. The true positive rate (TPR), also called *sensitivity* or *detection rate*, is equal to recall R. The false positive rate (FPR), also know as *1-specifity*, *fall-out* or *probability of false alarm*, is defined by:

$$FPR = 1 - \frac{TN}{TN+FP} = \frac{FP}{TN+FP} \tag{3.6}$$

We are now going to apply the pixel classification pipeline to the three different databases and analyze their results thanks to the six aforementioned criteria.

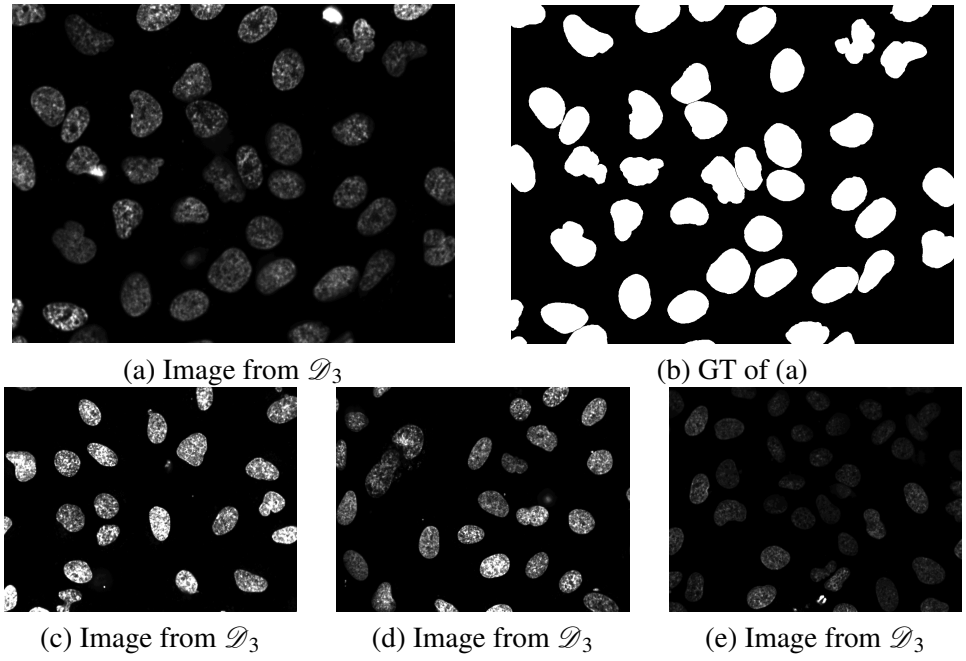
3.2 Assessment of the pixel classification workflow on the three databases

In this section, the pixel classification workflow is assessed with the evaluation procedure described in Sec. 4.3.

3.2.1 Construction of the pixel classification pipeline

As explained in the previous chapter, the first solution we propose in order to address our problematic is to set up the usual pixel classification pipeline. Usually, the different steps are specifically optimized for the considered database to provide the best possible performance. However, in our case, we must ensure that these steps keep general or adapt themselves without requiring an input from the user (as soon as she/he provided the training set). Let us study the dependence in the database for each step of the pipeline :

- The unit of classification (UC), *i.e.* the pixel, exists and is the same for each database. Hence no tuning is required.
- The set of features used to compute the data vector of each UC should be discriminant for the given database and the given segmentation task to offer good performance, that is why it is



(a) Image from $\mathscr{D}_3$ (b) GT of (a)

(c) Image from $\mathscr{D}_3$ (d) Image from $\mathscr{D}_3$ (e) Image from $\mathscr{D}_3$

Figure 3.5: Examples from the Coelho database $\mathscr{D}_3$.

often specifically optimized in the literature. The problem is that a set of features optimized for a given database is likely to perform badly, or at least with decreased performance, on another database. In our case, the idea will be to use a set of general features and rely on random forests to find, for each database, the most discriminant features among them while discarding as much as possible the non-pertinent ones. The advantage is to avoid a manual feature selection step which would require a time-consuming (and sometimes not obvious) input from the user otherwise.

- The parameter values of the random forest (machine learning method) are also to be tuned for each database. However, these parameters can be optimized automatically by cross-validation (nested cross-validation for $\mathscr{D}_1$), so no tuning has to be manually done by the user. Moreover, the subsampling parameters are kept constant for all training phases.

To sum up, only the feature selection procedure must be performed carefully to achieve good performance whatever the database is. We also pay attention to the fact that their computation should not be prohibitive for the user, in particular as far as computation time is concerned. That is why we limit the number of selected features in the pipeline, computing them only on the most informative channel for each database for instance.

As far as the *CS* is concerned, we will focus on two usual supports: the pixel itself and the sliding window centered on this very pixel. Therefore, we will compare the segmentation performance for two different versions of the pipeline:

- Pipeline $\mathscr{P}_{pix}$: pixel classification with *CS* = pixel only.
- Pipeline $\mathscr{P}_{win}$: pixel classification with *CS* = sliding window only.

There is no parameter to be tuned for the *CS* when it is the pixel. For *CS* = sliding window however, we must set its shape and size. We will use four different sizes of square windows to inte-

Pipeline	Family	#	scales		*CS*-scale		=	Sum
$\mathscr{P}_{pix}$	Identity	1	x1		x1	pixel	1	
	MOMA	7	x3	$size_{SE} \in \{1,3,5\}$	x1	pixel	21	
	Gabor	1	x(3x4x3)	$bd \in \{1,2,3\}$	x1	pixel	36	**58**
				$\theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, 3\frac{\pi}{4}\}$				
				$freq \in \{0.3, 0.5, 0.7\}$				
$\mathscr{P}_{win}$	Identity	1	x1		x4	$r \in \{5,10,15,20\}$	4	
	MOMA	7	x3	$size_{SE} \in \{1,3,5\}$	x4	$r \in \{5,10,15,20\}$	84	
	Gabor	1	x(3x4x3)	$bd \in \{1,2,3\}$	x4	$r \in \{5,10,15,20\}$	144	**232**
				$\theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, 3\frac{\pi}{4}\}$				
				$freq \in \{0.3, 0.5, 0.7\}$				

Table 3.1: Recap of the features used for pipelines $\mathscr{P}_{pix}$ and $\mathscr{P}_{win}$. Note: *MOMA* stands for "mathematical morphology operators", *Gabor* for "Gabor filters".

grate multi-scale information (radius $r \in \{5,10,15,20\}$ corresponding to squares of sizes 11×11, 21×21, 31×31 and 41×41 respectively). For operators, we take identity, the seven operators from Mathematical Morphology presented in Chapt. 2, with three scales of structuring element (V6 neighborhood), as well as Gabor filters with different bandwidths bd, directions θ and frequencies $freq$. This leads us to a set of 58 features for $\mathscr{P}_{pix}$ and a set 232 features for $\mathscr{P}_{win}$ (see a recap in Tab.3.1).

Remark: The lowest scale (11×11) is intended to define a support smaller than the smallest object in the image. If it is not the case, *i.e.* if the window is wider than the objects of interest, we advise the user to adapt the image resolution for objects to span more pixels and be more easily detected.

To understand better the behavior of both pipelines, we first conduct a preliminary study on database $\mathscr{D}_1$ with different families of operators.

3.2.2 Preliminary study on $\mathscr{D}_1$

In this subsection, we focus on the L'Oréal database. Random forest parameters are not yet optimized but are set to consistent values to perform the comparison (100 trees, 100 data samples at least in each leaf at the end of training). Since this database contains few images, a leave-one-out procedure is applied to evaluate the segmentation performance.

Quantitative results for both pipelines and for different families of features are presented in Tab. 3.2.

Let us start by the analysis of pipeline $\mathscr{P}_{pix}$'s performance (*CS* = pixel). We can see that using only the identity is not enough to capture the main information, as classification performance is poor: $F = 63\%$, $J = 46\%$ and $Acc = 86\%$. This is not surprising due to the high amount of noise in these images. Moreover, the number of connected components is very high ($Nb\_cc = 8153 \pm 147$), which traduces a poor spatial coherence of detected objects. Indeed, there are, in average, five melanocytes to detect per image. The higher Nb_cc is compared to five, the more detected objects are parceled out, which is not desirable. Using families of features with richer operators, such as MOMA, enables to improve these classification performance, leading to an increase by 14% of the F-score ($F = 72\%$) and 22% of the Jaccard index ($J = 56\%$). The accuracy Acc only increases of 5%, but this measure should be analyzed with caution as we are dealing with a database with unbalanced classes: in the ground truth, 15% of the pixels have been assigned the label 1 and

85% the label 0. We will focus on these two families of operators ("standard" and "MOMA"). Both offer similar classification performance. We can also notice that they notably decrease the number of connected components, even if the latter is still too high to guarantee spatial coherence of detected objects.

For the second pipeline $\mathscr{P}_{win}$, corresponding to *CS* = window (4 scales), morphological operators seem to be more discriminant than standard ones ($F = 75\%$ instead of $F = 72\%$). More importantly, the comparison between both pipelines tells us that **using sliding windows systematically maintains or improves the classification performance compared to using the pixel alone**. For example, the F-score obtained with the identity operator increases by 13% from $\mathscr{P}_{pix}$ to $\mathscr{P}_{win}$. It is also important to note that **sliding windows impact more positively the spatial coherence of detected objects compared to pixels**: for the three families, *Nb_cc* decreases by 95%, 28% and 50% respectively. Therefore, we can conclude, as expected, that sliding windows outperforms the pixels taken alone because these wider supports enable to capture richer and multi-scale information in the neighborhood of the pixel to be classified.

Ω	CS	*P*	*R*	*F*	*J*	*Acc*	*Nb_cc*
Identity	pixel	54	75	63	46	86	8153 ± 147
	windows (4)	60	87	71	55	89	380 ± 160
Standard	pixel	60	87	71	55	89	179 ± 68
	windows (4)	61	87	72	56	89	129 ± 32
MOMA	pixel	61	87	72	56	90	203 ± 71
	windows (4)	**64**	**89**	**75**	**59**	**91**	$\mathbf{101 \pm 33}$

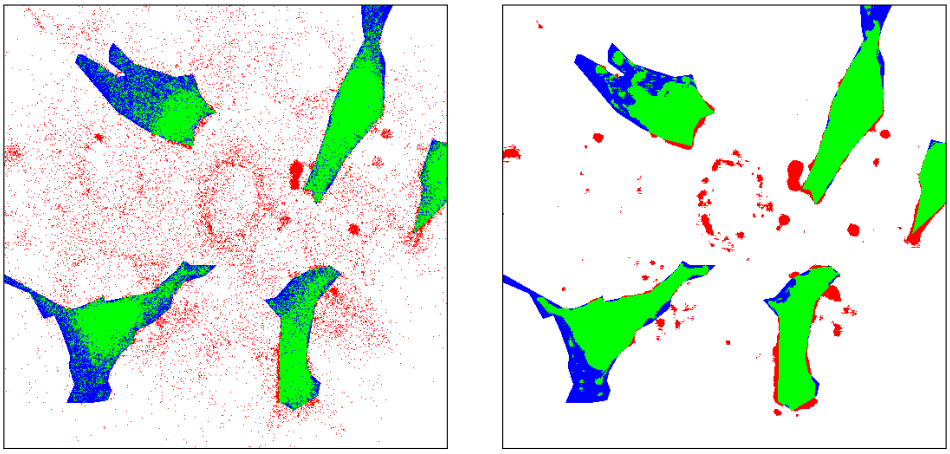
Table 3.2: Pixel classification: comparison between *CS* = pixel and *CS* = windows for database $\mathscr{D}_1$. Results are given on the test subset. Note: *standard* and *MOMA* denote usual operators used by Sommer et al. [2011] and operators from mathematical morphology respectively, all introduced in Chapt. 2.

Figure 3.6 illustrates on an image of $\mathscr{D}_1$ the effect of both pipelines with the three families of features.

3.2.3 Final results on the three databases

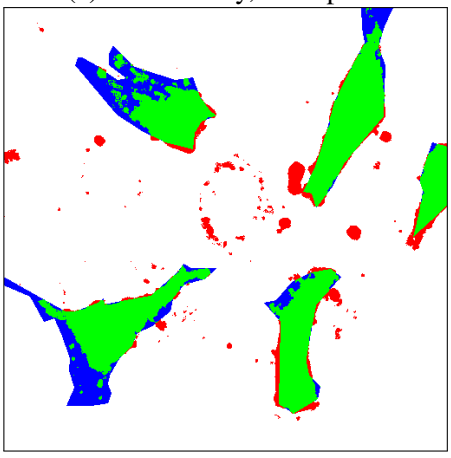
We now evaluate the performance of the two pipelines $\mathscr{P}_{pix}$ and $\mathscr{P}_{win}$ on the three databases $\mathscr{D}_1$, $\mathscr{D}_2$ and $\mathscr{D}_3$. This time, RF parameters are optimized by cross-validation (see resulting values in Tab. A.1 of appendix A.1). The features used are those presented in Tab. 3.1.

Qualitative analysis: Figures 3.7, 3.8 and 3.9 show some examples of classification results for the three databases. In Fig. 3.7, the task is to segment the tumoral region which is in the left part of the image. Corresponding probability maps are presented for $\mathscr{P}_{pix}$ and $\mathscr{P}_{win}$, along with a grey level profile taken along the yellow line. These profiles reveal that it is easier to discriminate the two regions when using the window as *CS* rather than the pixel. Indeed, a wider support captures more information on the texture, a discriminant characteristic in this case. Figure 3.8 presents the probability maps for both pipelines for im1 of $\mathscr{D}_1$ and binary results associated to three thresholds: 0.2, 0.5 and 0.8. The higher the threshold is, the higher the confidence is to be an object pixel. Therefore, increasing the threshold value tends to decrease the number of detected pixels. We can observe that using the window as *CS* decreases the number of detected connected components, which traduces a better spatial coherence than for the pixel-*CS*. Finally, results seem to be rather

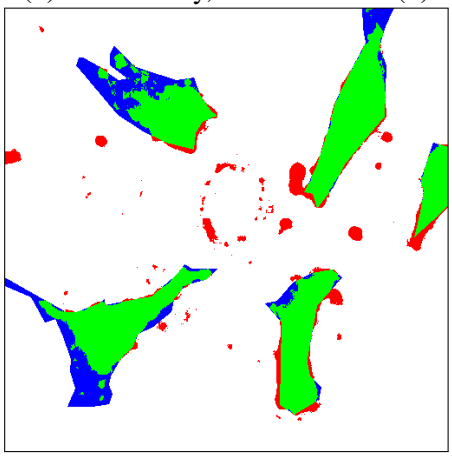


(a) Ω = Identity, *CS* = pixel

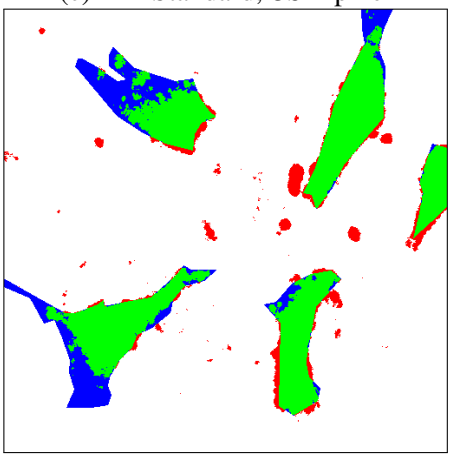
(b) Ω = Identity, *CS* = windows (4)



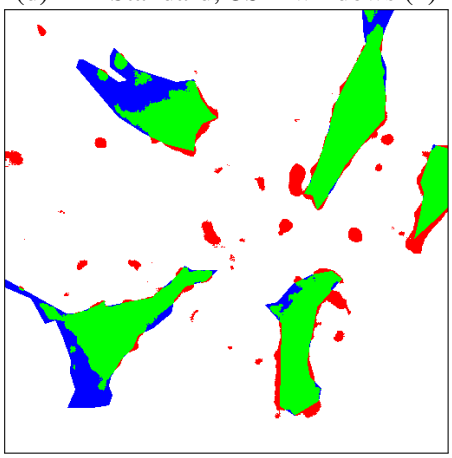
(c) Ω = Standard, *CS* = pixel



(d) Ω = Standard, *CS* = windows (4)



(e) Ω = MOMA, *CS* = pixel



(f) Ω = MOMA, *CS* = windows (4)

Figure 3.6: **Classification results for im1 of** $\mathscr{D}_1$. Legend: TP, TN, FP and FN are shown in green, white, red and blue respectively.

good and equivalent for both pipelines in Fig. 3.9 which is dedicated to an example of $\mathscr{D}_3$.

Quantitative analysis: Figures 3.10, 3.11 and 3.12 are dedicated to the quantitative evaluation of the pipelines on the three databases. $\mathscr{D}_3$ shows the best performance and $\mathscr{D}_2$ is more challenging, but overall performance is already satisfactory for these two general methods. Comparing the computational supports, classification results are better for the window approach, except for $\mathscr{D}_3$ where they are equivalent. This last observation may be explained by the fact that the $\mathscr{D}_3$ database can be easily segmented with reasonable performance at the pixel level (a simple thresholding of pixels with positive values would even be enough, even if not perfect). The window approach is advantageous when contextual information around the pixel, such as texture, is needed to discriminate classes. This is also confirmed by the ROC curves presented in Fig. 3.13. Besides, for databases $\mathscr{D}_1$ and $\mathscr{D}_2$, $\mathscr{P}_{win}$ offers a better spatial coherence, with a smaller number of detected connected components compared to $\mathscr{P}_{pix}$. As far as specific segmentation methods are concerned, they are equivalent ($\mathscr{D}_1$) or outperformed ($\mathscr{D}_3$) by the general approaches embodied by both pipelines.

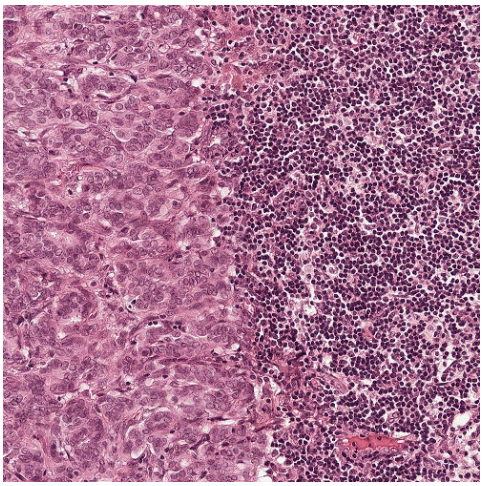
3.3 Conclusion

To sum up, we have seen that, when performing pixel classification, using a window instead of the pixel itself as *CS* achieves better performance for several reasons:

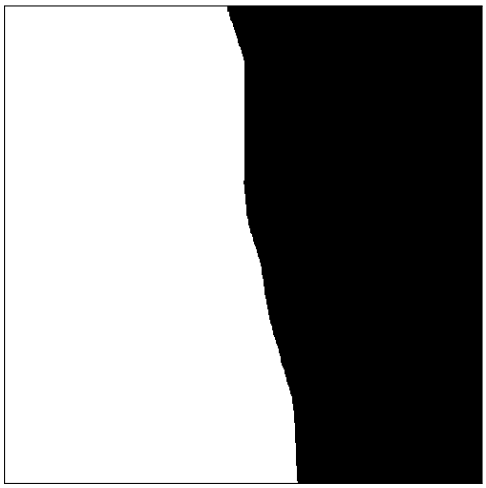
- Using a wider support enables to capture richer information on the pixel to be classified.
- Using several sizes of windows enables to capture multi-scale information, which is all the more valuable in our case as databases can present objects of different sizes and shapes.

Yet, even the results obtained by calculating features in a sliding window could be further improved, regarding classification as well as spatial coherence of detected objects. In particular, this second approach mainly suffers from misclassifications on contours, as illustrated in Fig. 3.14. Figure 3.14.b is a crop on a contour of the simplified (filtered) image 3.14.a. Let us suppose that we would like to classify a pixel (shown in black in Fig. 3.14.c) with a feature consisting in taking the mean intensity value over the sliding window drawn in yellow. We can see that if the window had been entirely included in the white object or in the black background, the mean intensity would have been really close to the white or to the black respectively; thus the pixel would have been easily classified as belonging to one or the other. However, for pixels lying close to a contour as in this case, the feature's value is not representative of either class (see Fig. 3.14.d), which leads to a higher misclassification rate in this area. **This contour issue is due to the fact that the computational support does not adapt to the image content.**

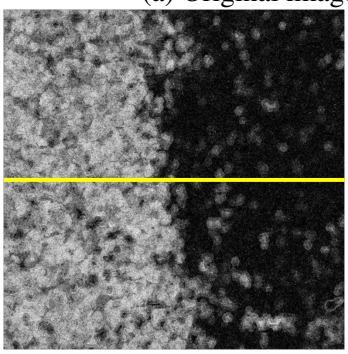
In the following, we decide to focus on another computational support which is wider than the pixel and which adapts to the image content: the *superpixel*. In the next part, we will expound how to generate such a support. Eventually, we will study, in Part 3, the pertinence of its insertion in our general classification pipeline.



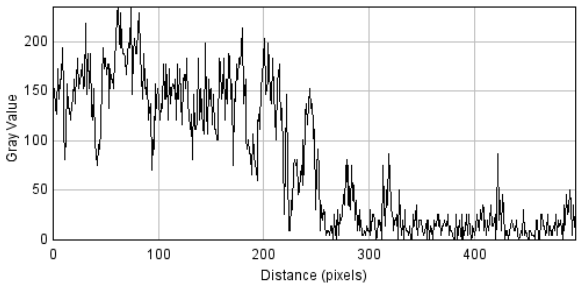
(a) Original image



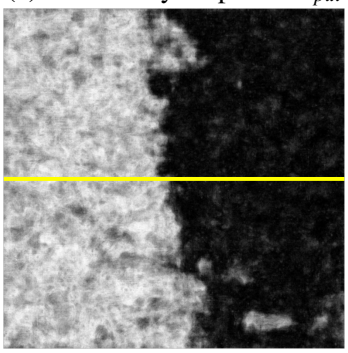
(b) GT of (a)



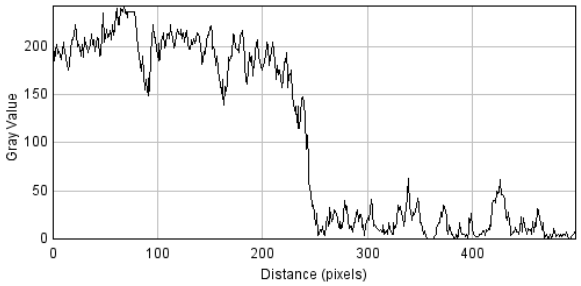
(c) Probability map for $\mathscr{P}_{pix}$



(d) Profile for $\mathscr{P}_{pix}$

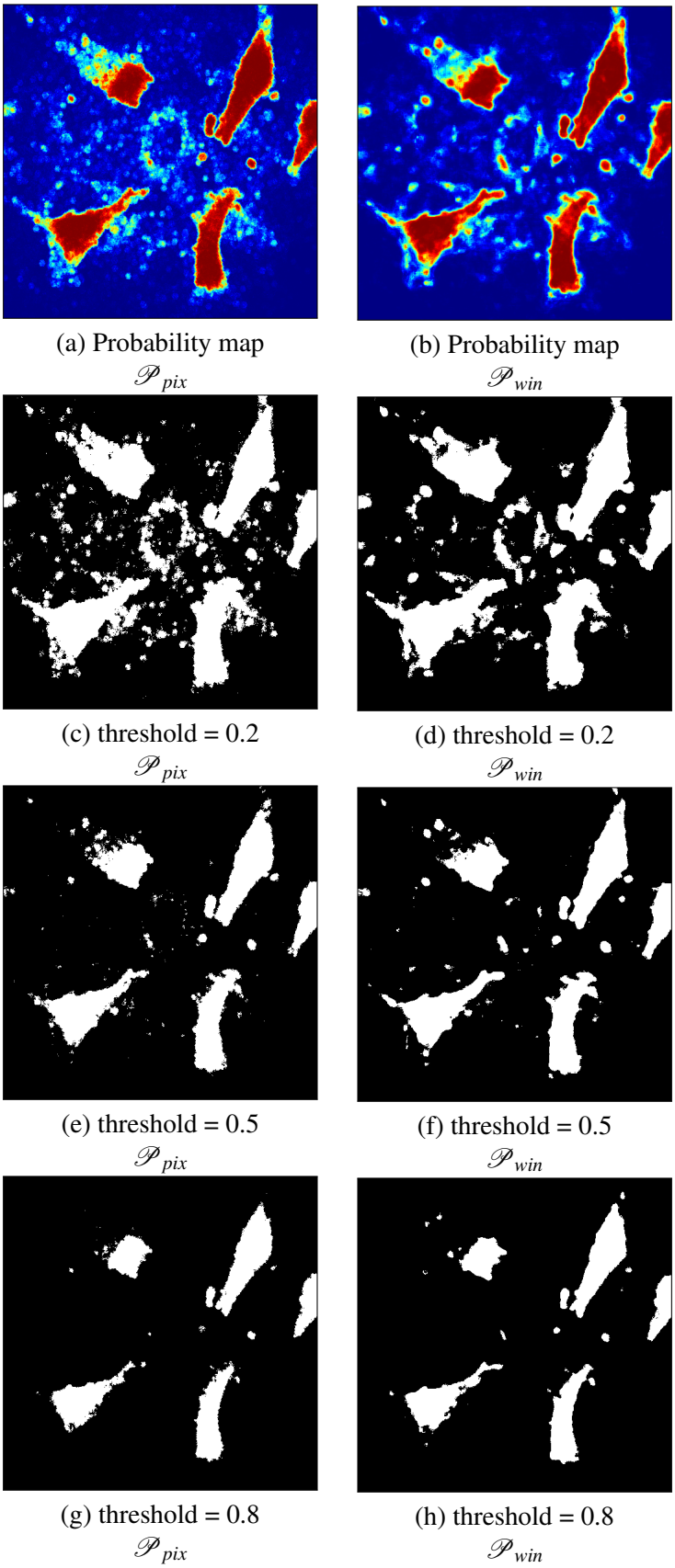


(e) Probability map for $\mathscr{P}_{win}$



(f) Profile for $\mathscr{P}_{win}$

Figure 3.7: **Classification results for an image of $\mathscr{D}_2$.** Grey level profiles are evaluated along the yellow line.



(a) Probability map $\mathscr{P}_{pix}$

(b) Probability map $\mathscr{P}_{win}$

(c) threshold = 0.2 $\mathscr{P}_{pix}$

(d) threshold = 0.2 $\mathscr{P}_{win}$

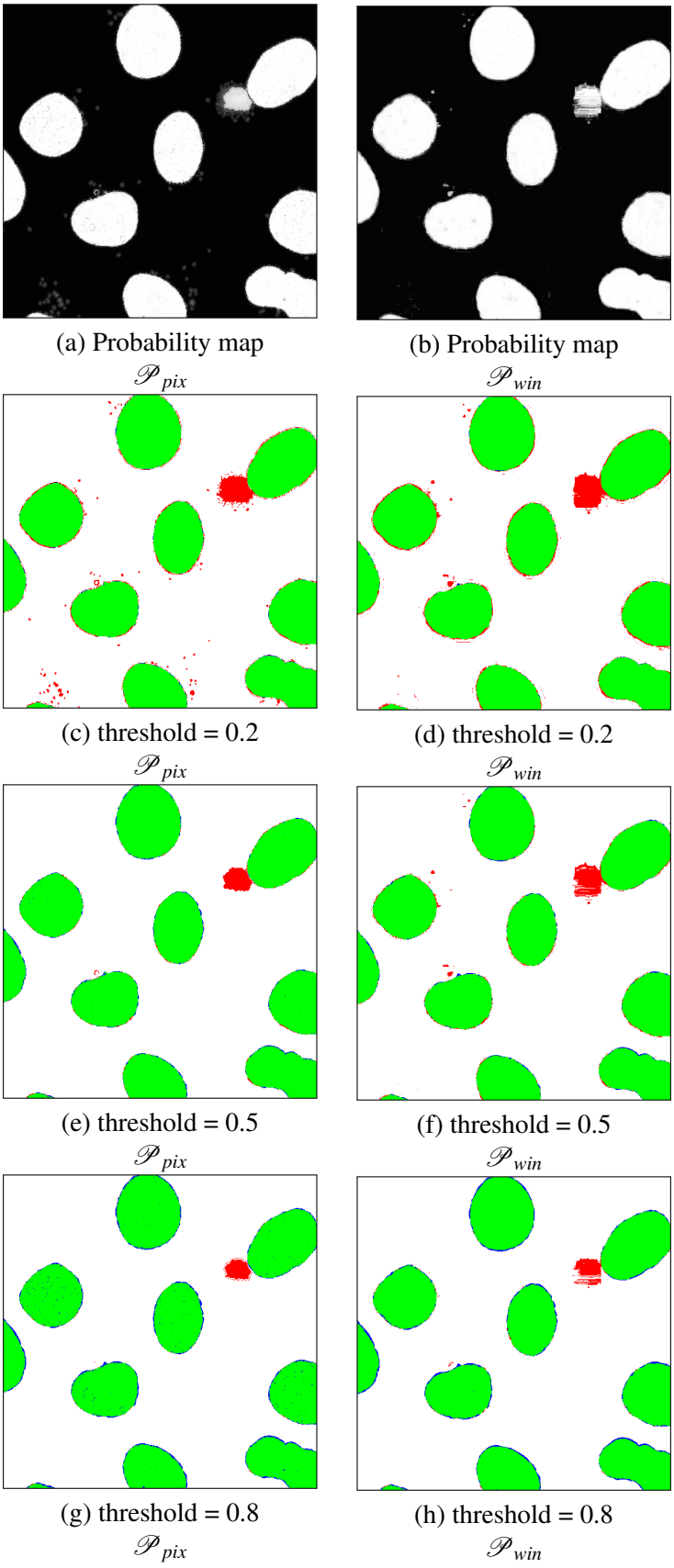
(e) threshold = 0.5 $\mathscr{P}_{pix}$

(f) threshold = 0.5 $\mathscr{P}_{win}$

(g) threshold = 0.8 $\mathscr{P}_{pix}$

(h) threshold = 0.8 $\mathscr{P}_{win}$

Figure 3.8: **Classification results on im1 of $\mathscr{D}_1$.**



(a) Probability map $\mathscr{P}_{pix}$

(b) Probability map $\mathscr{P}_{win}$

(c) threshold = 0.2 $\mathscr{P}_{pix}$

(d) threshold = 0.2 $\mathscr{P}_{win}$

(e) threshold = 0.5 $\mathscr{P}_{pix}$

(f) threshold = 0.5 $\mathscr{P}_{win}$

(g) threshold = 0.8 $\mathscr{P}_{pix}$

(h) threshold = 0.8 $\mathscr{P}_{win}$

Figure 3.9: **Final classification results on a crop of image *dna-30* of $\mathscr{D}_3$.** Legend: TP, TN, FP and FN are shown in green, white, red and blue respectively.

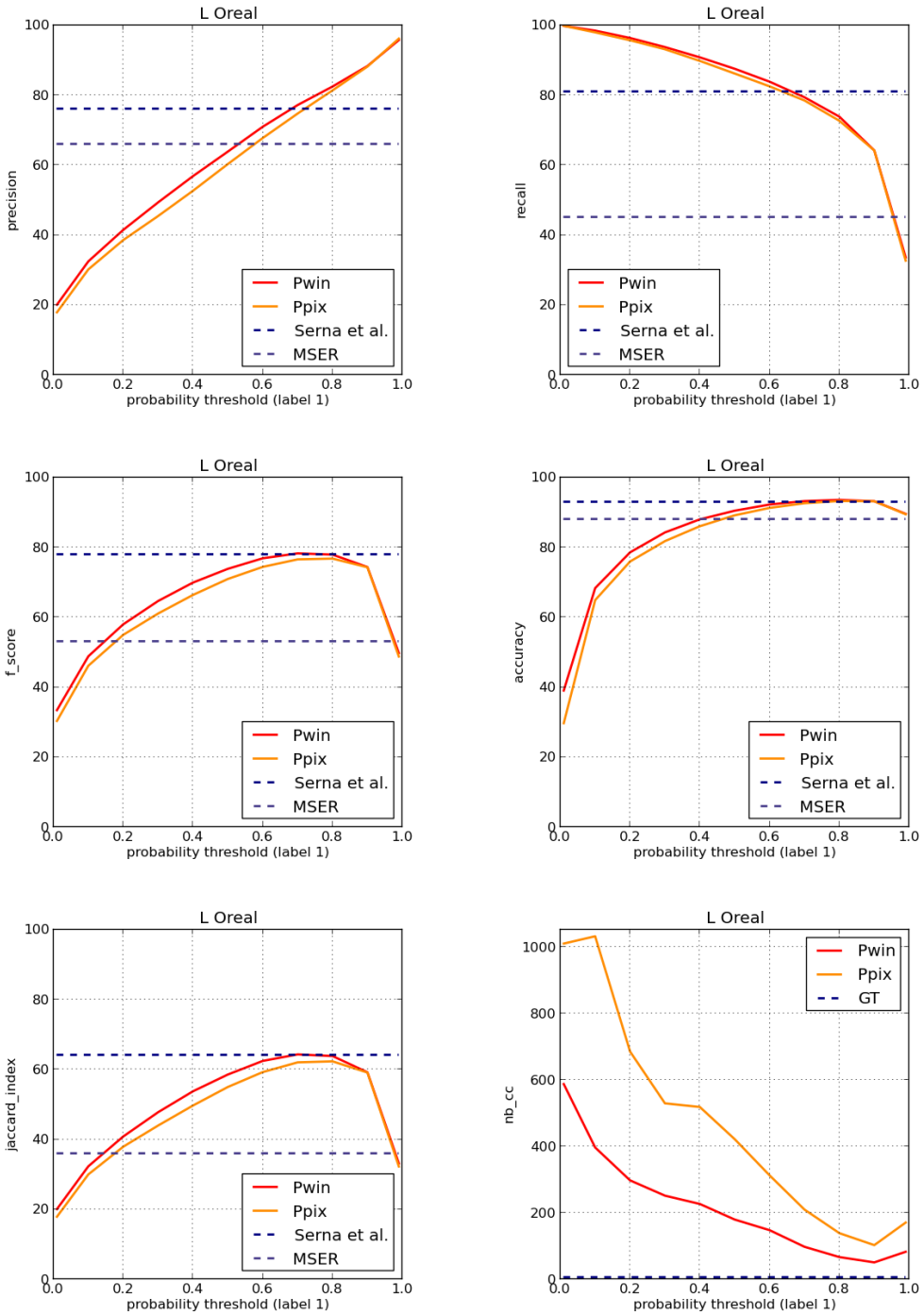


Figure 3.10: Quantitative results for $\mathscr{D}_1$.

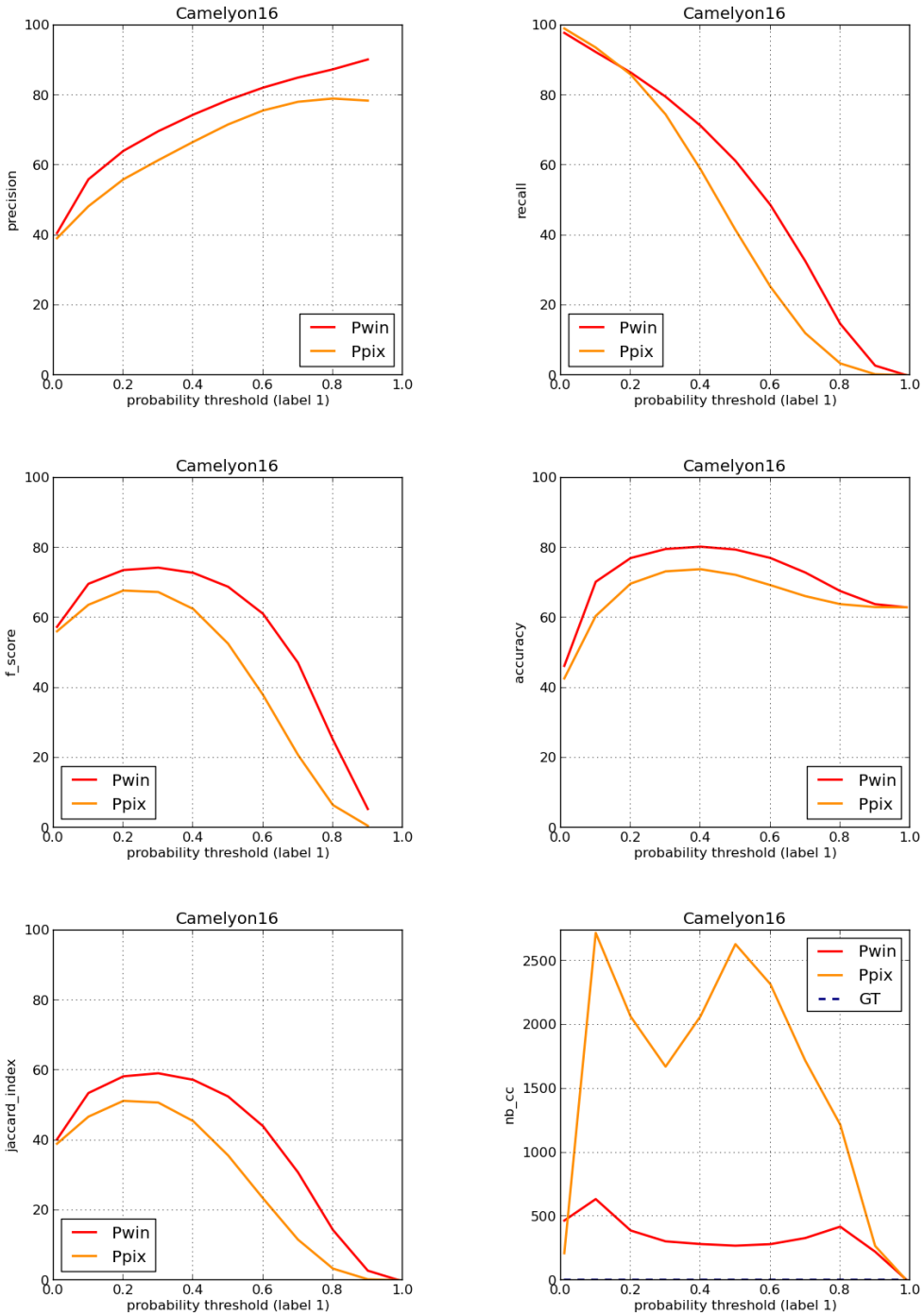


Figure 3.11: Quantitative results for $\mathscr{D}_2$.

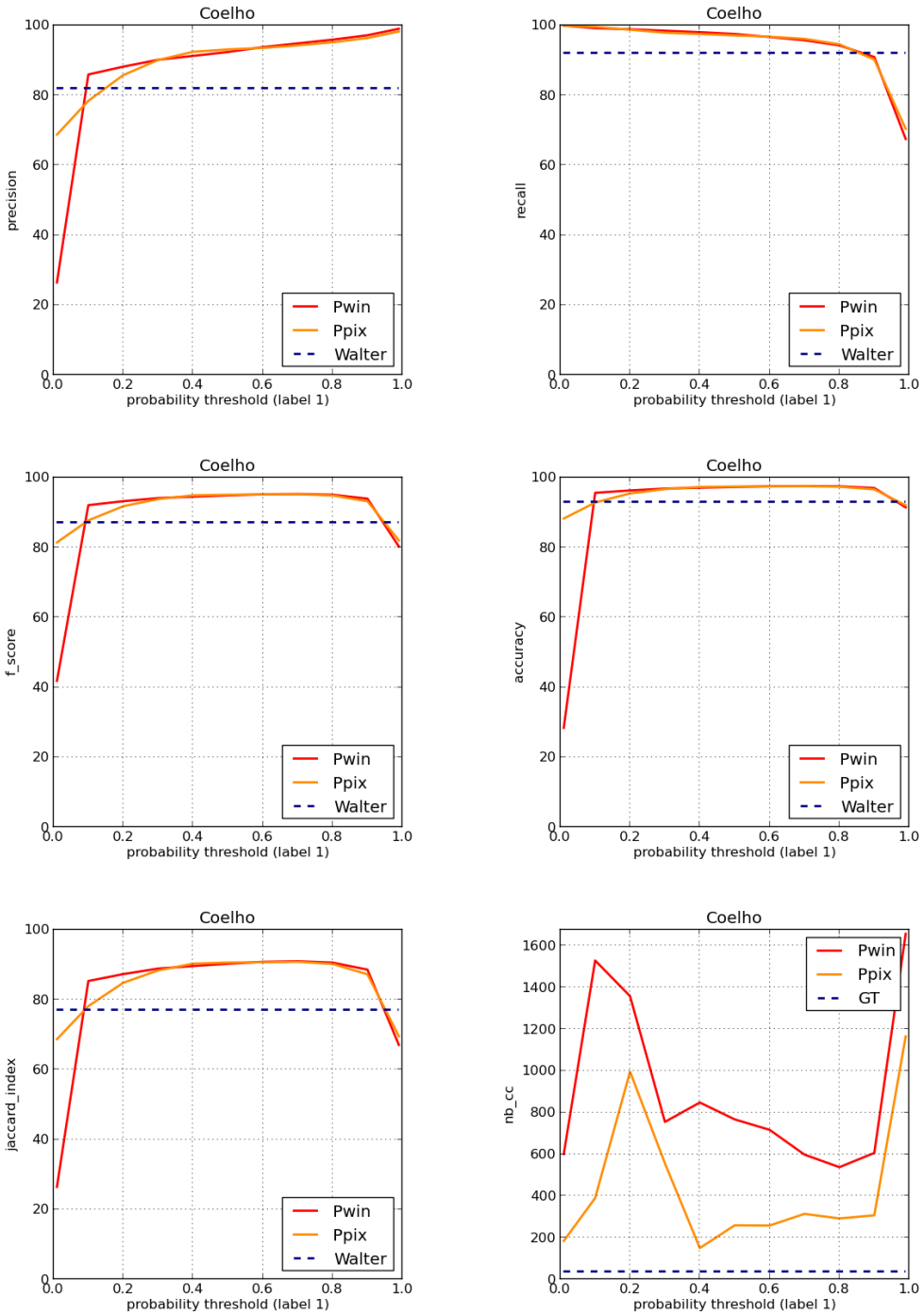
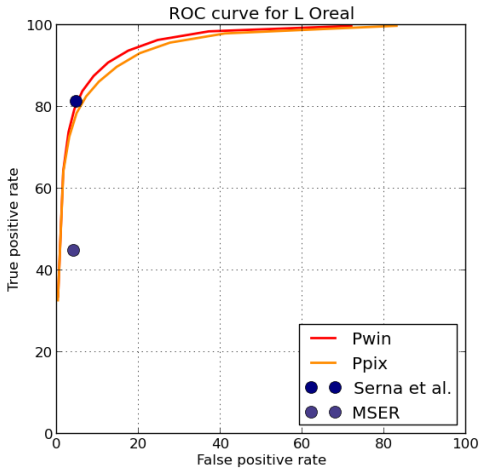
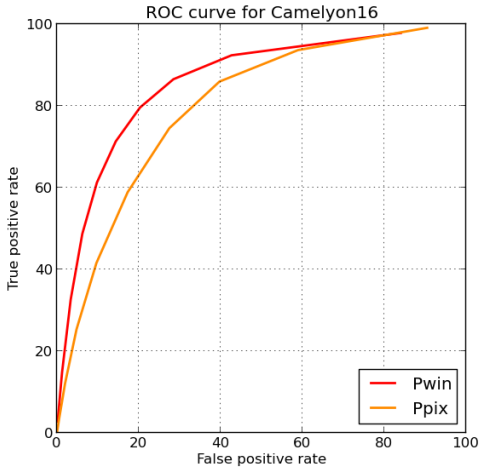


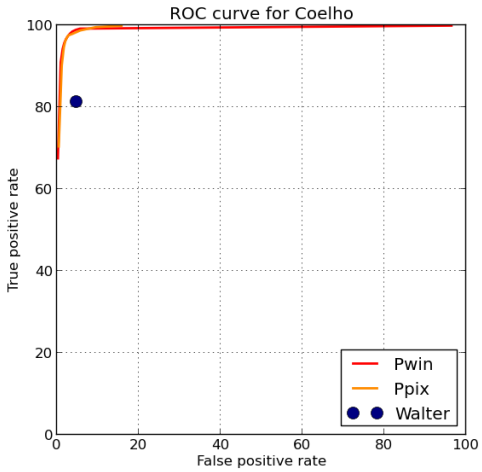
Figure 3.12: Quantitative results for $\mathscr{D}_3$.



(a) ROC curve for $\mathscr{D}_1$

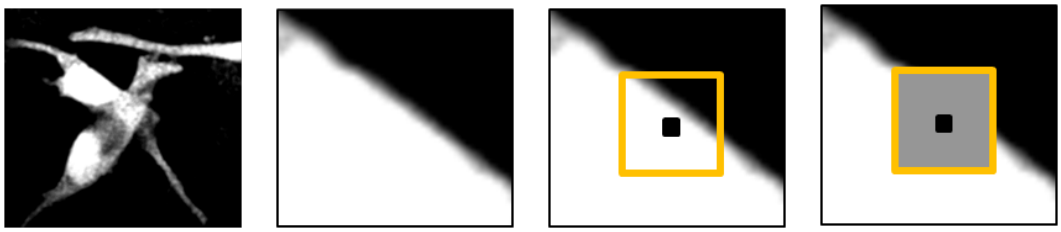


(b) ROC curve for $\mathscr{D}_2$



(c) ROC curve for $\mathscr{D}_3$

Figure 3.13: ROC curves for the three databases.



(a) crop from im5 (b) zoom of (a) (c) sliding window (d) feature computation

Figure 3.14: Sliding windows: the contour issue.

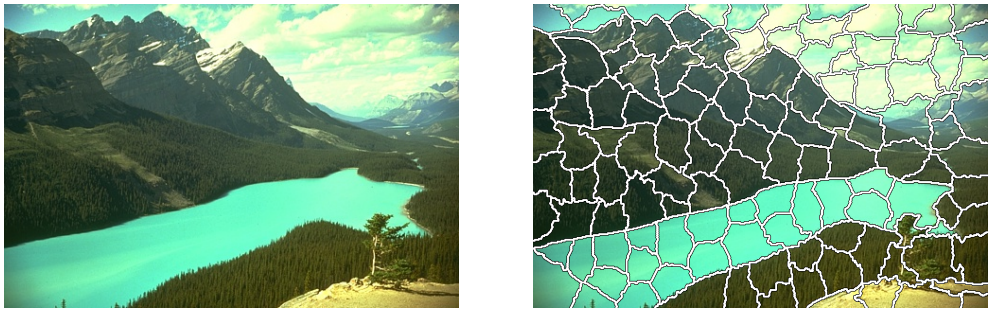
II Waterpixels

*****

In this part, we focus on **superpixels**, a special case of low-level segmentation.

We present the properties they should satisfy, the state-of-the-art of methods enabling to generate them, as well as how to evaluate their performance.

We then propose a new generation method based on the watershed transformation. The resulting superpixels are hence called ***waterpixels***. In addition to their good quality performance, they offer many advantages such as their low computation time and their easiness of generation (few, understandable, parameters to be tuned by the user), which makes them competitive with the state-of-the-art.



(a) Original image (b) Superpixels

Figure 3.15: Illustration of superpixels (waterpixels)

*****

Related Publications and Conferences

- *Waterpixels*
  V. Machairas, M. Faessel, D. Cardenas-Pena, T. Chabardes, T. Walter and E. Decencière
  IEEE Transaction on Image Processing, 24(11):3707–3716, 2015

- *Waterpixels: Superpixels Based on The Watershed Transformation*
  V. Machairas, T. Walter and E. Decencière
  International Conference on Image Processing (ICIP), pp. 4343-4347, 2014

- *Spatial Repulsion Between Markers Improves Watershed Performance*
  V. Machairas, E. Decencière and T. Walter
  International Symposium on Mathematical Morphology (ISMM), pp. 194–202, 2015

- *Watervoxel Representation for Supporting MRI Volume Segmentation*
  D. Cardenas-Pena, V. Machairas, G. Castellanos-Dominguez, E. Decencière, T. Walter
  Journée Géodis, Reims, France, 2014

- *Waterpixels: Superpixels Partition from Watershed Transformation*
  V. Machairas, E. Decencière and T. Walter
  37th session of ISS France, 2014

4 Superpixels: A special case of low-level segmentation

It was, he thought, the difference between being dragged into the arena to face a battle to the death and walking into the arena with your head held high. Some people, perhaps, would say that there was little to choose between the two ways, but Dumbledore knew [...] that there was all the difference in the world.
J.K. Rowling, Harry Potter and the Half-Blood Prince.

Résumé

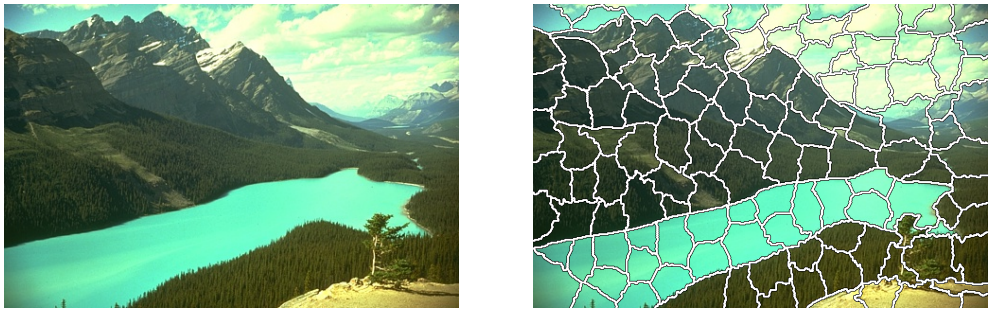
Dans ce chapitre, nous donnons la définition des superpixels et les propriétés essentielles qu'ils doivent remplir, notamment vis-à-vis de leur utilisation future dans des pipelines d'analyse d'image comme la classification. Nous présentons ensuite un certain nombre de méthodes qui ont été proposées dans la littérature pour générer efficacement des superpixels, en mettant en avant leurs avantages et leurs inconvénients. Nous rappelons ce qu'est la ligne de partage des eaux (*watershed* en anglais) et expliquons pourquoi nous pensons que c'est un outil pertinent pour générer efficacement des superpixels de manière compétitive. Nous terminons en présentant comment évaluer la qualité d'une partition en superpixels.

*****

This chapter is dedicated to the concept of *superpixels* (SPs), a special-case of low-level segmentation. We present the properties they should satisfy, the state-of-the-art of methods enabling to generate them, as well as how to evaluate their performance. In the next chapter, we will propose a new generation method: *waterpixels* (WP).

4.1 Definition and properties

We have seen that a classical approach to segment an image is to perform a classification of each of its pixels. The term *superpixels* has been chosen in the literature to express the fact that we can use "bigger pixels" instead of single pixels for the same task. Indeed, pixels are the result of the sensors we use and hence do not convey a perceptual meaning. The idea is then to group together pixels in order to obtain a new semantic unit. The natural way to achieve such a partition is to perform a low-level segmentation of the image (see Fig. 4.1). Moreover, recent methods highlight the need for shape and size similarity between such regions. There exist different variants of superpixels' characteristics in the corresponding and abundant literature. In the following, we will



(a) Original image (b) Superpixels

Figure 4.1: Illustration of superpixels (waterpixels)

refer to, as superpixels, regions which have the following properties:

1. **Connected partition:** the SPs constitute a partition of the image and each SP is made of a single connected component.

2. **Homogeneity**: pixels of a given SP should present similar colors or grey levels.

3. **Adherence to object boundaries**: objects boundaries should be included in SPs boundaries.

4. **Regularity**: SPs should form a regular pattern on the image.

5. **Easiness of computation**: SPs partition should be fast to compute and easily tuned by the user (few, understandable, parameters to control SP characteristics).

Indeed, superpixels are thought of as "pieces of an object" which will be assigned a unique object label. Therefore, properties 1 to 3 are required to avoid a superpixel from containing pixels lying on different objects (which is the case when it overlaps a contour between two objects, or is likely to happen when it is split into spatially separated connected components). On the other hand, property 4 enables to perform pertinent comparison of feature values for learning. Of course, the requirements on regularity and boundary adherence are to a certain extent oppositional, and a good generation method must aim at finding a compromise between both of them. Last but not least, property 5 is often an essential aspect (as in our case), as SPs are typically only the first step of an often complex and potentially time consuming workflow.

4.2 Related work

Low-level segmentations have been used for a long time as a first step towards segmentation (see the work of Monga [1987] and Marcotegui and Meyer [1997] for example). The term superpixel was coined much later by Ren and Malik [2003], albeit in a more constrained framework. This approach has raised increasing interest since then. Various methods exist to compute regular and non-regular SPs: Normalized cuts by Shi and Malik [2000] and Ren and Malik [2003], Mean Shift by Comaniciu and Meer [2002], graph-based superpixels by Felzenszwalb and Huttenlocher [2004], TurboPixels by Levinshtein et al. [2009], Veksler et al. [2010], Zeng et al. [2011], VCells by Wang and Wang [2012], SLIC by Achanta et al. [2012], Shen et al. [2014], or Duan and Lafarge [2015] to cite just a few. We have also proposed a new method called Waterpixels (notation : *WP*, see Machairas et al. [2014] and Machairas et al. [2015]) which is presented in the next chapter.

These methods offer various properties and varying performance. Some are based on graphs (Felzenszwalb and Huttenlocher [2004]), others on geometrical flows (Levinshtein et al. [2009]) or on k-means (Achanta et al. [2012]). Some only produce non-regular SPs or show high complexity. For our application, we need efficient and regular SPs that are as fast as possible to compute. That is why, in the following, we focus on linear complexity methods generating regular SPs: TurboPixels by Levinshtein et al. [2009], superpixels by Zeng et al. [2011], VCells by Wang and Wang [2012] and SLIC by Achanta et al. [2012].

Remark: Actually, the method proposed by Zeng et al. [2011] enables to generate superpixels which are similar in shape but not in size. Their superpixels aim at being compact but their size adapts to the image content (larger in homogeneous area, smaller in more textured regions). If this property seems appealing at first sight with regards to segmentation purposes, it is nevertheless non desirable in our application as we need similarity of supports to provide a fair comparison of feature values. Moreover, some parametrized feature families, such as Haralick features, may not be defined for regions which are small compared to the chosen parameter. A large variability in terms of superpixel size would therefore necessarily lead to missing values, which is not desired. Yet, these superpixels are presented here as they aim at coping with the trade-off between boundary adherence and compactness with state-of-the-art lowest complexity.

4.2.1 State-of-the-art generation methods

Methods for SP generation are all based on two steps: an initialization step where either seeds or a starting partition are defined and a (potentially iterative) assignment step, where each pixel is assigned to one superpixel, starting from the initialization.

Method	Levinshtein 2009	Wang 2012	Zeng 2011	Achanta 2012	WP 2014
Generation type	1	2	1 (iterated)	2	1
Seed type	A	C	C	C	B
Exact control on number of SPs	yes	yes	yes	no	yes
Control on regularity	no	yes	no	yes	yes
Post-processing free	no	no	no	no	yes
Complexity	$O(n)$	$O(in\sqrt{N})$	$O(in)$	$O(n)$	$O(n)$

Table 4.1: Recap chart of existing methods to compute regular superpixels (n is the number of pixels in the image; i is the number of iterations required; N the number of superpixels). "WP" corresponds to our method, called "Waterpixels" (presented in the next chapter).

Choosing the seeds

In the first step, a set of seeds is chosen, which are typically spaced regularly over the image plane and which can be either regions or single pixels:

- Type A seeds are independent of the image content. These are typically the cells or the centers of a regular grid.
- Type B seeds depend on the content of the image (compromise between a regular cover of the image plane and an adaption to the contour).
- Type C seeds are initially image independent, then they are iteratively refined to take into account the image contents.

If the seeds do not depend on the image, an iterative refinement is usually preferable, and therefore more time is spent on the computation of the SP. Type B methods may spend more time on finding appropriate seeds, but can therefore afford not to iterate the SP generation.

Building superpixels from seeds

In the second step, the partition into superpixels is built from the seeds. Among the methods with linear complexity, there are two main strategies for this:

Shortest Path methods (type 1) [Levinshtein et al. [2009]; Zeng et al. [2011]]: these methods are based on region growing: they start from a set of seeds (points or regions) and successively extend them by incorporating pixels in their neighborhood according to a usually image dependent cost function until every pixel of the image plane has been assigned to exactly one superpixel. This process may or may not be iterated.

Shortest Distance methods (type 2) [Wang and Wang [2012]; Achanta et al. [2012]]: these are iterative procedures inspired by the field of unsupervised learning, where at each iteration step, seeds (such as centroids) are calculated from the previous partition and pixels are then re-assigned to the closest seed (like for example the k-means approach).

Even though methods inspired by general clustering methods (type 2) seem appealing at first sight, in particular when they globally optimize a cost function, this class of methods does not guarantee connectivity of the superpixels for arbitrary choices of the pixel-seed distance (see Achanta et al. [2012],Wang and Wang [2012]). For instance, the distance metric proposed in Achanta et al. [2012] (a combination of Euclidean and color distance), leads to non-connected superpixels, which is undesirable. To tackle this issue, a post-processing step is necessary, consisting either in relabeling the image so that every connected component has its own label (see Wang and Wang [2012]), leading to a less regular distribution of SP sizes and shapes, or in reassigning isolated regions to the closest large enough superpixel, as in Achanta et al. [2012], leading to non-optimality of the solution. In both cases, the number of superpixels becomes unpredictable. In addition, such post-processing increases the computational cost and can turn out to be the most time-consuming step when the image contains numerous small objects/details compared to the average size of the superpixels.

By contrast, methods based on region growing (type 1) inherently implement a "path-type" distance, where the distance between two pixels does not only depend on the value and position of the pixels themselves, but on values and positions along the path connecting them. Type 1 methods yield connected superpixel regions, for which the number of superpixels is exactly the number of seeds.

Other properties

It is generally accepted that a good superpixel-generation method should provide to the user total control over the number of resulting superpixels. While this property is achieved by Levinshtein et al. [2009], Veksler et al. [2010], Wang and Wang [2012] and Zeng et al. [2011], some only reach approximatively this number because of post-processing (either by splitting too big superpixels, or removing small isolated superpixels as in Achanta et al. [2012]). Another parameter is the control on superpixels regularity in the trade-off between regularity and adherence to contours. Only Wang and Wang [2012] and Achanta et al. [2012] enable the user to weight the importance of regularity compared to boundary adherence, so that it can be adapted to the application.

As far as performance is concerned, one of the main criteria is undoubtedly the complexity

that the method requires. Indeed, for superpixels to be used as primitives for further analysis such as classification, their computation should neither take too long nor too much memory. This is the reason why we focus on linear complexity methods. Among them, SLIC, proposed by Achanta et al. [2012], appears to offer the best performance with regards to the trade-off between adherence to boundaries and regularity. Moreover, since its recent inception, this method has become very popular in the computer vision community. We will therefore use it as reference for the quantitative evaluation of our method (see Chapt. 5).

4.2.2 Superpixels and watershed

In principle, the watershed transformation (see Soille [2003] for a review) is well suited for SP generation:

1. It yields a good adherence to object boundaries when computed on the image gradient.
2. It allows to control the number and spatial arrangement of the resulting regions through the choice of markers.
3. The connectivity of resulting regions is guaranteed and no postprocessing is required.
4. It offers linear complexity with the number of pixels in the image.

Indeed, it has been used to produce low-level segmentations in several applications, including computationally intensive 3D applications (see for instance the work of Andres et al. [2008] and Stawiaski and Decencière [2008]) and in Hanbury [2008], in particular when shape regularity of the elementary regions was not required.

Previous publications claimed that the watershed transformation does not allow for the generation of spatially regular SP (Levinshtein et al. [2009]; Achanta et al. [2012]). Recently, we and others (Machairas et al. [2014], Machairas et al. [2015], Neubert and Protzel [2014], Benesova and Kottman [2014], Hu et al. [2015]) have shown that in principle the watershed transformation can be applied to SP generation. In the next chapter, we will introduce waterpixels, a family of methods based on the watershed transformation to compute superpixels.

4.3 Evaluation procedure to assess superpixel performance

In this section, we propose an evaluation procedure to assess the performance of any superpixel partition. It will be used in the next chapter to evaluate our method as well as state-of-the-art ones.

Database

Most of state-of-the-art methods are evaluated on the Berkeley segmentation database, introduced in Martin et al. [2001]. This database is divided into three subsets, “train”, “test” and “val”, containing respectively 200, 200 and 100 images of sizes 321×481 or 481×321 pixels. Approximately 6 human-annotated ground-truth segmentations are given for each image. These ground-truth images correspond to manually drawn contours.

Evaluation criteria

SP methods produce an image partition $\{s_i\}_{1 \leq i \leq N}$. In order to compute the SP borders, we use a morphological gradient with a 4-connectivity neighborhood. Note that the resulting contours are two pixels wide. To this set of contours S_c, we add the one pixel wide image borders S_b. The final set is denoted C. The ground truth image corresponding to the contours of the objects to be segmented, provided in the Berkeley segmentation database, is called GT.

In superpixel generation, we look for an image decomposition into regular regions that adhere well to object boundaries. We propose to use three measures to evaluate this trade-off, namely boundary-recall, contour density and average mismatch factor, as well as computation time. If

boundary-recall and computation time are classic measures, we also add contour density and mismatch factor to better estimate superpixel performance.

There are two levels of regularity: (1) the number of pixels required to describe the SP contours, which can be seen as a measure of complexity of individual SP, and (2) the similarity in size and shape between SP.

The first property is evaluated by the Contour Density, which is defined as the number of SP contour pixels divided by the total number of pixels in the image:

$$CD = \frac{\frac{1}{2}|S_c| + |S_b|}{|D|} \tag{4.1}$$

Note that $|S_c|$ is divided by 2 since contours are two-pixel-wide.

The second property, *i.e.* similarity in size and shape, is evaluated by an adapted version of the mismatch factor of Strachan et al. [1990]. The mismatch factor measures the shape and size dissimilarity between two regions. Given two sets, A and B, the mismatch factor mf between them is defined as:

$$\begin{aligned} mf(A,B) :=& \frac{|A \cup B \setminus A \cap B|}{|A \cup B|} \\ =& 1 - \frac{|A \cap B|}{|A \cup B|} \end{aligned} \tag{4.2}$$

The mismatch factor and the Jaccard index thus sum to one. Aiming to measure the superpixel regularity, we adapted the mismatch factor to estimate the spread of size and shape distribution. Hence, the average mismatch factor MF is proposed as:

$$MF = \frac{1}{N} \sum_{i=1}^{N} mf(s_i^*, \widehat{s}^*) \tag{4.3}$$

where s_i^* is the centered version of superpixel s_i, and $\widehat{s}^*$ is the average centered shape of all superpixels. The complete definition of the average mismatch factor is given in appendix A.2.

Note that although compactness is sometimes used in superpixel evaluation (see Schick et al. [2014]), it is a poor measurement for region regularity. For example, perfectly-rectangular regions are regular but not compact (because they are different from discs). Our method, *waterpixels*, can in principle tend towards differently shaped superpixels (rectangles, hexagons or other). Since the average mismatch factor compares each superpixel against an image dependent template, this measure is more appropriate for evaluating regularity than compactness.

To quantify the adherence to object boundaries, a classical measure used in the superpixel literature is the boundary-recall (BR). It is similar as the recall measure R introduced in Chapt. 3 but it is, this time, expressed for contour pixels (*label* 1) vs. non contour pixels (*label* 0). Boundary-recall is now defined as the percentage of ground-truth contour pixels GT which fall within strictly less than 3 pixels from superpixel boundaries C:

$$BR = \frac{|\{p \in GT, d(p,C) < 3\}|}{|GT|} \tag{4.4}$$

where d is the L_1 (or Manhattan) distance.

While precision cannot be directly used in the context of over-segmentations, boundary-recall has to be, in this particular case of superpixels, interpreted with caution. Indeed, as also noted by Kalinin and Sirota [2015], very tortuous contours systematically lead to better performance: because of their higher number, SP contour pixels have a higher chance of matching a true contour, artificially increasing the boundary-recall. Hence, we propose to always consider the trade-off between boundary-recall and contour density to properly evaluate the adherence to object boundaries, penalizing at the same time the cost in pixels to describe SP contours.

Finally, computation time is also evaluated. The lower it is, the better it is, since superpixels constitute most of the time only a pre-processing step towards further image analysis.

In the next chapter, we will introduce waterpixels, a family of methods based on the watershed transformation to efficiently generate superpixels.

5 Waterpixels: A new superpixel generation method based on the watershed transformation

What's in a name? that which we call a rose
By any other name would smell as sweet;
W. Shakespeare, Romeo and Juliet

Résumé

Dans ce chapitre, nous proposons une nouvelle famille de méthodes s'appuyant sur la ligne de partage des eaux morphologique (watershed) pour générer rapidement et efficacement des superpixels. Nous appelons les superpixels ainsi créés les **waterpixels**. Nous évaluons ensuite qualitativement et quantitativement les performances de ces waterpixels à l'aide des critères suivants : adhérence aux contours des objets, régularité de leur support et temps de calcul. Nous poursuivons avec une discussion des résultats et proposons des perspectives d'amélioration.

*****

As announced in the previous chapter, we now propose to use the watershed transformation in order to efficiently generate superpixels. The resulting superpixels will be called *waterpixels*. Actually, this will lead us to construct a family of waterpixel-methods, presented and evaluated in this very chapter.

One must keep in mind that the watershed transformation enables to obtain a low-level-segmentation which, alone, already shows good adherence to object boundaries. Our contribution is therefore to enforce regularity of resulting regions while keeping a reasonable trade-off between the two qualities. Moreover, the proposed family of methods satisfies two additional requirements which are keeping a low complexity and being easily handled by the user. As explained in the next section, there are only two main parameters, which control the size (*step* σ) and the regularity (weight k) of waterpixels.

5.1 Construction of Waterpixels

In this section, we will explain how to generate the partition of a given image f into a set of

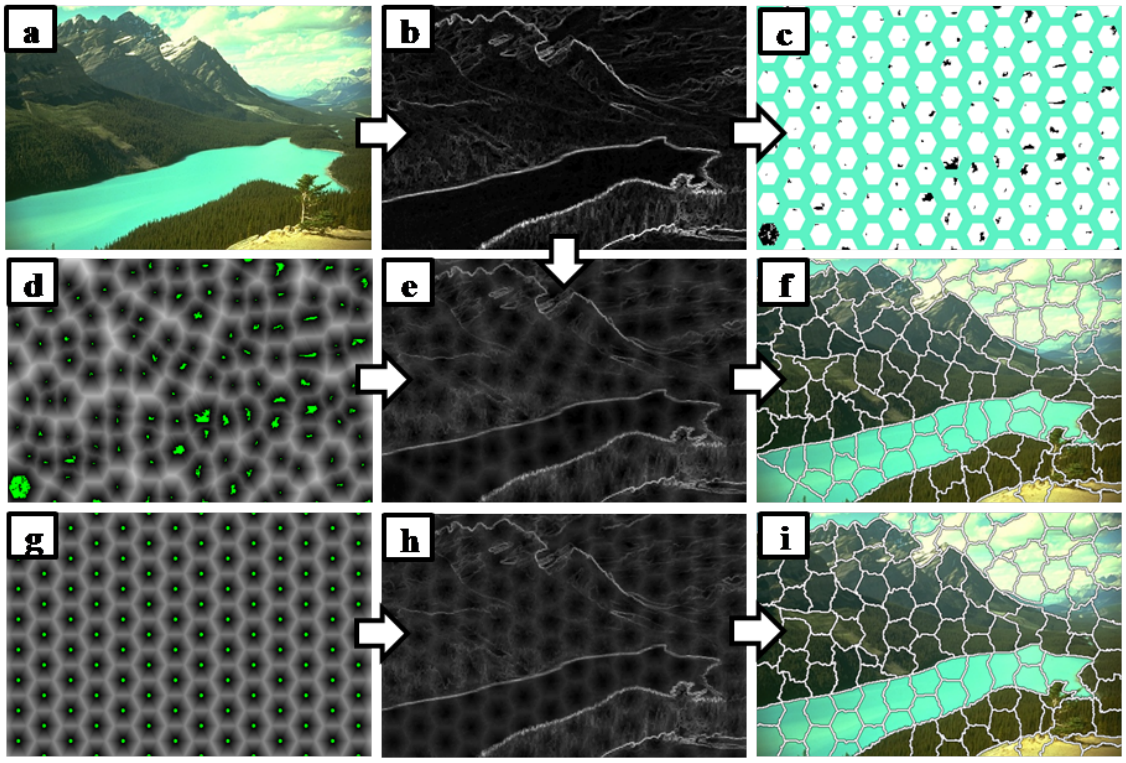


Figure 5.1: **Illustration of waterpixels generation:** (a) original image; (b) corresponding Lab gradient; (c) selected markers within the regular grid of hexagonal cells (step $\sigma = 40$ pixels); (d) distance function to markers; (g) distance function to cell centers; (e) and (h) spatially regularized gradient with distance functions to selected markers (d) and to cell centers (g) respectively; (f) and (i) resulting waterpixels obtained by respectively applying the watershed transformation to (e) and (h), with markers (c).

N waterpixels $\{s_i\}_{1 \leq i \leq N}$. The different steps are illustrated in figure 5.1 and developed in the next paragraphs.

Again, we will denote by $f : D \to V$ the image, where D is a rectangular subset of Z^2, and V a set of values, for example $\{0, \ldots, 255\}$ when f is a 8 bit grey level image, or $\{0, \ldots, 255\}^3$ for color images.

The marker-controlled version of the watershed is chosen here since it offers more control on the resulting regions. This transformation requires two steps: the definition of a gradient (the image to be flooded), and the definition of markers, from which the flooding starts. We would like to enforce regularity through this two steps, as explained in the following.

5.1.1 Selection of the markers

How can we select the markers so as to impose a control on the number and the regularity of resulting waterpixels? Firstly, we know that the number of markers should be equal to the number of desired superpixels in the final partition. Secondly, to obtain regions which are similar in shapes and sizes, one simple idea is to select the N markers so that they are regularly spaced out over the image.

For this purpose, we compute a grid of N regular cells which will serve as a *stake to grow superpixels* throughout the whole procedure. Indeed, these cells serve as model towards which superpixels will tend at the end of the process. We will then propose to select a unique marker per cell of the grid, which will enable to satisfy both requirements, i.e. the number and the regularity of final regions. These two steps, creating the grid and selecting one marker per cell, are investigated in the next paragraphs.

Creating the grid

To define such cells, we first choose a set of N points $\{o_i\}_{1\leq i\leq N}$ in D, called *cell centers*, so that they are placed on the vertices of a regular grid of the size of the image f (a square or hexagonal one for example, leading respectively to square or hexagonal superpixels in the end). Given a distance d on D, we denote by σ the grid step, *i.e.* the distance between closest grid points. A Voronoi tesselation allows to associate to each o_i a Voronoi cell. For each such cell, a homothety centered on o_i with factor ρ ($0 < \rho \leq 1$) leads to the computation of the final cell C_i. This last step allows for the creation of a margin between neighbouring cells, in order to avoid the selection of markers too close from each other.

Selecting one marker per grid cell

How to find the best candidate for each cell, i.e. the marker which will enable to obtain the best performance in terms of boundary adherence and regularity? We could choose a marker independently from the image content, taking for instance the center of the cell. While this simple solution seems appealing at first sight (attempting to enforce regularity), it can sometimes lead to smaller size of SPs and hence a larger variability in size, decreasing the regularity and segmentation performance of the final partition. On the contrary, we advise to select, instead, one of the minima of the gradient in this very cell. Indeed, markers are often chosen among the minima of the gradient as dams of the corresponding watershed are likely to fall on object contours, which is relevant for segmentation purposes.

Therefore, we first compute the gradient image g of the image f. The choice of the gradient operator depends on the image type, *e.g.* for grey level images we might choose a morphological gradient. We then consider the minima of the whole gradient image g. Each minimum is a connected component, composed of one or more pixels. These minima are truncated along the grid, i.e. pixels which fall on the margins between cells are removed.

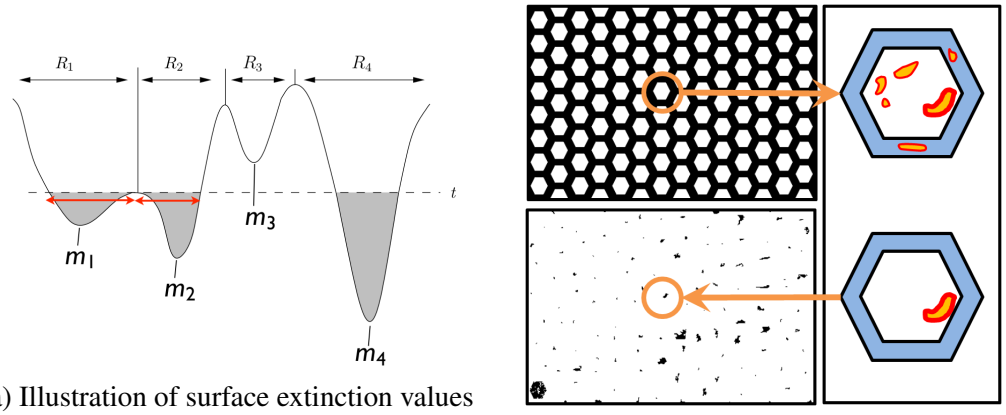
Each cell C_i of the grid defines a region of interest where the content of g is analyzed to select a unique marker. Three cases may happen:

1. There is no minimum of g in C_i.
2. There is a unique minimum of g in C_i.
3. There is more than one minimum of g in C_i.

The choice of the marker in the second case is obvious as only one candidate is available. If more than one minimum is present (third case), we must set up a strategy to select only one among them (see Fig. 5.2). We could choose randomly among the candidates or take the one which the closest to the center. Instead, we prefer to use a quantifiable criterion which will help determine which is the best candidate enabling to provide the best performances. Therefore, we focus on the extinction values introduced by Vachier and Meyer [1995].

Figure 5.2.a shows how to compute the surface extinction value of a given minimum. Let's consider a couple of minima $\{m_1, m_2\}$ for which we would like to find the surface extinction values

val^1_{extS} and val^2_{extS}. Just before the corresponding fusion during flooding, the area of their respective lakes is computed: $\mathscr{A}_1$ for m_1 and $\mathscr{A}_2$ for m_2. In this illustration, $\mathscr{A}_1 > \mathscr{A}_2$. The minimum presenting the smaller lake area, m_2 in our case, will be assigned this very value as surface extinction value: $val^2_{extS} = \mathscr{A}_2$. The fusion process will then keep on, and for each other later fusion regarding m_1, $\mathscr{A}_1$ will be evaluated again and compared to its new competing minimum's area. When the later is higher than $\mathscr{A}_1$, m_1 is assigned the current area value as surface extinction value: $val^1_{extS} = \mathscr{A}_1$. At the end of the fusion process, we obtain the extinction values af all minima. Dynamic and volume extinction values can be explained in the same way, with the evaluation of the dynamic and the volume of the lake instead of the surface. The reader might consult Vachier and Meyer [1995] for further insight into extinction values.



(a) Illustration of surface extinction values

(b) Marker selection procedure

Figure 5.2: Illustration of the marker selection procedure

Thus, if several minima are present, we choose the one with the highest surface extinction value. Indeed, this criterion encourages adherence to object boundaries as well as regularity of resulting regions. We found surface extinction values to give the best performances compared with volume and dynamic extinction values (data not shown).

It may happen that there is no minimum in a cell (first case). In such cases, we must add a marker for the cell which is not a minimum of g, in order to keep regularity. Once again, we will avoid to choose the center of the cell. We propose instead to take, as marker, the flat zone with minimum value of the gradient inside this very cell.

In all cases (i.e. either there exists at least one minimum in the cell or there is not), the selected marker has to be composed of a unique connected component to ensure regularity and connectivity of the resulting superpixel. However, it might not be the case, either if more than one minimum have the same highest extinction value, or if more than one flat zone present the same lowest gradient value in the cell. Therefore, an additional step enables to keep only one of the connected components if there is more than one potential "best" candidate.

The set of resulting markers is denoted $\{M_i\}_{1 \geq i \geq N}$, $M_i \subset D$. The result of the marker selection procedure is shown in Figure 5.1.c.

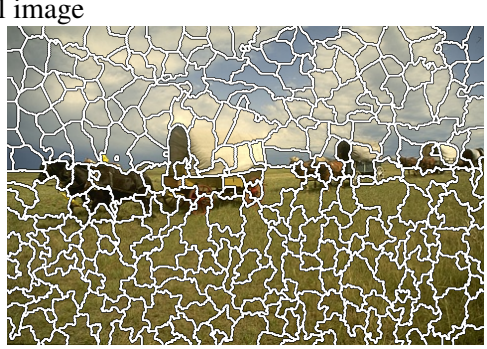
Impact of the spatial repulsion between markers on the segmentation performance

We have efficiently selected our N markers. Before presenting the gradient used for flooding, we would like to study the impact of the markers' distribution on the segmentation performances. We have encouraged the regularity of final regions by spacing out rather regularly the markers over the image, which can be seen as the introduction of a model of *spatial repulsion between markers*. What is the impact of such spatial repulsion on adherence to object boundaries?

To address this issue, we will compare the segmentation performance when the N markers are chosen as in the waterpixels method (selecting one marker per cell of a regular grid; notation: WP) or freely over the image (notation: WS). In both cases, the criterion to find the best candidates will be the extinction values (dynamic, surface and volumic) and the gradient used is image g. We will use hexagonal cells with a margin factor ρ equal to $\frac{2}{3}$.

(a) Original image

(b) Watershed without spatial constraint (WS)

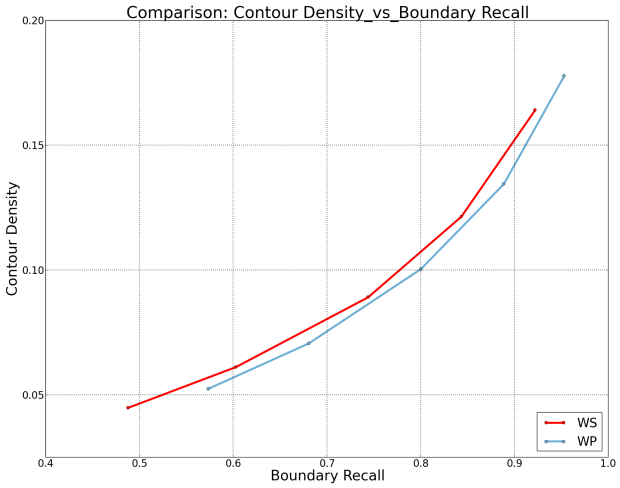


(c) Watershed with spatial constraint (WP)

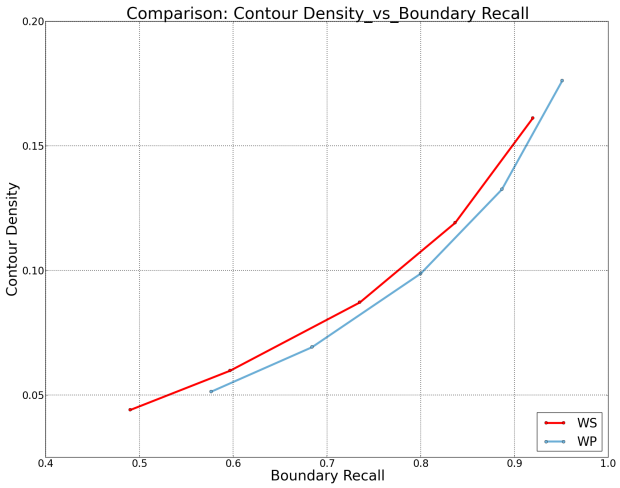
Figure 5.3: Low-level segmentation obtained without and with spatial repulsion.

As an illustration, both methods are applied on an image of the Berkeley segmentation database (see Fig. 5.3). Results for boundary-recall, expressed as a function of contour density, are averaged over the subset "val" of the database and shown in Fig. 5.4 for each extinction value (volume, surface and dynamic). The red curve corresponds to the watershed without repulsion constraint and the blue curve corresponds to watershed with the markers' spatial constraint used for waterpixels, for different values of N (number of markers, which is also the number of regions in the final partition). The ideal case being the highest boundary-recall for the lowest contour density, we can see that waterpixels outperform the usual watershed. We have found that the same behavior can be observed when the grid used for waterpixel generation is composed of square cells instead of hexagonal ones (data not shown).

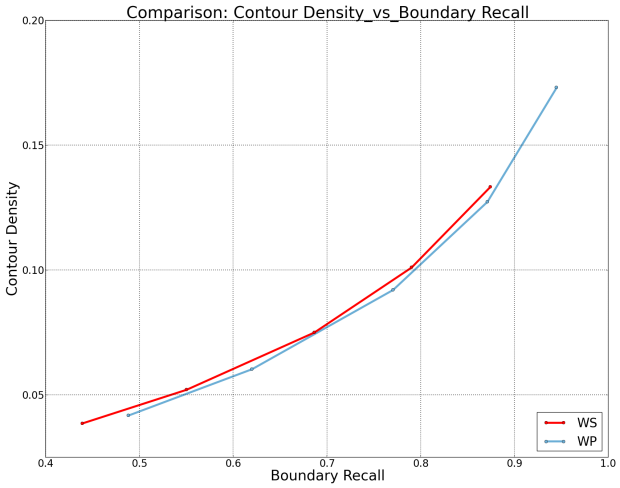
Qualitative results for surface extinction value can be seen in Fig. 5.5 on an image from the



(a) Surface extinction value



(b) Volumic extinction value



(c) Dynamic extinction value

Figure 5.4: Low-level segmentation obtained without and with spatial repulsion.

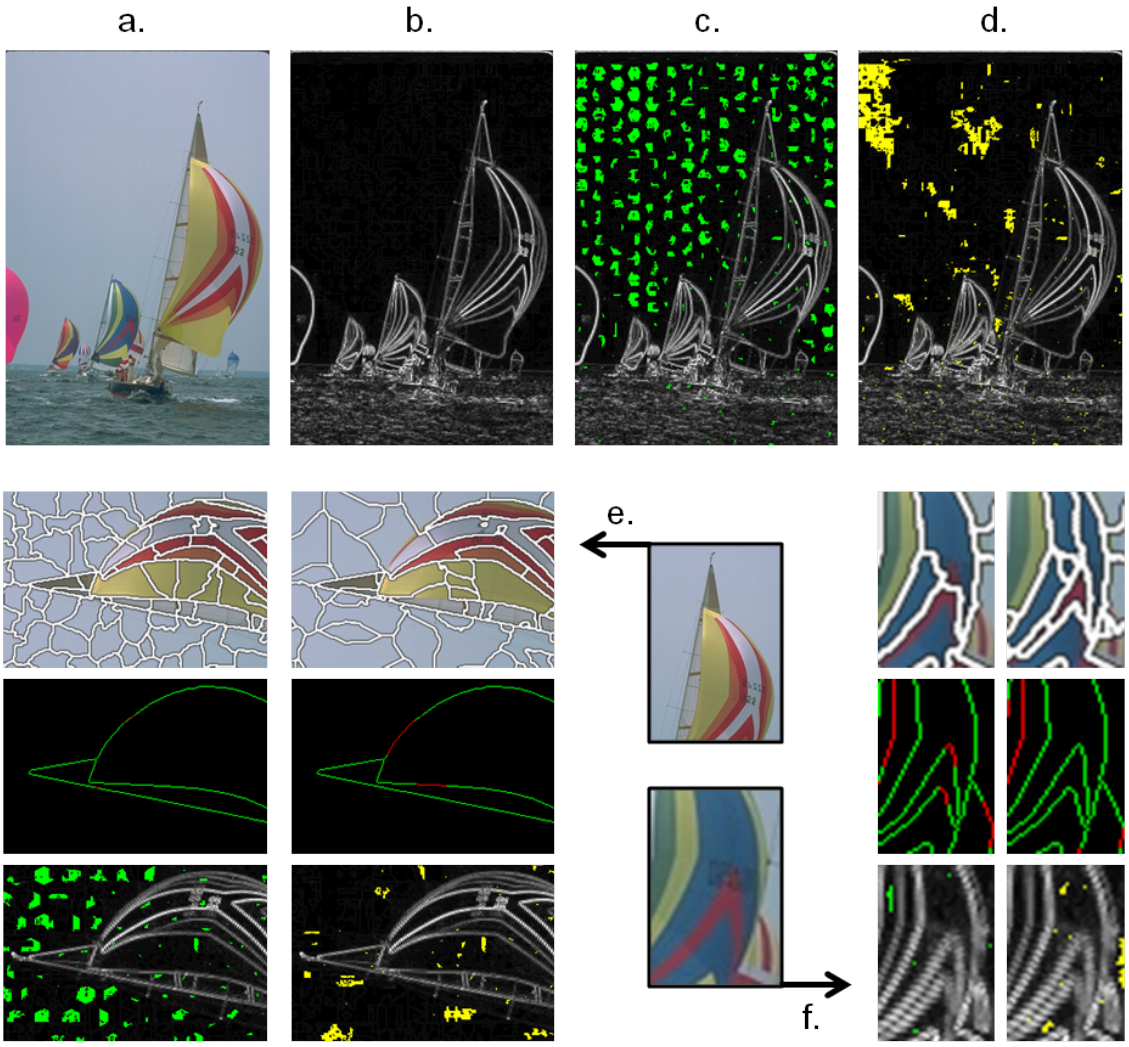


Figure 5.5: **Selected markers for WS and WP and their impact on the adherence to object boundaries:** (a) Original image from the Berkeley segmentation database. (b) Lab-gradient of (a). (c) and (d) selected markers superimposed on the gradient respectively in green for WP and in yellow for WS. **Left example (e)** a region where WP (left column) catches a contour that WS (right column) misses. **Right example (f)** a region where WS (right column) catches a contour that WP (left column) misses. Note that missed contours appear in red whereas reached contours appear in green.

Berkeley segmentation database. In the first row, we see that the spatial distribution of markers for WP is rather regular thanks to the choice of one marker per grid cell, contrary to the markers of WS. It is interesting to note that most of the markers for WS are trapped in the sea because it is a highly textured region. Thanks to the repulsion constraint between markers for WP, these ones are less numerous in the highly textured regions to the profit of other regions such as the sail: in the left example, we can see a contour which has been caught by WP but missed by WS. By imposing a repulsion between markers, we give a chance to weaker contours to be caught, which would not have been caught when we consider only the N best markers without any spatial constraint. The example on the right, on the other hand, enables to understand when WP is outperformed by WS: by imposing a margin between two markers, WP cannot entirely catch small details, i.e. objects smaller than waterpixels size. WS on the other hand can afford to locally increase the number of markers to catch these small objects.

With this study (see Machairas et al. [2016]), we have shown that when we select markers among the minima with a pertinence criterion such as the extinction values, using in addition spatial repulsion for marker selection, improves the recovery of pertinent boundaries. As explained, we believe that it is mainly due to the fact that a high percentage of markers are trapped in textured regions when no constraint on their spacing is imposed, which leads to missing weaker but nonetheless meaningful contours.

It is also interesting to note that, if we randomly choose the marker of each cell for waterpixels (instead of taking the best in terms of surface extinction value), we will still obtain results as good as the"optimal marker choice" according only to extinction values, i.e. WS.

We now close the topic of the impact of spatial repulsion between markers on segmentation performances to address the construction of the gradient used for flooding in the waterpixels method.

5.1.2 Spatial regularization of the gradient and watershed

The selection of markers has enforced the pertinence of future superpixel-boundaries but also the regularity of their pattern (by imposing only one marker per cell). In this paragraph, we design a spatially regularized gradient in order to further compromise between boundary adherence and regularity.

Let $Q = \{q_i\}_{1 \leq i \leq N}$ be a set of N connected components of the image f. For all $p \in D$, we can define a distance function d_Q with respect to Q as follows:

$$\forall p \in D, d_Q(p) = \frac{2}{\sigma} \min_{i \in [1,N]} d(p, q_i) \tag{5.1}$$

where σ is the grid step defined in the previous section. The normalization by σ is introduced to make the regularization independent from the chosen SP size. We have studied two possible choices of the q_i. The first one is to choose them equal to the markers: $q_i = M_i$. Resulting waterpixels are called *m-waterpixels*. The second one consists in setting them at the cell centers: $q_i = o_i$, which leads to *c-waterpixels*. We have found that the first gives the best adherence to object boundaries, while the second produces more regular superpixels. At any rate, varying the set Q of connected components q_i leads us to a family of waterpixels methods.

The spatially regularized gradient g_{reg} is defined as follows:

$$g_{reg} = g + k d_Q \tag{5.2}$$

where g is the gradient of the image f, d_Q is the distance function defined above and k is the spatial regularization parameter, which takes its values within $\Re^+$. The choice of k is application dependent: when k equals zero, no regularization of the gradient is applied; when $k \to \infty$, we approach the Voronoi tessellation of the set $\{q_i\}_{1 \geq i \geq N}$ in the spatial domain.

In the final step, we apply the watershed transformation on the spatially regularized gradient g_{reg}, starting the flooding from the markers $\{M_i\}_{1 \leq i \leq N}$, so that an image partition $\{s_i\}_{1 \leq i \leq N}$ is obtained. The s_i are the resulting waterpixels.

5.2 Recap of the proposed method

We now propose a recap of the proposed family of methods. A waterpixel-generation method is characterized by the following steps:

1. Computation of the gradient of the image;
2. Definition of regular cells on the image, centered on the vertices of a regular grid;
3. Selection of one marker per cell;
4. Spatial regularization of the gradient with the help of a distance function;
5. Application of the watershed transformation on the regularized gradient defined in step 4 from the markers defined in step 2.

Remark 1: This type of methods requires four parameters: three to generate the grid of regular cells (step σ controlling the size, and hence the number of SPs, the shape of the cells and the homothety factor ρ for the margin) and one to spatially regularize the gradient (the weight k). Note that creating the grid is the step which requires the higher number of parameters as it is designed to have a strong impact on waterpixels' properties. In practice, parameters of shape and margin are set to consistent and robust default values (see implementation details in 5.4.1), which only requires to the user to deal with the size σ and the amount of regularization k. Thus, the proposed family of methods not only offers an efficient control on final superpixels but is also easy for the user to handle.

Remark 2: Only the gradient image is needed to compute waterpixels. This gives flexibility to the user, as the latter can produce her/his own gradient image, optimized according to the type of objects to be dealt with, and give it directly as input to the waterpixel method.

An implementation of waterpixels is available from http://cmm.ensmp.fr/∼machairas/waterpixels.

5.3 Comparison with other watershed superpixels methods

In parallel and after the publication of waterpixels, other methods have been proposed to generate regular superpixels with the watershed transformation. In Neubert and Protzel [2014] and Hu et al. [2015], the distance function to the markers is taken into account while performing the flooding of the watershed tranformation. However, the markers are only the centers of the (square) cells, which yields poorer performance. In Benesova and Kottman [2014], on the contrary, the markers are chosen among the minima of the gradient after pertinent filtering of the original image (one per cell, even if the exact selection procedure is not explained into detail), but the gradient is not regularized afterwards.

5.4 Benchmark

We are now going to compare waterpixels to the state-of-the-art method SLIC proposed by Achanta et al. [2012] on the Berkeley segmentation database. This approach offers the best performance among the methods generating regular SPs with low complexity (linear with the number of pixels in the image).

5.4.1 Implementation details

We have found that it is beneficial to pre-process the images from the database using an area opening followed by an area closing, both of size $\sigma^2/16$ (where σ is the chosen step size of the regular grid). This operation efficiently removes details which are clearly smaller than the expected waterpixel area and which should therefore not give rise to a superpixel contour. In practice, filtering only impacts the regularity of final regions.

The Lab-gradient is adopted here in order to best reflect our visual perception of color differences and hence the pertinence of detected objects. The margin parameter ρ, described in 5.1.1, is set to $\frac{2}{3}$.

The cell centers correspond to the vertices of a square or a hexagonal grid of step σ. The grid is computed in one pass over the image, by first analytically calculating the coordinates of the set of pixels belonging to each cell and then assigning to them the label of their corresponding cell. We will display the results for the hexagonal grid, as hexagons are more isotropic than squares. Interestingly, they also lead to a better quantitative performance, which was intuitively expected.

5.4.2 Results

Figure 5.6 shows various images from the Berkeley segmentation database and their corresponding waterpixels (m-waterpixels for distance function to the marker and c-waterpixels for distance function to the cells, hexagonal and square grids, different steps). Figures 5.6.b and 5.6.c (zooms of the original image presented in 5.6.a for m-waterpixels and c-waterpixels respectively) show the influence of the regularization parameter k (0, 4, 8, 16) for a homogeneous (blue sky) and a textured (orange rock) regions. As expected, when $k \to \infty$, m-waterpixels tend towards the Voronoi tessellation of the markers, while c-waterpixels approach the regular grid of hexagonal cells. Both show good adherence to object boundaries, as shown in Figures 5.6.d, 5.6.e, 5.6.f. Of course, enforcing regularity decreases the adherence to object boundaries (see the zoom in Figure 5.6.f for k=16). One advantage of waterpixels is that the user can choose the shape (and size) of the resulting superpixels depending on the application requisites. Figure 5.6.d, for example, presents waterpixels for hexagonal (second and third columns) and square (fourth column) grids.

As a gradient-based approach, the quality of the watershed is dependent on the borders' contrast. If we look at the contours of objects missed by waterpixels, we see that it is due to the weakness of the gradient, as illustrated in Figure 5.8.

In the following, we will use m-waterpixels and denote them directly as "waterpixels" for the sake of simplicity.

During the design of the algorithm, we used intermediate results from the train and test subsets of the Berkeley database. Therefore, we report the results obtained for the validation subset

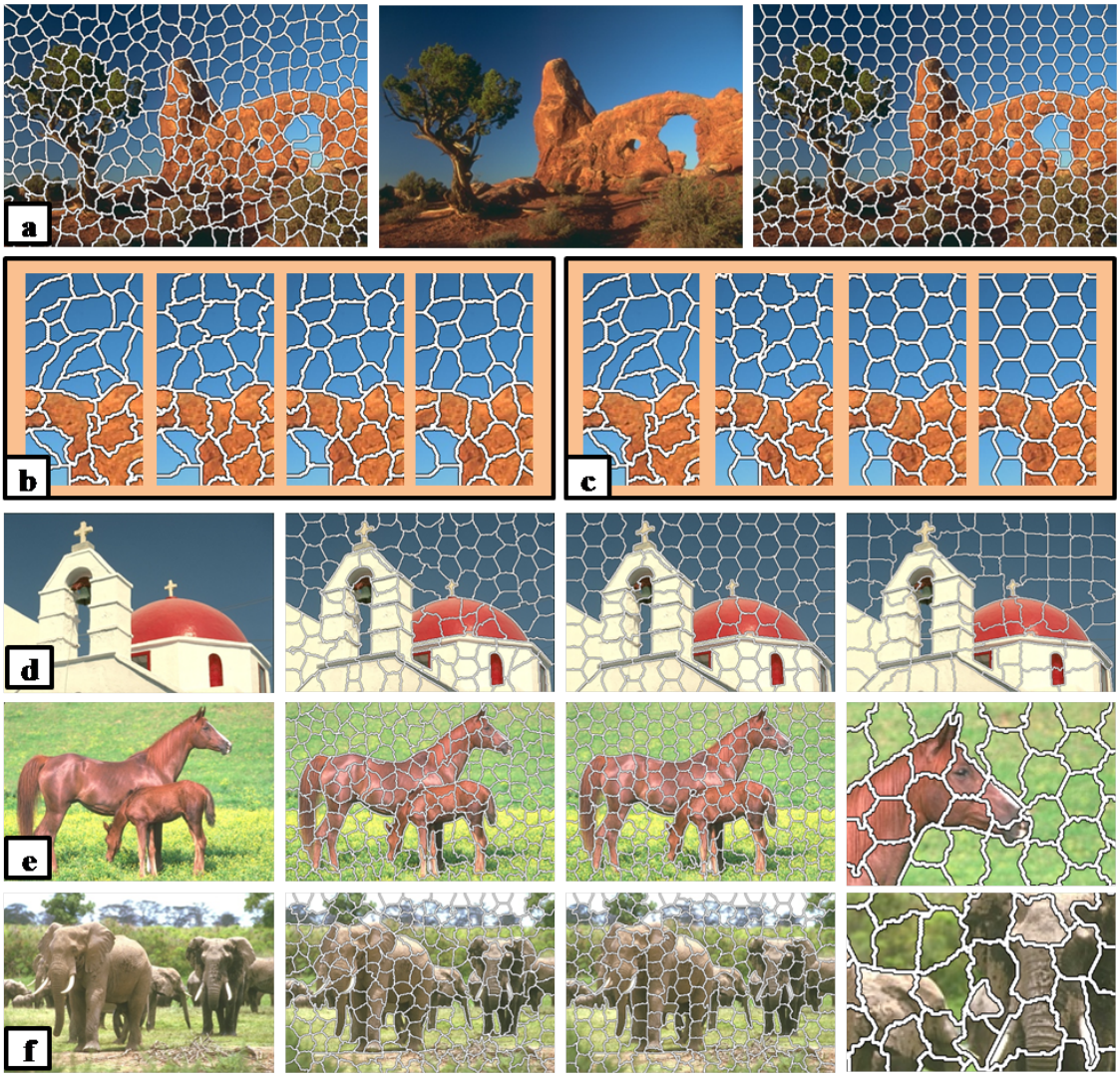
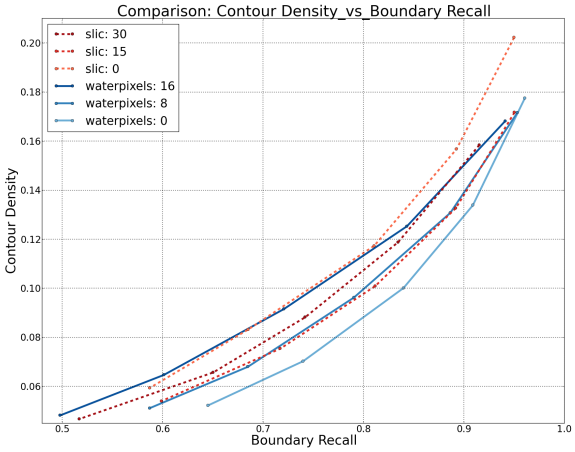
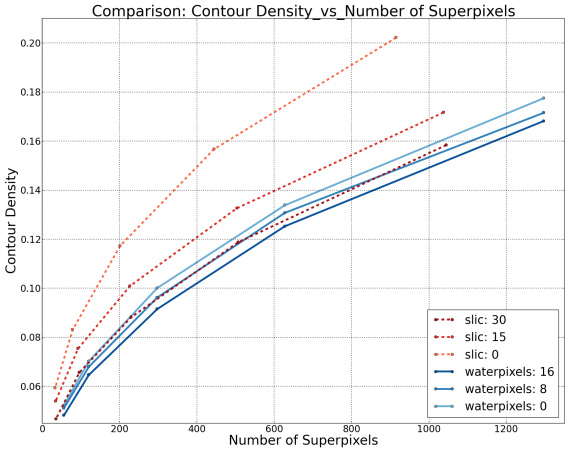


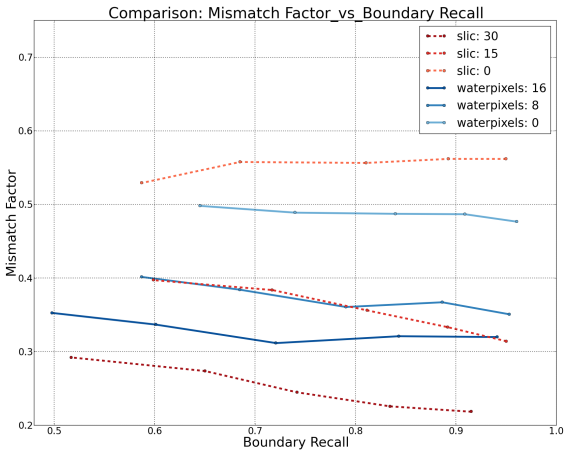
Figure 5.6: **Illustrations of waterpixels on the Berkeley segmentation database:** All waterpixels images are computed with a hexagonal grid with step σ = 30 pixels and a regularization parameter k = 8, unless otherwise specified. (a) original image (middle) with corresponding *m-waterpixels* (left) and *c-waterpixels* (right). $\sigma = 25$ pixels, k=16. (b) zooms of *m-waterpixels* (a) for k = 0,4,8,16. (c) zooms of *c-waterpixels* (a) for k = 0,4,8,16. (d) original image - *m-wat.* - *c-wat.* - *m-wat.* with square grid and $\sigma = 40$ pixels. (e) original image - *m-wat.* - *c-wat.* - zoom of *c-wat.*. (f) original image - *m-wat.* - *c-wat.* - zoom of *m-wat.* with k = 16.



(a) Contour density against boundary-recall.



(b) Contour density against number of superpixels.



(c) Mismatch factor against boundary-recall.

Figure 5.7: Benchmark: performance comparison between waterpixels and SLIC.

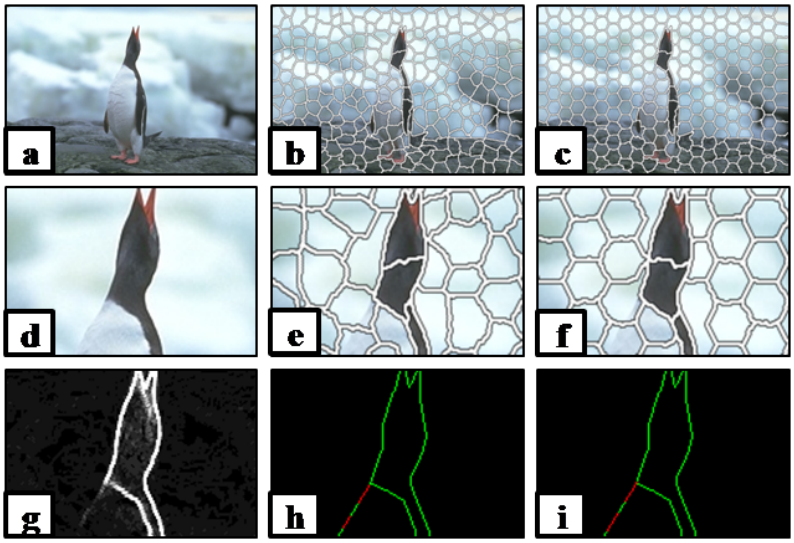


Figure 5.8: **Contours missed by waterpixels:** (a) original image from the Berkeley segmentation database. (b) m-waterpixels with $step = 27$ and $k = 10$. (c) c-waterpixels with $step = 27$ and $k = 10$. (d)–(f) zoom of (a)–(c) respectively. (g) zoom of the non-regularized gradient image. (h) and (i) reached (green) and missed (red) contours, respectively by m-waterpixels and c-waterpixels.

(a) SLIC $k' = 15$ (b) Waterpixels $k = 8$

Figure 5.9: Comparison between Waterpixels and SLIC superpixels for $\sigma = 25$ pixels on a zoom of an image from the Berkeley segmentation database.

("val"), which contains 100 images. Results for boundary-recall, average mismatch factor and contour density are averaged for this subset and shown in Figure 5.7. Blue and red curves correspond to varying regularization parameters k and k' respectively for waterpixels and SLIC. The values for k and k' have been chosen such that they cover a reasonable portion of the regularization space between no regularization ($k = 0$) and a still acceptable level of regularization.

Figure 5.7.a shows contour density against boundary-recall for waterpixels and SLIC. The ideal case being the lowest contour density for the highest boundary-recall, we can see that the trade-off between both properties increases with decreasing regularization, as expected. On the other hand, SLIC exhibits another behavior: the trade-off improves, then gets worse with regularization. At any rate, it is important to note that waterpixels achieves a better "best" trade-off than SLIC (see waterpixel $k = 0$ and SLIC $k' = 15$). Besides, this observation is valid for the whole family of waterpixel-methods as the zero-value regularization does not take into account d_Q. In order to make a fair comparison between waterpixels and SLIC over all criteria, we choose corresponding curves in the trade-off contour density/boundary-recall, *i.e.* waterpixels with $k = 8$ and SLIC with $k' = 15$, and compare this couple for the other criteria.

Figure 5.7.b shows that, for a given number of superpixels, contour density of waterpixels is more stable and most of the time lower than SLIC when varying regularization. More particularly, contour density is lower for waterpixels ($k = 8$) than for SLIC ($k' = 15$). This means that for the same number of superpixels, waterpixels contours are shorter than SLIC contours, which is explained, to a large extent, by less tortuous contours.

Figure 5.7.c shows average mismatch factor against boundary-recall for waterpixels and SLIC. We can see that the curves for waterpixels with $k = 8$ and SLIC with $k' = 15$ are here again close to each other.

These properties are illustrated in Figure 5.9, where we can see examples of reached and missed contours by both methods, as well as their different behaviors in terms of regularity (shape, size, tortuosity).

5.4.3 Computation time

Computing time was measured on a personal computer based on Intel(R) Core(TM) i7 central processing units (4 physical cores, 4 virtual ones), operating at 2.93GHz.

The implementation of the waterpixels was done using the Simple Morphological Image Library (SMIL, Faessel and Bilodeau [2013]). SMIL is a Mathematical Morphology library that aims to be fast, lightweight and portable. It brings most classical morphological operators redesigned in order to take advantage of recent computer features (SIMD, parallel processing, ...) enabling to handle very large images and real time processing.

Both methods have linear complexity with the number of pixels in the image. For an image of size 481×321, average computing time for SLIC was 149 ms, and 132 ms for waterpixels (82 ms without pre-filtering).

A more detailed comparison of computation times is presented in Figure 5.10 (showing average and standard deviation for different numbers of superpixels). We can see that waterpixels are generally faster to compute than SLIC superpixels . Contrary to the latter's, their computation time

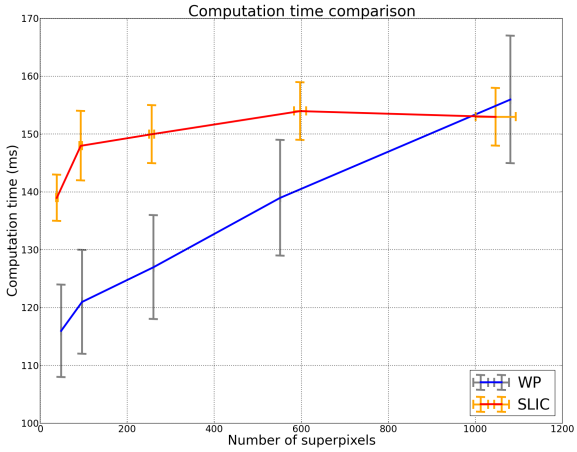


Figure 5.10: Computation time comparison with images of the Berkeley database.

increases slightly with the number of superpixels. An analysis of computation times for the different steps of waterpixels reveals that this variability is only introduced by the grid computation and the minima selection procedure. When it comes to the grid computation time, it rises from 2 ms for small numbers of waterpixels to 27 ms for large numbers of waterpixels. This simply means that we still have to optimize this step. Concerning the computation time of the minima selection procedure, it decreases as waterpixels become larger because of the pre-filtering step. Indeed, the size of this filtering is directly proportional to the cell size. As such, resulting images contain less minima, which simplifies the selection procedure. Besides, the variance observed when we change images is explained by the fact that the difficulty of minima evaluation/computation depends on the content of each image. We are currently working on a new implementation of minima computation/evaluation which would be less dependent on the number of superpixels.

To conclude this section, waterpixels are generally faster to compute than SLIC superpixels, and they are at least as performant in terms of the trade-off between adherence to object boundaries and regularity in shape and size, while using much less pixels to describe their contours.

5.5 Conclusion, discussion and prospects

This chapter introduces waterpixels, a family of methods for computing regular superpixels based on the watershed transformation. Both adherence to object boundaries and regularity of resulting regions are encouraged thanks to the choice of the markers and the gradient to be flooded. Different design options, such as the distance function used to spatially regularize the gradient, lead to different trade-offs between both properties. The computational complexity of waterpixels is linear. Our current implementation makes it one of the fastest superpixel methods. Experimental results show that waterpixels are competitive with respect to the state-of the art, and even tend to outperform one of the best and most widely used methods for superpixel generation, albeit with only a small margin. The trade-off between speed and segmentation quality achieved by waterpixels, as well as their ability to generate hierarchical segmentations at negligible extra cost, offer interesting perspectives for this superpixel generation method.

We have shown that waterpixels produce competitive results with respect to the state-of-the-art. These advantages are valuable in the classification/detection/segmentation pipeline, where

superpixels play the part of primitives. Moreover, there is one major difference in the construction of the algorithm: the SLIC approach does not impose any connectivity constraint. The resulting superpixels are therefore not necessarily connected, which requires an *ad hoc* postprocessing step. In contrast, waterpixels are connected by definition thanks to the watershed.

The proposed approach is gradient-based. Standard methods can be used to compute this gradient, or a specific gradient computation method can be designed for a given application. In any case, this offers flexibility to waterpixels. One limitation, though, is the quality of the signal in such a gradient image. As seen in 5.8, alteration by noise or insufficiently contrasted contours may lead to the prevalence of regularity over adherence to object boundaries. If filtering steps are usually enough to deal with noise and remove irrelevant small details, parameter values have to be optimized for each database. Future work will aim at overcoming this limitation by adding a learning step of optimal filtering values for specific databases.

The general design of waterpixels offers many prospects. Among them, one promising field of improvement resides in the placement of markers, as they constitute the main degree of freedom of the method. Future work could investigate the possibility to select the markers in an optimal manner, for example by formulating the marker placement as a p-dispersion problem (see Erkut [1990]) in an augmented space.

Last but not least, waterpixels lead to the efficient construction of hierarchical partitions based on superpixels. Indeed, the computation of the watershed can produce at the same time a segmentation and a hierarchy of partitions based on that segmentation, with only minor overhead computation times (see Meyer [1994]; Beucher [1994]; Meyer [2001]).

We are now going to use waterpixels as pertinent supports in the classification pipeline to learn segmentation.

II Learning Segmentation with Waterpixels

*****

Our aim is to provide a general segmentation method enabling to offer good performance on any database with the only help of supervised learning.

In Part 1, we have decided to address this issue with pixel classification (UC: pixel). We have noticed that a wider computational support (CS), such as the window, could capture a richer information than the pixel itself. However, performance could be further improved (especially on object borders) with a support adapting to the image content.

Therefore, in Part 2, we have focused on another image representation: superpixels. Indeed, contrary to windows, superpixels seem to constitute a more pertinent support as they adapt well to the image content (these regions result from a special case of low-level segmentation). We have proposed a novel superpixel generation method, *waterpixels*, which is efficient, fast and easy to compute for the user. These advantages make them a good candidate to be used for this generation pre-processing step.

Part 3 is hence dedicated to the ***insertion of waterpixels in the classification pipeline*** and the ***study of their pertinence*** in such a workflow, as illustrated in Fig. 5.11. In this example, we can see that computing a feature (for instance the mean intensity in the support) on a superpixel rather than on a window will lead to an easier and hence better classification result on borders. We will now present two ways of using superpixel-CS in this workflow.

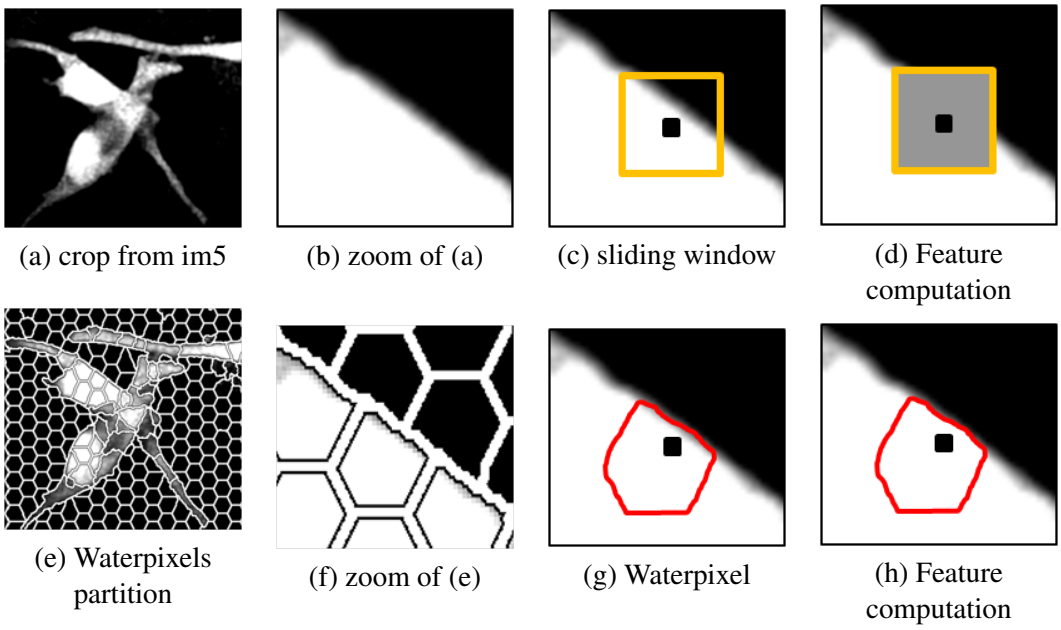


Figure 5.11: Comparison between *CS* = window and *CS* = superpixel.

*****

*****

Related Publications and Conferences

- *New General Features Based on Superpixels for Image Segmentation Learning*
  V. Machairas, T. Baldeweck, T. Walter and E. Decencière
  International Symposium on Biomedical Imaging (ISBI), 2016

- *SAF: A New Superpixel-Based Feature for Pixel Classification*
  V. Machairas, T. Baldeweck, T. Walter and E. Decencière
  39th session of ISS France, 2016

*****

6 How superpixels are used in the literature

I learned that courage was not the absence of fear, but the triumph over it.
The brave man is not he who does not feel afraid, but he who conquers that fear.
Nelson Mandela

Résumé

Ce chapitre présente un état de l'art de l'utilisation des superpixels pour la segmentation/détection d'objets dans les images. Nous nous intéressons en particulier à la classification de superpixels, alternative qui semble pertinente pour améliorer la chaîne de classification de pixels. Nous montrons que les waterpixels sont des superpixels qui offrent de bonnes performances de sur-segmentation et qu'ils constituent des supports plus adaptés pour le calcul des descripteurs permettant de les classer.

*****

In this chapter, we explain how superpixels are used in the literature. We then focus on one particular application, superpixel classification, which is a good candidate to replace pixel classification. After expounding the process, we quantitatively evaluate the segmentation performance of this approach, benchmarked against the pixel and sliding window ones.

6.1 Examples of use in the literature

Superpixels have raised huge interest over the past years, as they provide a compact representation of an image and can be efficiently computed. They can be found in many applications, for instance in text detection (Zhou et al. [2012]) or street scene segmentation (Micusik and Kosecka [2009]). They are of course widely used in the biological and medical fields where the data desperately needs such a simplification (high resolution and 3D lead to big data). We can cite tumor segmentation (brain in Wu et al. [2014], liver in Conze et al. [2015] and Conze et al. [2016]), optic cup localization (Xu et al. [2012], Tan et al. [2015]), and segmentation of EM images (Lucchi et al. [2012], Andres et al. [2008], Andres et al. [2012]).

Before analyzing the different families of methods using superpixels in the literature, we will start by presenting four examples of articles with different approaches of their use.

In Xu et al. [2012], the aim is to segment the optic cup in eye fundus images in order to compute the cup-to-disk ratio, a useful clinical indicator of glaucoma (a pathology causing blindness). The proposed method consists in partitioning each image into superpixels, removing those corresponding to the blood vessels and the outside area of the optic disk, and classifying those which remain into "cup" and "rim" regions. They use normalized features to describe superpixels, based on spatial information, mean RGB colors and grey-level histograms. A SVM classifier with linear kernel performs the learning. They eventually provide a superpixel label refinement scheme integrating prior knowledge on such biological structures (taking into account the distance to the center of the optical disk as well as the labels of the neighboring superpixels), which enables them to outperform state-of-the-art methods based on pixel classification and sliding window approaches applied on the same task.

In Andres et al. [2008], the aim is to segment neurons present in 3D images acquired by serial block-face scanning electron microscopy (SBFSEM). The proposed method consists in partitioning each image into superpixels, and classifying each face (*i.e.* each frontier between two adjacent superpixels) as "real object contour" (1) or "false object contour" (0). They use intensity distribution characteristics as well as the difference of mean intensities between the two corresponding superpixels to compute the vector of features of each face, and a random forest classifier to perform the learning. It is also interesting to note that the first step, SP partitioning, also contains a learning approach, consisting in two phases. First, a voxel classification procedure is applied on the images to determine if each voxel belongs or not to an object contour. Second, the resulting probability map, considered as a topographical surface, is flooded starting by a reduced number of minima (marker-controlled watershed) to obtain a partition of non-regular but nonetheless homogeneous regions used as SPs. The authors show that they achieve the same accuracy as the state-of-the-art method based on a convolutional neural network (CNN) designed for the same task, and with a lower cost (training a CNN being quite expensive).

The approach of Levinshtein et al. [2010] also considers the faces between adjacent superpixels. However, contrary to the previous paper, they are not dealing with classification but with fusion of such units. Indeed, they focus on *contour closure*, a general problem consisting in finding the best cycle of contour fragments that enables to separate a figure from a background. To reduce the computational complexity of state-of-the-art methods which perform an exhaustive search over all grouping possibilities, they propose to partition each image into superpixels and then find the "best" subset of superpixels, defined as the one whose overall boundary is the most strongly supported by image edges. Thus, they only consider the pixels lying on SP contours. The edginess of such an SP contour pixel, *i.e.* the probability that it belongs to an object contour, is learned, its vector of features containing information on the local geometry of the SP boundary at this point and on the detected image edge evidence in its neighborhood. Then, the best closure is detected by minimizing, with parametric maxflow, the amount of points with weak edginess along the boundary of an SP grouping. The result is then a cycle of SP faces which enables to perform the best separation between the figure and its background.

The last example, Chai et al. [2011], addresses the pre-processing step of *background removal*, useful, later on, to ease and improve multi-class classification of objects found in the foreground. For this unsupervised first phase, a GrabCut approach is used to segment each image of the database into "foreground" and "background" parts. The idea of GrabCut is to estimate the Gaussian mixture of each class in order to infer how to cut the graph of corresponding pixels. However, when computing this estimation, they propose to weight the importance of each pixel

by taking into account information provided by superpixels. Indeed, the GrabCut method works well on a given image, but is less efficient on the whole dataset as classes' Gaussian mixtures are more likely to overlap so are less easy to discriminate. Therefore, the idea consists in adding a supervised step, *i.e.* the classification of superpixels (of the whole dataset this time) into the same two classes, "foreground" and "background", performed with a linear SVM. Each superpixel is then assigned the distance to the learned separating hyperplane in the feature space. This value is then used to weight every pixel of a given superpixel, when computing the Gaussian mixture estimation of a class. We can thus notice that superpixels sometimes do not constitute the main workflow but can serve as a help to improve the performance of another segmentation method.

Through these four examples, we have seen that superpixels can be used in many different ways, specific or not to a given type of objects and images. We now propose to give different categorizations of the literature, highlighting each time a different aspect such as the use or not of classification, the nature of the classification unit, the importance of SPs in the pipeline etc.

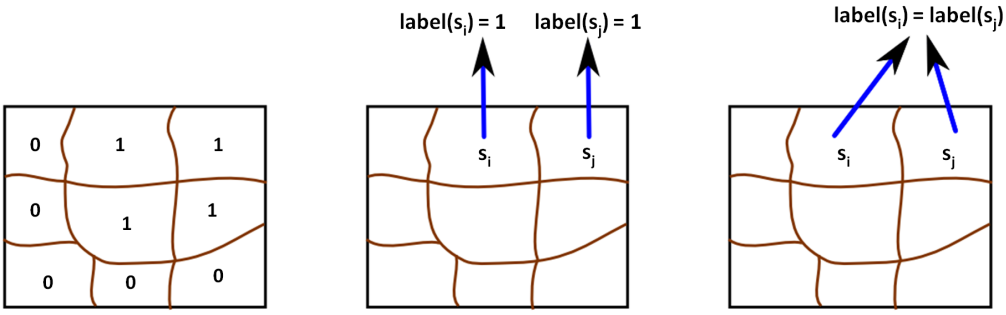
(a) SP as a region.

(b) A face between two SPs.

Figure 6.1: Region and face : two classification units for superpixels.

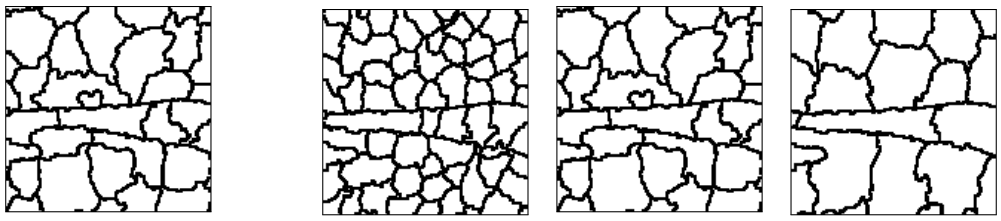
Region / Face: A superpixel can be seen either as a region (as in Xu et al. [2012]) or a set of faces (as in Manfredi et al. [2014]), each face corresponding to a frontier between itself and one of its neighbors. We can then choose to deal with a region or a face for the *unit of classification* "superpixel" (see Fig. 6.1.a. vs. Fig. 6.1.b.). This choice determines the meaning of the labels when performing classification, for instance. In our case, working on regions lead to labels "object" (1) and "background" (0), whereas working on faces imposes labels "contour" (1) and "not a contour" (0). It also guides the design of associated features: a region will be described by its inner color and texture, whereas a face is defined mainly by the difference of such properties between its corresponding pair of superpixels, as well as by its intensity distribution along itself. In this work, we choose the region approach for two reasons. First because its results can easily be transferred to the second approach. Second because the result of face classification may not always lead to a proper segmentation and would hence require an additional pruning step (which has a cost and is also an additional potential source of errors). We will then deal with superpixels as regions in the following.

Classification / Fusion: We have expounded how pixel classification works in Part I. **Superpixel classification** is similar, however superpixels, instead of pixels, are used as classification units, *i.e.* the aim is to assign to each superpixel the label of the object it belongs to. All pixels of a given superpixel will thus obtain the same label. This approach enables to reduce the computational cost by reducing the number of classification units, as well as to integrate a more pertinent information on the objects by using a computational support which is wider than a pixel and which adapts to the image content (contrary to a fixed window). Classification of superpixels (either regions or their faces) is hence pretty often used in the literature, as in Andres et al. [2012], Manfredi et al. [2014] or Xu et al. [2012] for instance. Another way to start with this low-level segmentation



(a) SPs and corresponding GT values (b) Classification (c) Fusion

Figure 6.2: Classification vs. fusion (of SPs considered as regions).



(a) Single scale (b) Multiple scales

Figure 6.3: Single-scale vs. multi-scale.

into superpixels and end with a final segmentation is to perform a **fusion** process on these very units. It can be achieved locally, gathering two neighboring regions if they satisfy or optimize a given criterion (see Fig. 6.2.c.). The result is a hierarchy of partitions whose finest partition is the superpixels and the coarsest is the final segmentation or the whole image according to the criterion used. In the later case, the best partition can be found by searching the "best" cut in the hierachy according to a learned cost for instance. Or the aim can be to study each possible grouping of superpixels to say whether this gathering is optimal or not (either being a group of regions, or a cycle of fragments). In our work, we will focus on superpixel classification so that it can be compared to the pixel and window approaches.

One partition / multiple partitions: Classification and fusion, as previously explained, only deal with a unique partition of superpixels. However, as pointed out in the work of Malisiewicz et al. among others, it is incautious to rely on a unique low-level segmentation. Firstly because it has its own part of mistakes (some missed contours). Secondly because the information which is captured is conditioned by the method and parameters used to obtain such a partition. Let us consider for instance the impact of the parameter controlling the size of superpixels. We can easily see that objects which are smaller than the superpixel size will not, or not entirely, be detected. Hence the size of the regions has an impact on the segmentation performance. But how can we choose the best size to compute the unique partition? A unique answer may not even be possible to give, as there may be objects with different sizes in the images, hence corresponding to different superpixel sizes. Thus, the idea is to produce different partitions, one for each size, and "combine" them to obtain the final segmentation taking into account multi-scale information. But size is only one of the parameters. This procedure could also be enriched by partitions coming from the variation of other parameters'values, or even being generated by different SP generation methods, on different images of the same scene (another wavelength, another acquisition time, ...). The advantage of combining multiple partitions, such as in Conze et al. [2016] and Geremia et al. [2013], is to

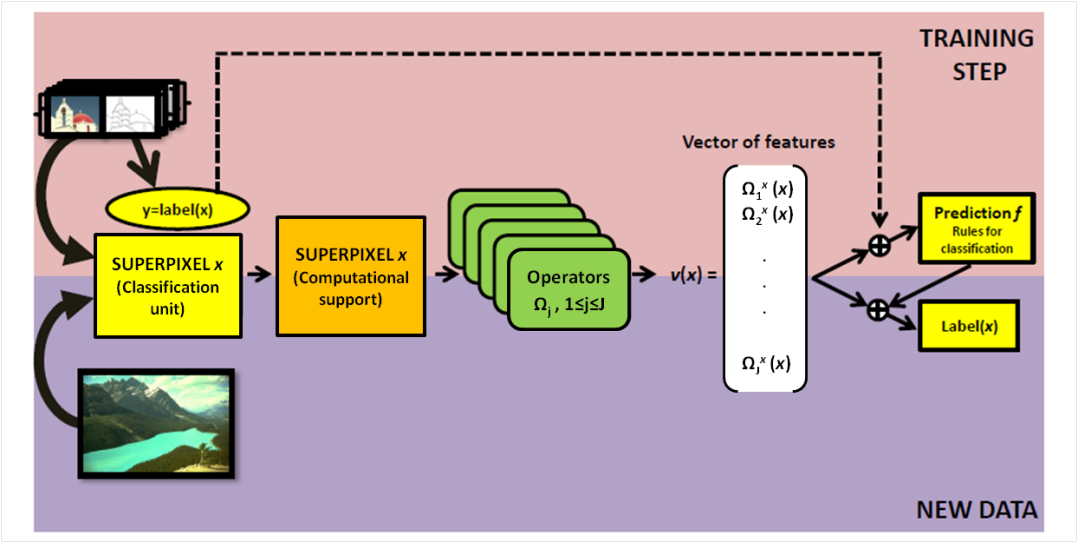


Figure 6.4: General scheme for superpixel classification.

capture more information on the image content, leading to an improvement of the segmentation performance.

Principal element / support element: As we have seen in the four examples, superpixels may be at the core of the segmentation pipeline, or just serve as a help to improve other segmentation/detection methods. In both cases, they are used as pertinent supports because they adapt to the image content.

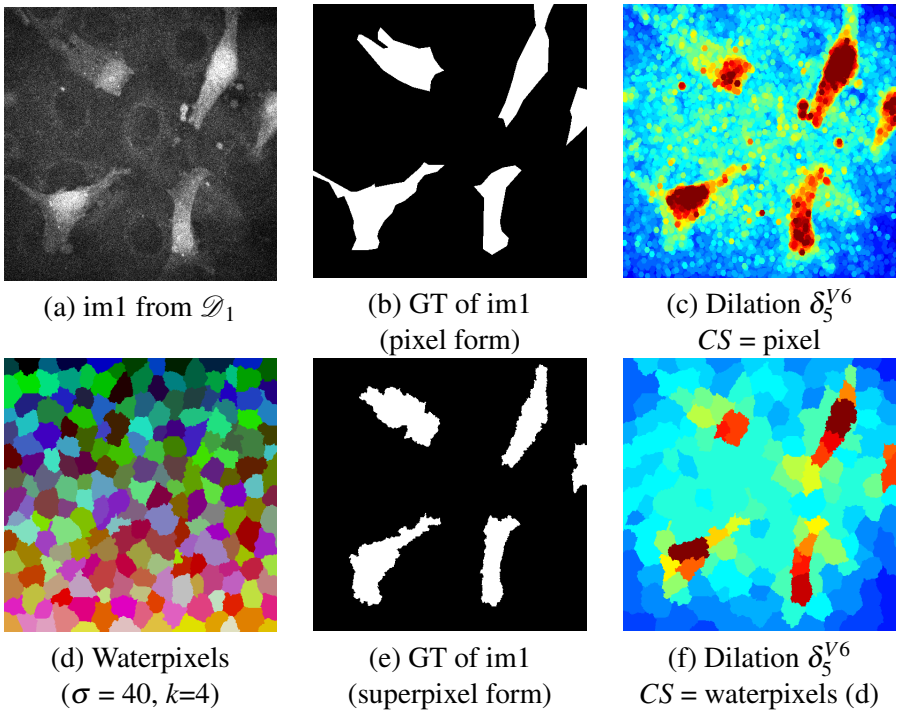
To replace pixel classification (either with pixel or window supports), we are now going to investigate one solution proposed in this literature: superpixel classification (based on one partition of SPs). This second pipeline will also be assessed on the three databases $\mathscr{D}_1$, $\mathscr{D}_2$ and $\mathscr{D}_3$ to see the impact on segmentation performance.

6.2 Superpixel classification

This section is dedicated to the investigation of superpixel classification, and its benchmark against pixel and window approaches.

6.2.1 Principle

As explained in the previous section, superpixels are now used as classification units, instead of pixels. They are also their own (and only) computational support to compute features, as shown in Fig. 6.4. Features $\{F_j\}_j$ are defined the same way as in Chapt. 2: $F_j = \{c, \Omega, GEO, CS, \Sigma\}$, where c is the channel, Ω is the operator, GEO is a Boolean for geodesic or non-geodesic way of applying Ω, CS is the computational support (here, the superpixel) and Σ is the integrator of the CS. It is important to note that the integration over the CS (resulting in a unique value) is now compulsory to ensure that all vectors of features have the same length (superpixels do not have exactly the same number of pixels even when regularity is enforced) and can thus be processed by the machine learning method. As far as groundtruth is concerned, for training, each SP is assigned the label of the majority class among its pixels. Note that evaluation is performed afterwards with the pixel form, as previously explained in Chapt. 3. Figure 6.5 illustrates the difference between pixel and superpixel features (and groundtruth).



(a) im1 from $\mathscr{D}_1$ (b) GT of im1 (pixel form) (c) Dilation δ_5^{V6} CS = pixel

(d) Waterpixels (σ = 40, k=4) (e) GT of im1 (superpixel form) (f) Dilation δ_5^{V6} CS = waterpixels (d)

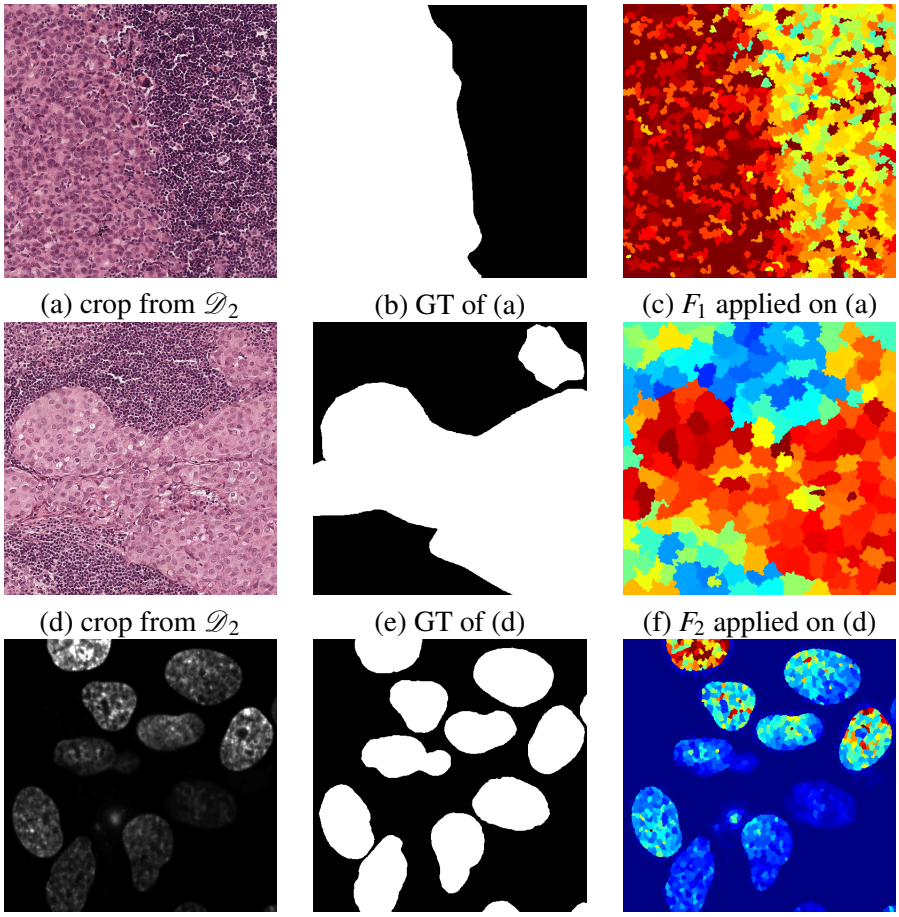
Figure 6.5: Superpixel features vs. pixel features.

In the following, waterpixels will be computed to obtain superpixels, but any other efficient and fast SP-generation method could have been used. Figure 6.6 shows more examples of features computed on the two other databases $\mathscr{D}_2$ and $\mathscr{D}_3$, taking each time the mean value over the waterpixels. They are to be compared with the groundtruth (shown in the second column in their pixel partition form). This set of features have been chosen on purpose in this illustration as they are rather discriminant, thus likely to be used by random forests to perform a good classification of superpixels.

As far as the parameter GEO is concerned, it is set to false (*i.e.* non-geodesic way) when the operator Ω is first applied to the image before integrating over the CS. On the contrary, the geodesic way consists in applying the operator directly on the CS, and then integrating information of this very CS into a unique value. The second way enables to prevent border influence, as illustrated in Fig. 6.7 where two operators (two dilations: δ_5^{V6} of size 5 and δ_{10}^{V6} of size 10, both with V6 neighborhood) are applied in both geodesic and non-geodesic ways with the same waterpixel partition. The geodesic way seems to better help discrimination between objects and background, but there may also be cases where the non-geodesic way is preferable in order to incorporate contextual information.

6.2.2 Preliminary study on $\mathscr{D}_1$

As in Chapt. 3, a preliminary study is conducted on the L'Oréal database $\mathscr{D}_1$. We remind that the random forest parameters have been set to consistent values (100 trees, 100 data at least in each leaf at the end of training) and that a leave-one-out procedure is applied to evaluate the segmentation performance. General results, including results on the other two databases, will only be shown in the next chapter for the new proposed method (see Chapt. 7).



(a) crop from $\mathscr{D}_2$ (b) GT of (a) (c) F_1 applied on (a)

(d) crop from $\mathscr{D}_2$ (e) GT of (d) (f) F_2 applied on (d)

(g) crop from $\mathscr{D}_3$ (h) GT of (g) (i) F_3 applied on (g)

Figure 6.6: **Examples of features computed on the three databases.** Legend: $F_1 = \{0, Haralick'SumAverage', False, waterpixels(\sigma = 20, k = 4), meanvalue\}$, $F_2 = \{0, Haralick'SumAverage', False, waterpixels(\sigma = 40, k = 50), meanvalue\}$, $F_3 = \{0, Identity, False, waterpixels(\sigma = 10, k = 4), meanvalue\}$

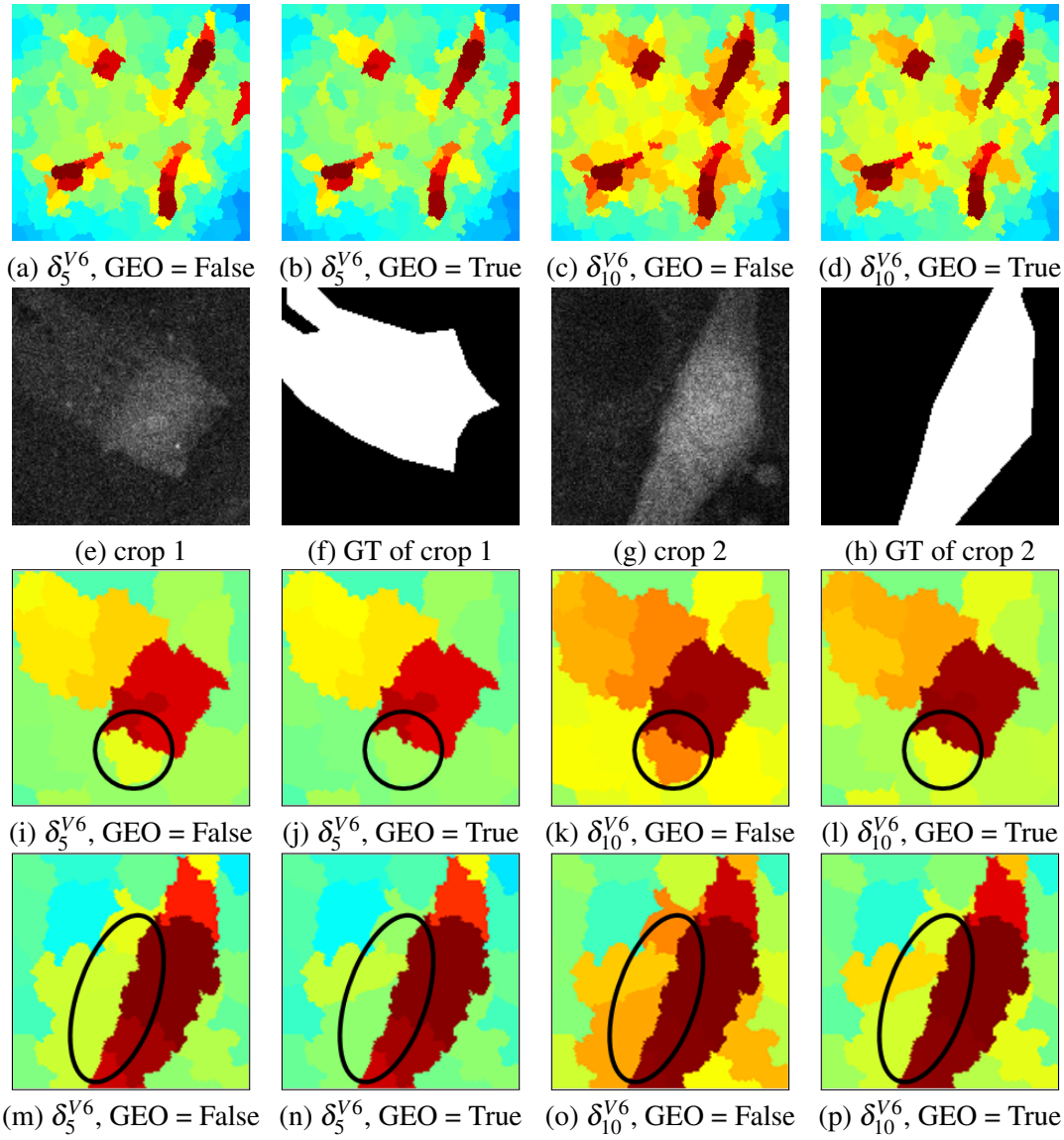
Waterpixels on $\mathscr{D}_1$

In this paragraph, we show that waterpixels achieve an efficient low-level segmentation of $\mathscr{D}_1$ images, and are hence suitable for classification purposes.

As explained in the previous paragraph, this approach enables to perform a single-scale classification since superpixels used as UC are generated with a given size (*i.e. step* σ for waterpixels). We will study the performance for 4 different sizes (the same used previously for windows as shown in Tab. 6.1) leading each time to a different classification result.

radius (windows)	σ (waterpixels)
5	10
10	20
15	30
20	40

Table 6.1: Corresponding sizes (in pixels) for windows and waterpixels support.



(a) δ_5^{V6}, GEO = False (b) δ_5^{V6}, GEO = True (c) δ_{10}^{V6}, GEO = False (d) δ_{10}^{V6}, GEO = True

(e) crop 1 (f) GT of crop 1 (g) crop 2 (h) GT of crop 2

(i) δ_5^{V6}, GEO = False (j) δ_5^{V6}, GEO = True (k) δ_{10}^{V6}, GEO = False (l) δ_{10}^{V6}, GEO = True

(m) δ_5^{V6}, GEO = False (n) δ_5^{V6}, GEO = True (o) δ_{10}^{V6}, GEO = False (p) δ_{10}^{V6}, GEO = True

Figure 6.7: Geodesic and non-geodesic ways of applying operators.

Figure 6.8 shows how waterpixels behave on the $\mathscr{D}_1$ database: four crops (columns) are partitioned into superpixels of four different sizes (rows). These low-level segmentations succeed in catching most object contours, especially in the third and fourth cases (third and fourth columns). Sometimes, a given scale is preferable, as for crop 1 where the object can be better distinguished with the larger scale (σ = 40), or for crop 2 where a lower size (e.g. σ = 20) enables to detect thinner parts of melanocytes.

To go further on waterpixels evaluation, we compute the performance of the best segmentation achievable with waterpixels of each size. To do so, we simulate a perfect superpixel classification by assigning to each SP the label of the majority class among its pixels (as done to compute the groundtruth in the training phase). This will give us the best reachable performance with this approach. Figures 6.9 and 6.10 show the results for the eight images of database $\mathscr{D}_1$. Qualitatively, we can observe that objects are nearly entirely recovered for all sizes, which confirms that waterpixels perform well also on this database. The study of each size emphasizes once again the fact that superpixels detect objects which are equal or bigger than their own size. This phenomenon is highlighted with the case of im3 for instance, where we can see that objects which are big enough are kept for all sizes while thin elongated parts tend to be split or assimilated to the background when σ increases.

Table 6.2 gives the corresponding quantitative results, computing the precision, recall, F-score, Jaccard index and accuracy. As observed before, performance increases when σ decreases: recall is higher since thinner structures can be caught by SP contours, increasing from 76% to 91%. Precision stays rather stable when σ varies (P_{σ_10}=93, $P_{\sigma_{20},\sigma_{30},\sigma_{40}}$=91), which means that errors due to SP overlapping remain low and are approximately constant. These mistakes are then more due to the quality of image content more than to an effect of SP size. On the whole (apart from im6 which appears more difficult to segment due to the presence of objects with many different sizes), the images present rather similar and good results, which means that waterpixels are rather robust over the database. This can be clearly noted when considering the largest σ.

Study with different families of features

We are now going to study the segmentation performance obtained when classifying superpixels of $\mathscr{D}_1$ with two sets of features: features based on operators from Mathematical Morphology (notation: MOMA) and Haralick. Let us start by operators from Mathematical Morphology.

Figure 6.11 shows classification results for im1, im5 and im3 of $\mathscr{D}_1$, when using as UC **and** CS waterpixels of size σ=10, 20, 30 and 40 respectively. The integrator Σ is the mean over the waterpixel. Colors mark pixels as follows (as previously done in Chapt. 3): green for true positives, white for true negatives, red for false positives and blue for false negatives. The case of im1 illustrates well the effect of size on segmentation performance. We can see that when σ increases, the number of false positives decreases. Indeed, small artifacts, which can induce errors when dealing with the region they are contained in, have less impact when the region is larger. For the same reason, small parts of real objects have not enough importance in larger regions to be assigned the good label. Hence, the number of false negatives also increases with σ. This behavior can also be observed on the case of im5 (second row). False positives are less spread over the image, but more and more gathered. Image 3 constitutes a more difficult case because of the distribution of illumination in the original image (see Fig. 3.2.c in Chapt. 3). Thus, more false positives can be observed in the center of the image.

Quantitative results taking into account the eight images of the database are shown in Tab. 6.3.

Measure	σ	im1	im2	im3	im4	im5	im6	im7	im8	all images
Precision	10	93	93	93	93	95	90	92	93	93
	20	91	93	90	91	95	86	91	93	91
	30	89	92	92	92	93	89	91	91	91
	40	91	91	89	91	95	87	87	90	91
Recall	10	94	89	90	91	91	90	92	92	91
	20	89	84	87	88	86	81	86	87	86
	30	84	79	83	80	86	68	84	80	81
	40	80	73	73	76	78	62	80	83	76
F-score	10	93	91	91	92	93	90	92	92	92
	20	90	88	88	89	90	83	88	90	88
	30	86	85	87	86	89	77	87	85	86
	40	85	81	80	83	86	72	83	86	83
Jaccard index	10	88	84	84	85	86	82	85	86	85
	20	83	79	79	81	82	72	79	81	80
	30	76	74	78	75	80	62	77	74	75
	40	74	68	68	70	75	54	72	76	70
Accuracy	10	98	96	98	97	98	98	97	98	98
	20	97	95	98	97	97	96	96	97	97
	30	96	94	98	96	96	95	96	96	96
	40	96	93	96	95	95	94	95	96	95

Table 6.2: Ideal classification of waterpixels: performance on $\mathscr{D}_1$.

For Σ = mean, precision tends to increase with σ, while recall decreases. It is difficult to find the best size for $\mathscr{D}_1$, as the best precision is achieved for σ_3 whereas the best recall is reached for σ_1. For Σ = standard deviation, it is σ_2 which seems to be the best suited. In any case, the number of connected components, Nb_{cc} decreases with σ, which means that increasing the size of superpixels enables to improve the spatial coherence of the detected objects. It is also true if we consider Haralick features instead of features based on operators from Mathematical Morphology. The number of connected components gets closer to the real number of objects in the images when we increase the size of the support (see Tab. 6.4). This time, the biggest studied size seems to be the best to compute these features, as highlighted by the classification performance.

What we can conclude on $\mathscr{D}_1$ is that the first three scales σ_1, σ_2 and σ_3 seem to be rather suitable whereas σ_4 is too large compared to the size of the objects we would like to detect. However, choosing the best size among the three appears to be difficult as we have seen that one can be better for a given criterion and another better for another criterion. Besides, it also depends on the considered feature. Features of texture such as Haralick have to be computed on supports which are rather *similar* and large enough for the texture to be fairly estimated. Hence, we must use in this case more regular waterpixels (increasing parameter k) and with high σ (which should not exceed the size of the largest objects though). This is illustrated in Fig. 6.13 where the feature "SumAverage" is applied on an image of $\mathscr{D}_2$. The larger and the more regular waterpixels are, the better the texture is estimated. As texture is very helpful to discrimininate tumoral and normal tissues in this database, it is important to use supports which will not bias its estimation. At this stage, we can already highlight the need not only for *multiple scales* but also for *multiple partitions* to improve segmentation results.

UC/CS	base	*P*	*R*	*F*	*J*	*Acc*	*Nb_cc*
SP mean	train	66	88	75	60	91	33±2
σ1	test	63	**86**	72	57	90	35±10
SP mean	train	72	86	78	64	93	13±1
σ2	test	65	81	72	57	91	14±3
SP mean	train	72	86	79	65	93	7±0
σ3	test	**67**	80	**73**	57	91	8±2
SP mean	train	70	85	77	62	92	6±1
σ4	test	64	76	69	53	90	**7±3**
SP std	train	62	87	73	57	90	47±3
σ1	test	60	**85**	71	55	89	50±16
SP std	train	67	88	76	61	91	13±1
σ2	test	**63**	**85**	**72**	**57**	90	13±4
SP std	train	67	88	76	61	91	7±1
σ3	test	62	84	71	55	90	9±3
SP std	train	67	86	75	60	91	6±0
σ4	test	61	82	70	54	89	**6±2**
UC = pixel	train	63	88	74	58	90	202±14
CS = pixel	test	61	87	72	56	90	203±71
UC = pixel	train	67	90	76	62	92	97±7
CS = windows (4), Σ = mean	test	64	**89**	**75**	**59**	91	101±33

Table 6.3: **Classification results with MOMA operators on $\mathscr{D}_1$.** Waterpixels: $\sigma_1 = 10$, $\sigma_2 = 20$, $\sigma_3 = 30$, $\sigma_4 = 40$; k=4.

Comparison with pixel and window supports

Figure 6.12 and Tab. 6.3 deal with the comparison between three computational supports: pixel, window and waterpixel. Note that the first two are used when classifying pixels, whereas the last one also constitutes the unit of classification *UC*. In Fig. 6.12, we can see that **waterpixels offer a better spatial coherence of detected objects than pixels and windows supports**. This observation is also quantitatively confirmed in Tab. 6.3, as the number of connected components *Nb_cc* is always smaller (thus closer to the real number of objects) for waterpixels (from 50 to 6) than for pixels (203) and windows (101). Classification performance is close to the one offered by pixels and windows, but the latter still shows an improvement due to the advantage of integrating multi-scale information.

σ	base	*P*	*R*	*F*	*J*	*Acc*	*Nb_cc*
σ1	train	66	88	76	61	91	38±1
	test	61	**83**	70	54	89	41±10
σ2	train	72	91	81	68	93	9±1
	test	64	82	72	56	90	13±1
σ3	train	70	90	79	65	93	9±1
	test	**65**	82	**73**	**57**	**91**	**8±2**

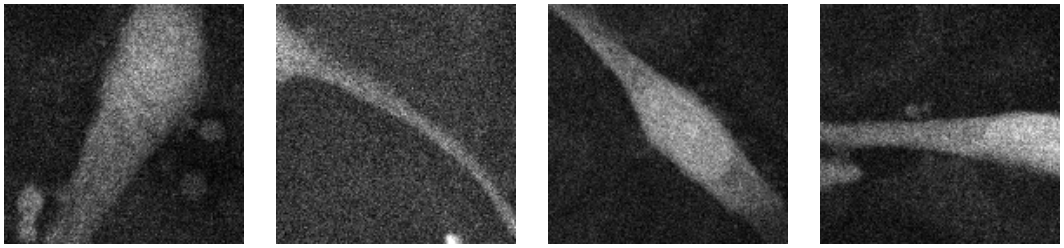
Table 6.4: **Classification results with Haralick operators on $\mathscr{D}_1$.** Waterpixels :$\sigma_1 = 10$,$\sigma_2 = 20$,$\sigma_3 = 30$; $k = 4$.

6.3 Discussion and conclusion

In conclusion, the advantage of superpixel classification over pixel classification (either with *CS* = pixel or *CS* = window) is to use a support which adapts to the content of the image, hence using a more relevant support for feature computation. The drawback of superpixel classification compared to pixel classification with windows support is that it does not allow to capture multi-scales information as it uses only one partition of SPs.

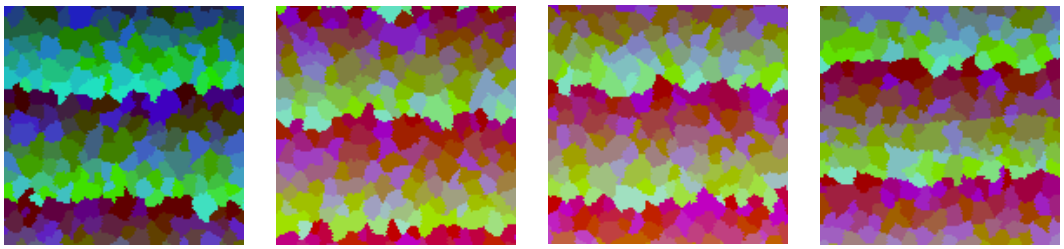
Some solutions exist to integrate multiple partitions with SP classification. Among them, one possibility is to perform a classification for each scale/partition and combine the different classification results with a given consensus rule. Another way would be to combine first all SP partitions into one and apply classification on the latter, as illustrated in Fig. 6.14. This second solution is equivalent to classifying each region in the new partition. However, regions are no longer regular, which is not good to compute features (as pointed out earlier, regularity is necessary for some features such as Haralick). Also, the works of Conze et al. [2016] and Geremia et al. [2013] integrate multi-scale information while performing superpixel classification (see the comparison in Chapt.7).

In the next chapter, we will introduce a novel way to integrate multiple partitions of SPs (not only multi-scale): the Superpixel-Adaptive Features (SAF).



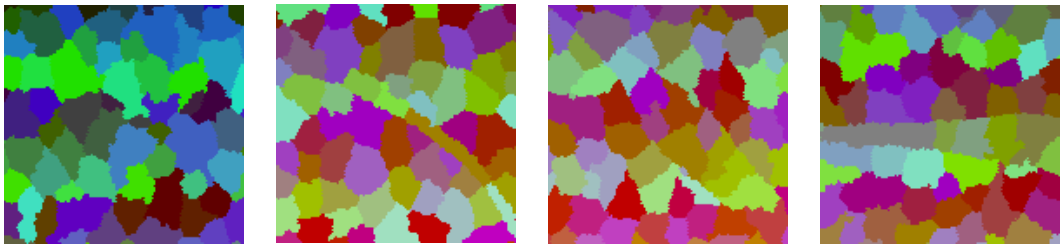
(a) zoom of im1 (b) zoom of im3 (c) zoom of im4 (d) zoom of im5

Waterpixels $\sigma = 10$, $k = 4$



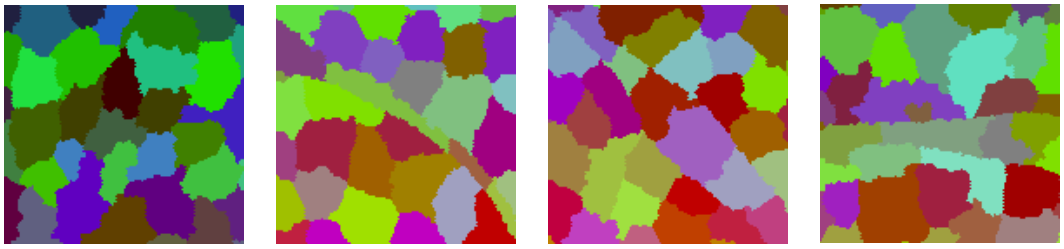
(e) WP of (a) (f) WP of (b) (g) WP of (c) (h) WP of (d)

Waterpixels $\sigma = 20$, $k = 4$



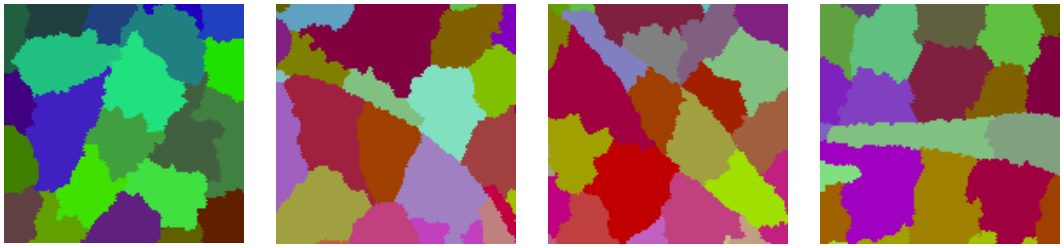
(i) WP of (a) (j) WP of (b) (k) WP of (c) (l) WP of (d)

Waterpixels $\sigma = 30$, $k = 4$



(m) WP of (a) (n) WP of (b) (o) WP of (c) (p) WP of (d)

Waterpixels $\sigma = 40$, $k = 4$



(q) WP of (a) (r) WP of (b) (s) WP of (c) (t) WP of (d)

Figure 6.8: Waterpixels applied to the L'Oréal database $\mathscr{D}_1$.

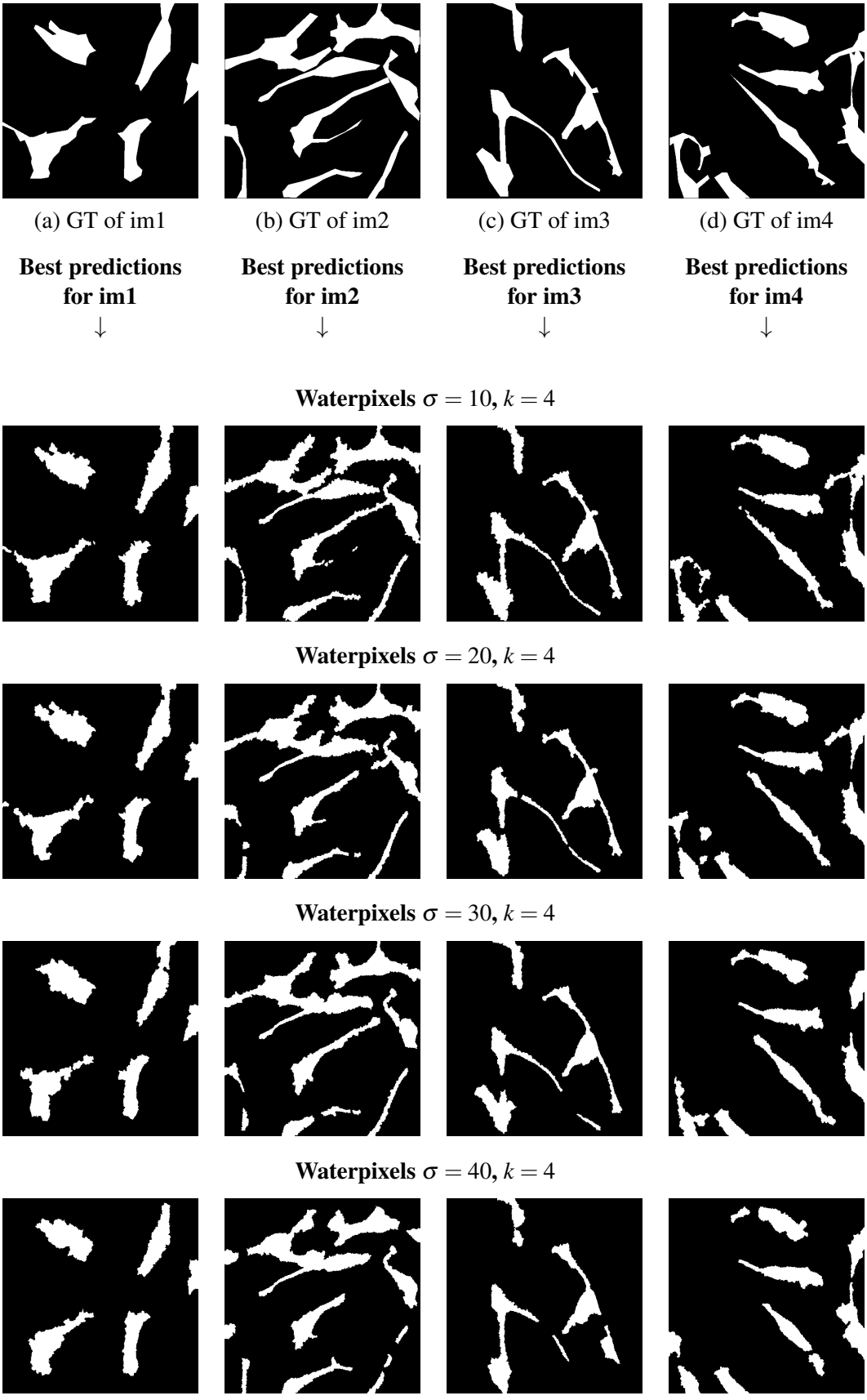


Figure 6.9: Best prediction at each scale for the L'Oréal database $\mathscr{D}_1$ (1/2).

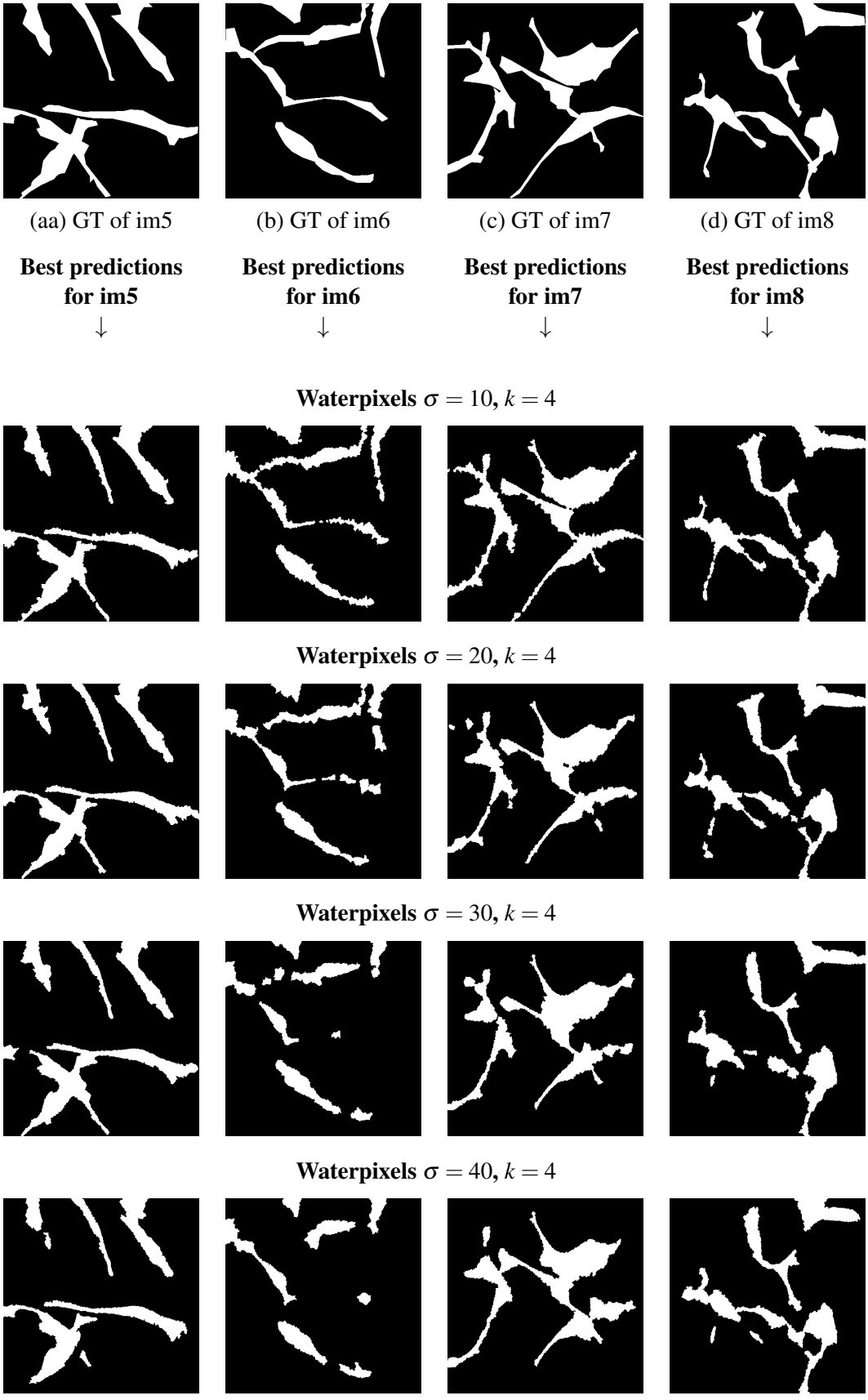
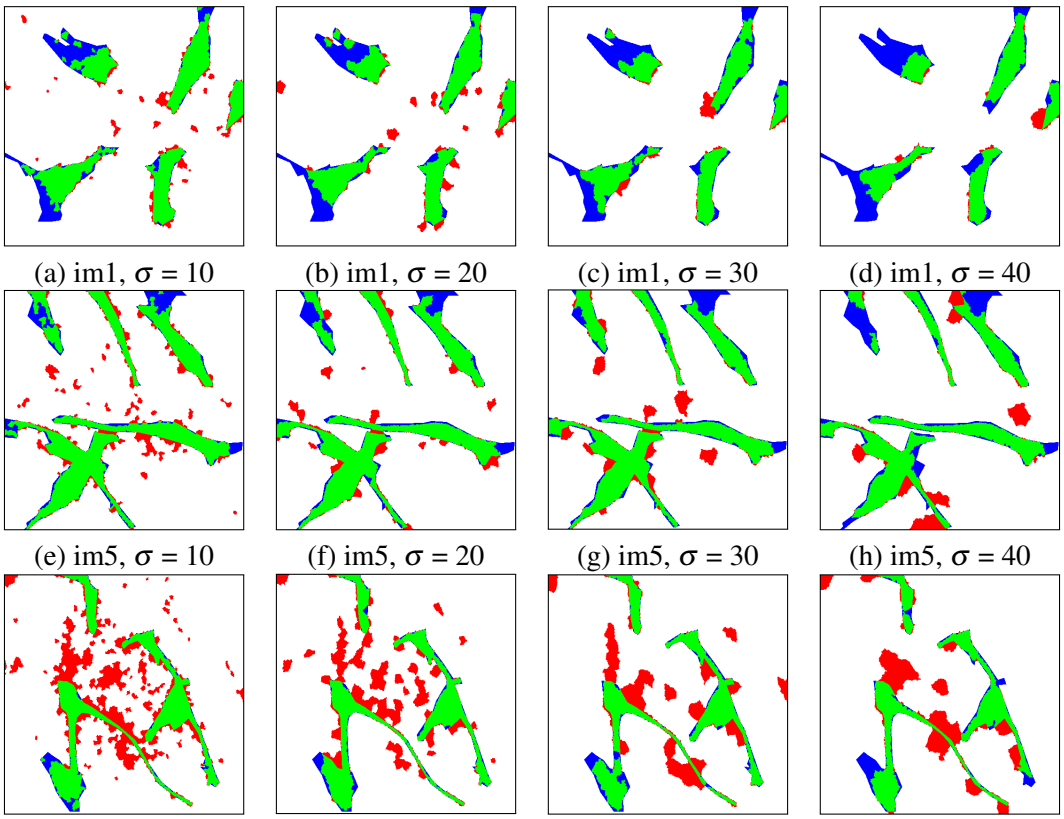


Figure 6.10: Best prediction at each scale for the L'Oréal database $\mathscr{D}_1$ (2/2).

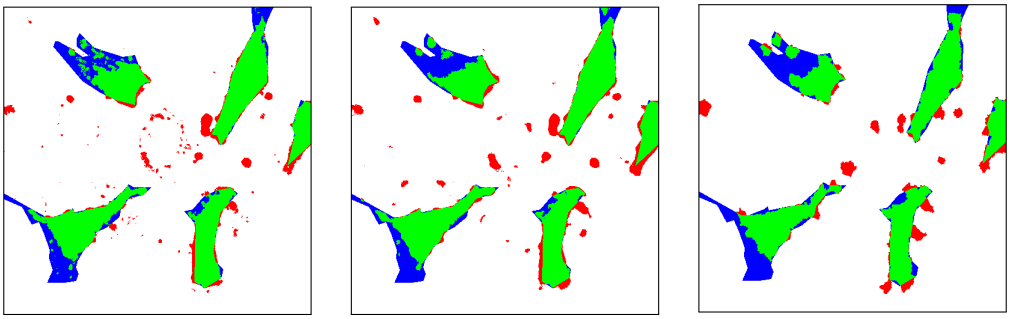


(a) im1, $\sigma = 10$ (b) im1, $\sigma = 20$ (c) im1, $\sigma = 30$ (d) im1, $\sigma = 40$

(e) im5, $\sigma = 10$ (f) im5, $\sigma = 20$ (g) im5, $\sigma = 30$ (h) im5, $\sigma = 40$

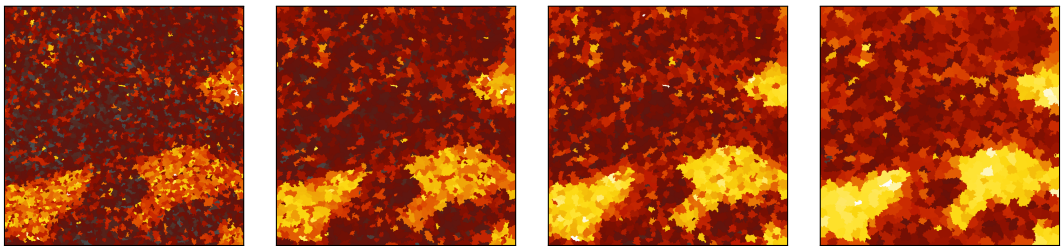
(i) im3, $\sigma = 10$ (j) im3, $\sigma = 20$ (k) im3, $\sigma = 30$ (l) im3, $\sigma = 40$

Figure 6.11: **Classification results with MOMA operators and waterpixels (σ=10, 20, 30, 40; k=4)**. Legend: green = true positive, white = true negative, red = false positive, blue = false negative.



(a) im1, CS = pixel (b) im1, CS = windows (4) (c) im1, CS = waterpixels ($\sigma = 20$, $k = 4$)

Figure 6.12: **Comparison of classification results (MOMA operators) with three different CS: pixel, window and waterpixel**. Legend: green = true positive, white = true negative, red = false positive, blue = false negative.



(a) CS = waterpixels $\sigma = 20$, $k = 10$ (b) CS = waterpixels $\sigma = 30$, $k = 10$ (c) CS = waterpixels $\sigma = 30$, $k = 50$ (d) CS = waterpixels $\sigma = 40$, $k = 50$

Figure 6.13: Haralick features need large as well as similar supports to be fairly estimated and compared (regions of different textures are more distinguishable in the last configuration).

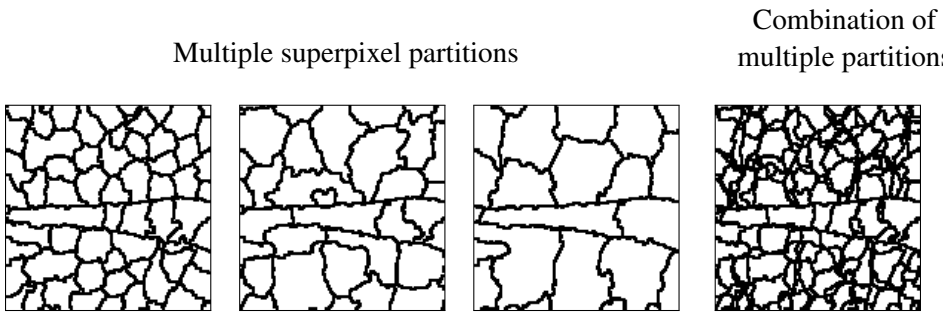


Figure 6.14: How to combine multiple partitions (second solution).

7 SAF: Superpixel-Adaptive Features

It always seems impossible until it's done.
Nelson Mandela

Résumé

Dans ce chapitre, nous proposons une nouvelle méthode permettant de répondre au sujet de thèse. Elle consiste à réaliser une classification de pixels, s'aidant cette fois-ci des superpixels (les waterpixels dans notre cas) pour calculer leurs descripteurs. Elle a l'avantage de permettre l'intégration de l'information sur un support plus pertinent que le pixel et la fenêtre, ainsi que de permettre une approche multi-échelle contrairement à la classification de superpixels habituelle.

*****

In this chapter, we present a novel general segmentation method based on pixel classification using superpixels as computational supports (see Sec. 7.1). This approach is then assessed on the three databases $\mathscr{D}_1$, $\mathscr{D}_2$ and $\mathscr{D}_3$ and compared to the other previously introduced pipelines (from Chap.3 and 6). It is also benchmarked against state-of-the-art segmentation methods designed specifically for these very databases.

7.1 Principle

In this chapter, we propose to perform pixel classification using superpixels as computational supports instead of windows. The proposed strategy is summarized in Fig. 7.1. As previously done for superpixel classification (Chapt. 6), superpixels are used as pertinent supports to compute features. However, this time, pixels constitute the classification units. We call these pixel features computed on SP supports the *Superpixel-Adaptive Features* (SAF).

Let us consider a given partition into superpixels $\{s_j\}_j$. For each superpixel s_j, one feature (or more) is calculated and the resulting value is stored in the vector of features of every pixel belonging to s_j. This enforces similarity between vectors of pixels belonging to the same superpixel and hence improves their chance of being classified with the same label.

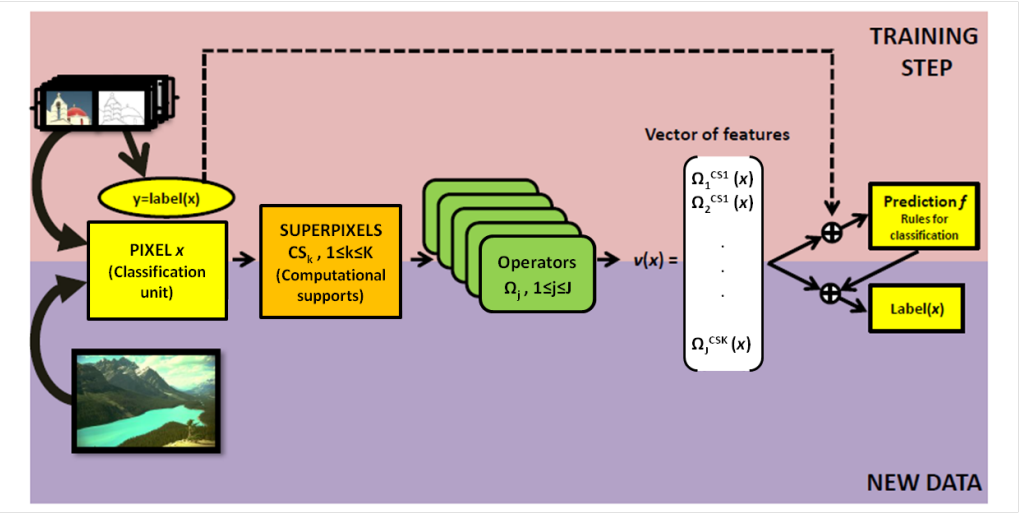


Figure 7.1: General scheme for our new approach: *Superpixel-Adaptive Features* (SAF).

Actually, if we use only one partition, vectors of pixels belonging to the same superpixel will be identical, which is equivalent to using the superpixel as classification unit. But here the process is repeated on different partitions of superpixels (computed with different parameters such as size, or even with different generation methods) in order to enrich the pixel's vector of features with information captured with different points of view. Figure 7.2 shows the similarity between the two pipelines of pixel classification using CS = windows and CS = superpixels (waterpixels in our case) respectively. They both enable multi-scale integration of the information; however, superpixels are more pertinent supports as they adapt to the image content.

Remark: Using the pixel itself as CS is a special case of SAF as it can be seen as a superpixel of size one pixel.

7.2 Comparison with other state-of-the-art methods

In this section, we present (to the limit of our knowledge) two interesting methods from the literature which succeed in integrating multi-scale information provided by different partitions of superpixels.

The work of Conze et al. [2016], done in parallel and published at the same time, is, for example, close to ours, as features are computed on superpixels of different sizes. This paper deals with liver tumor segmentation in dynamic contrast-enhanced CT scans. The need for multi-scale is motivated by the variety of tissues, as well as by the tediousness of the required (manually exhaustive) scale optimization which would have to be done by the user otherwise. In a first phase, a hierarchy of K partitions is built where each layer corresponds to a given SP size (decreasing from top to bottom). Starting from a coarse SLIC superpixel partition, each region is iteratively split into a set of smaller superpixels. This guarantees the hierarchical property between all SP partitions (which would not have hold if each partition, i.e. each scale, had been computed directly on the whole image, independently from the others). This ensures that all superpixels in the finest partition have exactly $K-1$ superpixel parents (one in each of the $K-1$ other partitions of the hierarchy). The former are taken as classification units and their features are computed on their $K-1$ parents as well as on themselves (leading to K supports). This idea is interesting as it

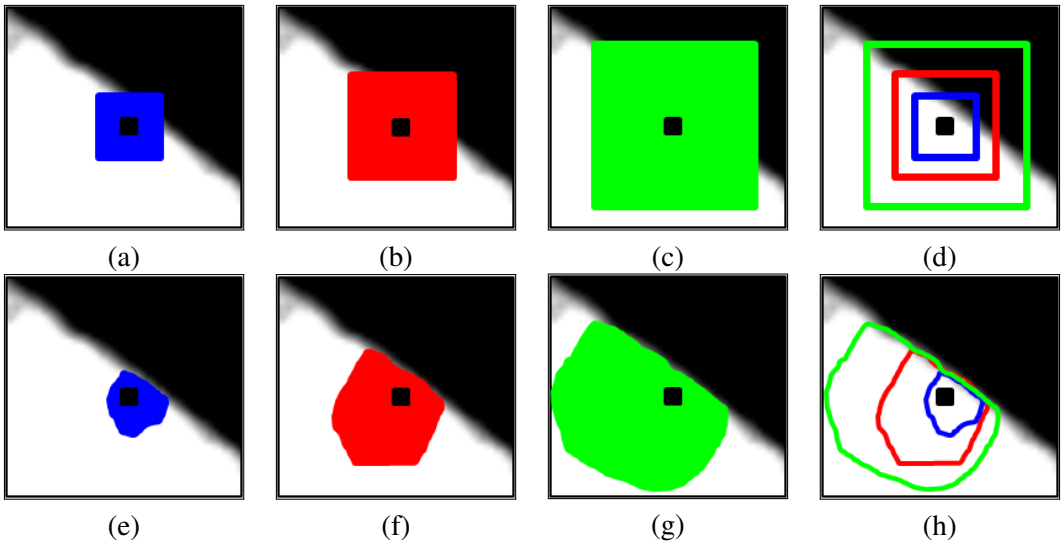


Figure 7.2: **Using multiple partitions to compute different *CS*: comparison between windows and superpixels (SAF features).** Legend: (a)–(c) windows with increasing size (radius r), (d) using windows (a), (b) and (c) ; (e)–(g) waterpixels of increasing size (step σ), (h) using waterpixels (e), (f) and (g).

reduces the computational cost by reducing the number of classification units (the set of smallest superpixels instead of all pixels). However, a hierarchy property is required between the partition, which limits the type of partitions used to enrich the vector of features. On the contrary, our method does not require any relation between partitions, therefore they can be generated not only with different scales but also by making more parameters vary, with different generation methods or even on different complementary channels (wavelengths, etc) according to our needs. A richer information is hence available with SAF.

Similarly to the work of Conze et al. [2016], the approach proposed by Geremia et al. [2013] is also based on superpixel classification with the help of multi-scale information captured on superpixels of different sizes (which are no more regular after fusion). This time, the hierarchy of SP partitions is built bottom-up (clustering superpixels of a given layer to obtain a coarser partition in the layer above). Let k be the index of a layer in this hierarchy, ranging from 0 (finest) to K (coarsest). The procedure starts by performing a superpixel classification with random forests (as explained in the previous chapter) where the classification units are SPs from the coarsest partition (layer K). Note that it is not possible to use the same procedure as in Conze et al. [2016] to integrate multi-scale information in this case: indeed, all nodes of a given layer, i.e. superpixels of a given partitions, do not have exactly the same number of children in the hierarchy. Hence, vectors of features corresponding to SPs to be classified would not have the same size, which cannot be handled by random forests. The multi-scale information is rather captured in a different way. The idea is that the computational support is always equal to the classification unit, however the classification unit can change in the RF learning procedure. Indeed, during training, if a set of classification units (a set of SPs from the coarser partition at the beginning) is not easy to discriminate with the available features computed on themselves, then the classification units are "split", i.e. become each one a set of smaller superpixels (present in the next lower layer) to be considered, in turn, as new classification units. Features are then recomputed on these regions of lower size and routed towards the next RF child node. Note that for each node, this new scale k is stored together with the split function $(f, \hat{\Theta})$ during training. The number of total classifica-

tion units hence increases during the procedure, but at the end it is still less than the number of SPs in the finest scale (layer 0). The aim here is to quickly classify "easy" regions (such as large homogeneous ones) and only consider finer scales if needed to better classify "difficult" (mixed) regions. Contrary to the previous approach as well as to ours, multiple partitions are integrated in the random forest procedure, whereas in the other two cases, the integration is independent from the machine learning method which can thus be changed without impacting the construction of the pipeline. Once again, this technique only enables to catch multi-scale information (and not information from other type of partitions) because it relies on a hierarchy.

In conclusion, we have presented two methods which are based on *superpixel classification* and which integrate multi-scale information thanks to superpixels of different sizes. Contrary to the SAF approach (*pixel classification*), both require a relation of hierarchy between the different SPs partitions, which limits the type of partitions used, and thus the diversity of captured information.

7.3 Preliminary study on $\mathscr{D}_1$

As in Chap. 3 and 6, we start by some preliminary studies on the L'Oréal database $\mathscr{D}_1$. We remind that the random forest parameters have been set to consistent values (100 trees, 100 data at least in each leaf at the end of training) and that a leave-one-out procedure is applied to evaluate the segmentation performance. Four sizes of waterpixels will be used to compute SAF in the pixel classification pipeline: $\sigma \in \{10, 20, 30, 40\}$ and $k = 4$.

7.3.1 Best achievable prediction with waterpixels

It is interesting to note that the proposed approach is equivalent to a classification pipeline where the UCs are the (irregular) regions obtained by combining the four SP partitions (see Fig. 6.14 in Chap.6 for an illustration). However, this time, features are guaranteed to be computed on *regular* supports, i.e. waterpixels, which overcomes the issue arising when directly performing region classification (as in Chap.6) of such a partition.

Therefore, we first simulate perfect classification by assigning to each of these regions the label of the majority class among its pixels (as done in order to compute the groundtruth in the training phase), which gives us the best performance reachable by this approach. Figure 7.3 shows the eight resulting images of $\mathscr{D}_1$. We can see that objects are almost perfectly detected. Very few misclassifications appear, mainly on objects contours (due to noise and GT own imprecision) and on elongated object parts that are thinner than the smaller SP size used (see the case of im2). Quantitative results, presented in Tab. 7.1, confirm that the SAF approach outperforms single-scale superpixel classification (even the smallest one) on the five classification criteria used: precision P, recall R, F-score F, Jaccard index J and accuracy Acc. This observation validates the intuition that using multiple scales enables to capture more information on object contours and hence helps to more accurately classify the pixels.

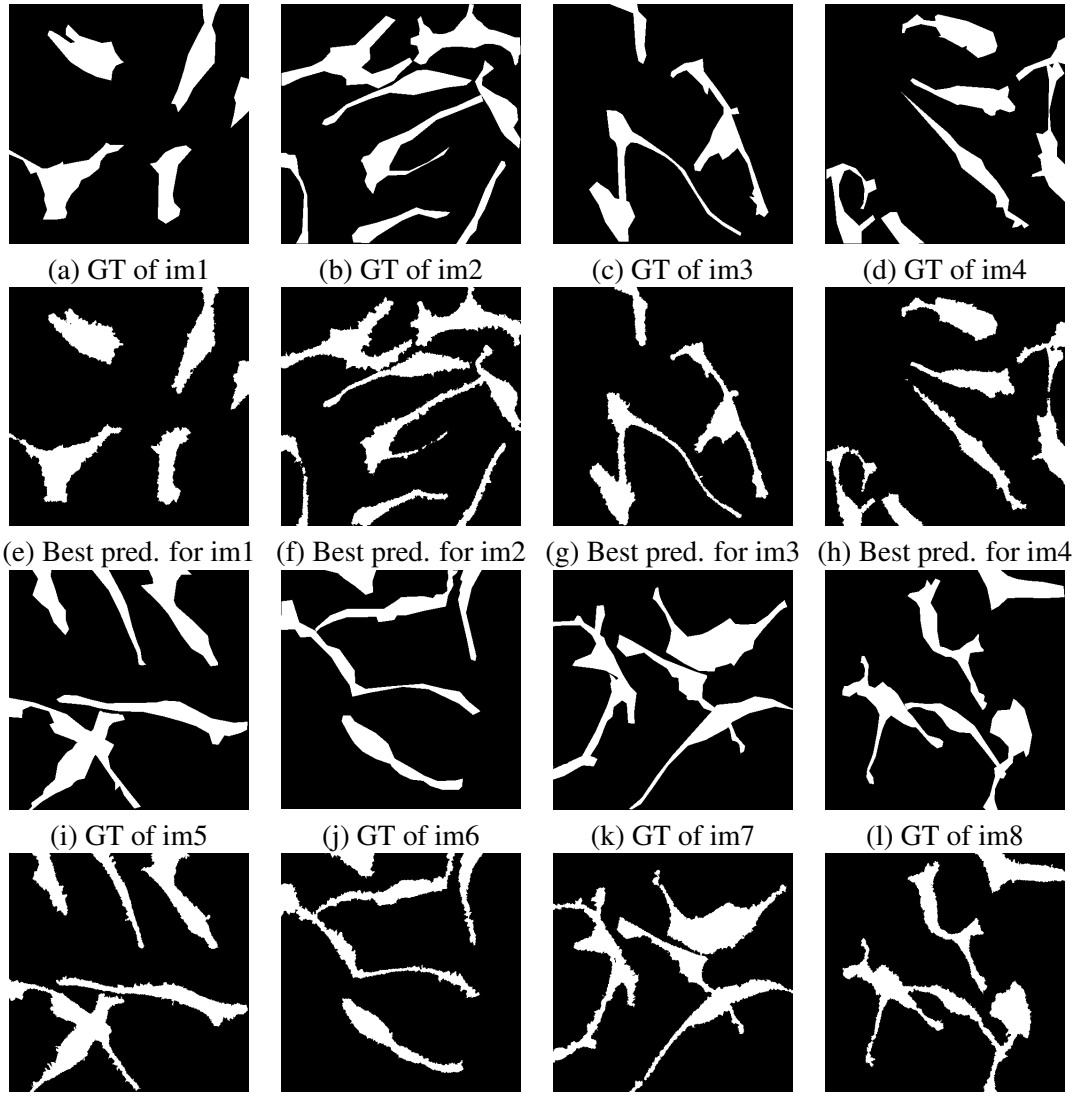
7.3.2 Study with different families of features

We now use SAF features to compute the feature vector $v(p)$ of each pixel p.

Table 7.2 presents the quantitative results for MOMA operators. For each integrator Σ = mean and Σ = standard deviation (notation: *std*), we can observe that the SAF approach outperforms the four pipelines of superpixel classification in terms of classification performance. This confirms

Measure	σ	im1	im2	im3	im4	im5	im6	im7	im8	all images
Precision	10	93	93	93	93	95	90	92	93	93
	20	91	93	90	91	95	86	91	93	91
	30	89	92	92	92	93	89	91	91	91
	40	91	91	89	91	95	87	87	90	91
	4 σ	**95**	**95**	**95**	**95**	**96**	**94**	**95**	**95**	**95**
Recall	10	94	89	90	91	91	90	92	92	91
	20	89	84	87	88	86	81	86	87	86
	30	84	79	83	80	86	68	84	80	81
	40	80	73	73	76	78	62	80	83	76
	4 σ	**95**	**92**	**93**	**93**	**93**	**93**	**93**	**94**	**93**
F-score	10	93	91	91	92	93	90	92	92	92
	20	90	88	88	89	90	83	88	90	88
	30	86	85	87	86	89	77	87	85	86
	40	85	81	80	83	86	72	83	86	83
	4 σ	**95**	**93**	**94**	**94**	**94**	**93**	**94**	**94**	**94**
Jaccard index	10	88	84	84	85	86	82	85	86	85
	20	83	79	79	81	82	72	79	81	80
	30	76	74	78	75	80	62	77	74	75
	40	74	68	68	70	75	54	72	76	70
	4 σ	**91**	**87**	**89**	**89**	**89**	**87**	**89**	**90**	**89**
Accuracy	10	98	96	98	97	98	98	97	98	98
	20	97	95	98	97	97	96	96	97	97
	30	96	94	98	96	96	95	96	96	96
	40	96	93	96	95	95	94	95	96	95
	4 σ	**99**	**97**	**99**	**98**	**98**	**98**	**98**	**98**	**98**

Table 7.1: Ideal classification: performance on $\mathscr{D}_1$.



(a) GT of im1 (b) GT of im2 (c) GT of im3 (d) GT of im4

(e) Best pred. for im1 (f) Best pred. for im2 (g) Best pred. for im3 (h) Best pred. for im4

(i) GT of im5 (j) GT of im6 (k) GT of im7 (l) GT of im8

(m) Best pred. for im5 (n) Best pred. for im6 (o) Best pred. for im7 (p) Best pred. for im8

Figure 7.3: Best achievable prediction for the L'Oréal database $\mathscr{D}_1$ with four scales of SAF.

the advantage of using a multi-scale method instead of a single-scale one. Taking into account different integrators also improves classification performance, as shown in the last row of the table ($F = 78\%$, $J = 63\%$). The same observations can be done for Haralick features (see Tab. 7.3).

Thanks to this richer information, the SAF approach is, as expected, more efficient than the pixel classification with pixel support. With MOMA operators, F increases from 72% to 76% with mean integrator (or even 78% when using both integrators), J increases from 56% to 61% (or 63% respectively). Interestingly, it also shows an improvement compared to multi-scale windows support, apart from recall which is equivalent or a little smaller ($R_{win} = 89\%$ vs. $R_{saf} = 86\%$ for Σ = mean): the precision is better by 6%, the Jaccard index by 3% and the accuracy by 1%. Again, we can highlight the same observations for identity and standard operators[1] (Tab. 7.4 and Tab. 7.5).

As far as the spatial coherence of detected object is concerned, the number of connected components of SAF with MOMA operators is slightly higher than (Σ = mean) or equivalent (Σ = std) to

[1] Operators presented in Sec. 2.2.2.

the single scale pipelines. This is due to the fact that there are more UCs (regions) to classify in the former approach as we consider all regions created by the intersection of the four SPs partitions. Still, this number stays reasonably low compared to pixel and windows supports ($Nb\_cc^{pix} = 203$ and $Nb\_cc^{win} = 101$ vs. $Nb\_cc^{SAFmean} = 40$, $Nb\_cc^{SAFstd} = 30$). Using both mean and std integrators enables to further decrease this number as using different types of information leads to fewer region misclassifications. These observations are also confirmed with identity and standard operators (see Tab. 7.4 and Tab. 7.5).

Qualitative results can be seen in Fig. 7.4. The first example is presented in Fig. 7.4.a and its superimposed GT (in red) in Fig. 7.4.b. We can already see that the manual groundtruth does not perfectly coincide with the object in the image. The third row presents the classification results when using SAF with Σ = mean. The object and the background are nearly recovered, except for the frontier area. We can distinguish different types of errors:

1. Pixels considered as misclassified compared to the GT but are in fact well classified according to the image. Example: false positives and false negatives in the upper right part of the melanocyte arm.
2. Some superpixels (or fragments of them in the SAF case) overlapping object contours due to the high level of noise in the image. Thus, a certain number of their pixels will be misclassified. Example: aggregated false positives in the lower left part.
3. Good superpixels which are nonetheless misclassified by RF based on their vector of features (in difficult areas, for example when noise prevails). Example: the isolated connected component considered as a false positive in the upper left part.

Adding features computed with another integrator (Σ = std) increases performance by decreasing the number of good SP misclassifications (3rd case), as shown in Fig. 7.4.c. Finally, Fig. 7.4.d shows the tendency of the window computational support to enlarge object with numerous false positives on borders (even when removing type 1 errors): misclassifications on borders were expected for this approach as explained in Chapt. 3. These observations can also be made for the other two crops. Note that the third example emphasizes the fact that SAF succeed in catching more accurately the shape of the melanocyte termination than the GT itself (artificially increasing the number of type 1 errors).

7.4 Final results on the three databases

We now proceed to the construction of the new pipeline based on SAF: $\mathscr{P}_{saf}$. It will be assessed and compared to the previous considered general approaches, $\mathscr{P}_{pix}$ and $\mathscr{P}_{win}$, and specific methods (all presented in Chap. 3) for the three databases. A recap of the features used is provided in Tab. 7.6. Random Forest parameter values are optimized by cross-validation (see appendix A.1).

7.4.1 Presentation of the final results

Qualitative analysis: Figures 7.7, 7.8 and 7.9 show examples of classification for the three databases and the three pipelines. In the case of Fig. 7.7, where texture is a pertinent characteristic, the use of adaptive supports such as superpixels leads to a better discrimination between the classes, as shown by the profile plot along the yellow line. On the $\mathscr{D}_1$ database (see Fig. 7.8), the SAF present less false positives than the other two pipelines, with a better spatial coherence of detected objects. Figure 7.5 opposes SAF to specific methods designed for $\mathscr{D}_1$: Serna et al. [2014] and MSER (Matas et al. [2002]). The MSER presents a low recall and appears to favor compact objects (corresponding more or less to the nuclei of melanocytes), which is not desired here. Serna et al. [2014], on the contrary, is specifically designed to detect well elongated objects, therefore

CS	base	*P*	*R*	*F*	*J*	*Acc*	*Nb_cc*
pixel	train	63	88	74	58	90	202±14
	test	**61**	**87**	**72**	**56**	**90**	**203±71**
windows (4)	train	67	90	76	62	92	97±7
	test	**64**	**89**	**75**	**59**	**91**	**101±33**
SP mean σ1	train	66	88	75	60	91	33±2
	test	63	86	72	57	90	35±10
SP mean σ2	train	72	86	78	64	93	13±1
	test	65	81	72	57	91	14±3
SP mean σ3	train	72	86	79	65	93	7±0
	test	67	80	73	57	91	8±2
SP mean σ4	train	70	85	77	62	92	6±1
	test	64	76	69	53	90	7±3
SAF mean (4)	train	73	90	81	68	93	35±2
	test	**68**	**86**	**76**	**61**	**92**	**40±8**
SP std σ1	train	62	87	73	57	90	47±3
	test	60	85	71	55	89	50±16
SP std σ2	train	67	88	76	61	91	13±1
	test	63	85	72	57	90	13±4
SP std σ3	train	67	88	76	61	91	7±1
	test	62	84	71	55	90	9±3
SP std σ4	train	67	86	75	60	91	6±0
	test	61	82	70	54	89	6±2
SAF std (4)	train	72	91	80	67	93	26±2
	test	**68**	**87**	**76**	**62**	**92**	**30±8**
SAF mean+std (4)	train	75	92	82	70	94	23±2
	test	**70**	**87**	**78**	**63**	**92**	**25±7**

Table 7.2: $\mathscr{D}_1$: MOMA ($\sigma_1 = 10$, $\sigma_2 = 20$, $\sigma_3 = 30$, $\sigma_4 = 40$).

σ	*P*	*R*	*F*	*J*	*Acc*	*Nb_cc*
σ1	61	83	70	54	89	41±10
σ2	64	82	72	56	90	13±1
σ3	65	82	73	57	91	**8±2**
SAF: 3 σ	**67**	**86**	**75**	**61**	**91**	32±8

Table 7.3: $\mathscr{D}_1$: Haralick features ($\sigma_1 = 10$, $\sigma_2 = 20$, $\sigma_3 = 30$). Comparison between superpixel classification (one-scale) and SAF (multi-scale). Results are given on the test subset.

CS	*P*	*R*	*F*	*J*	*Acc*	*Nb_cc*
pixel	54	75	63	46	86	8153±147
windows (4)	60	**87**	71	55	89	380±160
SAF (4)	**62**	86	**72**	**56**	**90**	**44±10**

Table 7.4: $\mathscr{D}_1$: Identity ($\sigma_1 = 10$, $\sigma_2 = 20$, $\sigma_3 = 30$, $\sigma_4 = 40$). *UC*: pixel. Results are given on the test subset.

CS	*P*	*R*	*F*	*J*	*Acc*	*Nb_cc*
pixel	60	**87**	71	55	89	179± 68
windows (4)	61	**87**	72	56	89	129±32
SAF mean (4)	65	85	74	58	**91**	54±11
SAF std (4)	64	86	73	58	90	**32±8**
SAF mean+std (4)	**66**	86	**75**	**60**	**91**	**32±8**

Table 7.5: $\mathscr{D}_1$: standard operators (σ_1 = 10, σ_2 = 20, σ_3 = 30, σ_4 = 40). *UC*: pixel. Results are given on the test subset.

Pipeline	Family	#	scales		*CS*-scale		=	Sum
$\mathscr{P}_{pix}$	Identity	1	x1		x1	pixel	1	
	MOMA	7	x3	$size_{SE} \in \{1,3,5\}$	x1	pixel	21	
	Gabor	1	x(3x4x3)	$bd \in \{1,2,3\}$	x1	pixel	36	**58**
				$\theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, 3\frac{\pi}{4}\}$				
				$freq \in \{0.3, 0.5, 0.7\}$				
$\mathscr{P}_{win}$	Identity	1	x1		x4	$r \in \{5,10,15,20\}$	4	
	MOMA	7	x3	$size_{SE} \in \{1,3,5\}$	x4	$r \in \{5,10,15,20\}$	84	
	Gabor	1	x(3x4x3)	$bd \in \{1,2,3\}$	x4	$r \in \{5,10,15,20\}$	144	**232**
				$\theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, 3\frac{\pi}{4}\}$				
				$freq \in \{0.3, 0.5, 0.7\}$				
$\mathscr{P}_{saf}$	Identity	1	x1		x4	$\sigma \in \{10,20,30,40\}$	4	
	MOMA	7	x3	$size_{SE} \in \{1,3,5\}$	x4	$\sigma \in \{10,20,30,40\}$	84	
	Gabor	1	x(3x4x3)	$bd \in \{1,2,3\}$	x4	$\sigma \in \{10,20,30,40\}$	144	**232**
				$\theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, 3\frac{\pi}{4}\}$				
				$freq \in \{0.3, 0.5, 0.7\}$				

Table 7.6: Recap of the features used for pipelines $\mathscr{P}_{pix}$, $\mathscr{P}_{win}$ and $\mathscr{P}_{saf}$. Note: *MOMA* stands for "mathematical morphology operators", *Gabor* for "Gabor filters".

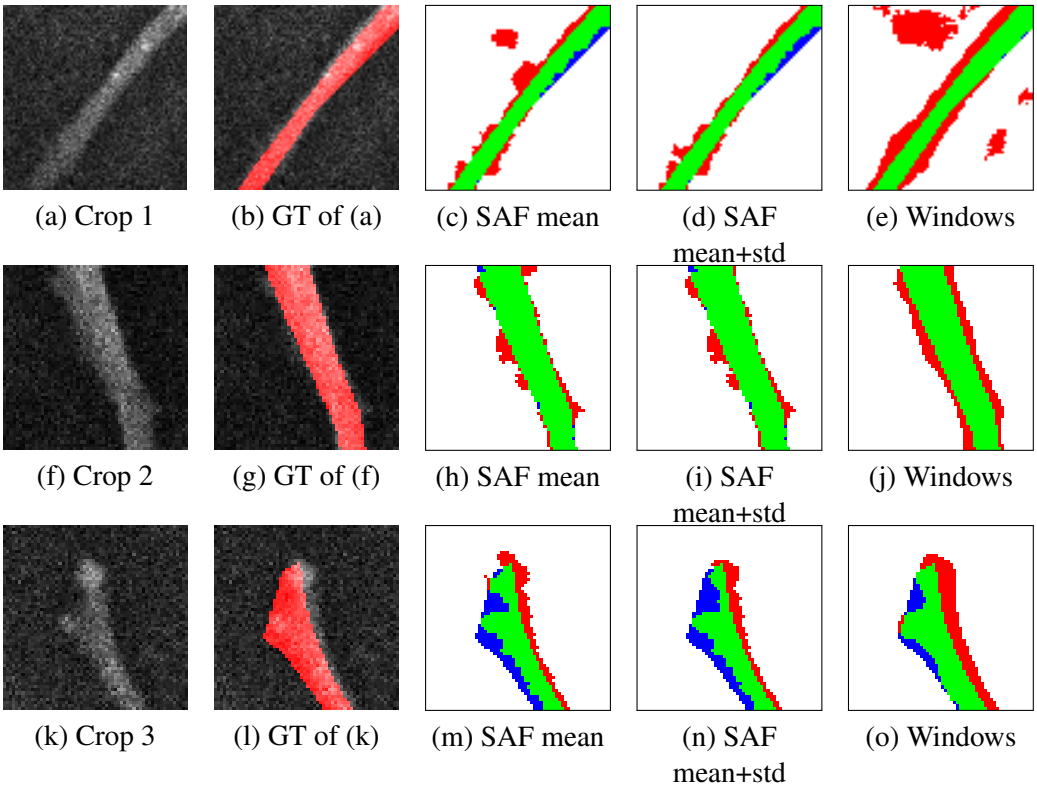


Figure 7.4: Classification results for $\mathscr{D}_1$: comparison between SAF and window approaches.

misses more compact ones (see the upper left of the image). Compared to this specific method, SAF detect more components (false and true positives). For the last database, $\mathscr{D}_3$, performances on contours appear to be not as good as for the other pipelines. It seems that superpixels lying on the contour (with a potential overlap) were more challenging to classify, leading to false positives with a low threshold, or false negatives with a high threshold, on the probability map.

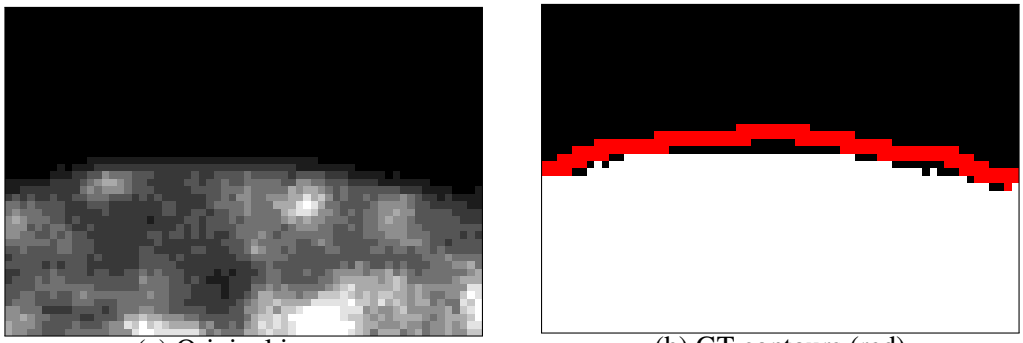
Quantitative analysis: As in Chapt. 3, we present the six evaluation criteria and the ROC curve for all these configurations (see Fig. 7.10 to 7.13). We had previously noticed that a wider support, such as the window, was advantageous for images where the pixel information is not enough, where we need more contextual information on it (for example the texture). Indeed, SAF are also efficient on the $\mathscr{D}_2$ database, and their capacity to adapt to the image content improves the classification performances obtained by the window approach. For database $\mathscr{D}_1$, the three pipelines are rather equivalent, and reach (Serna et al. [2014]) or outperform (MSER) the specific segmentation methods. For database $\mathscr{D}_3$, SAF overall classification performances are good, better than the specific method, but are slightly lower than $\mathscr{P}_{pix}$ and $\mathscr{P}_{win}$. This behavior is discussed in 7.4.2. Finally, SAF always present a better spatial coherence than the other two approaches, whatever the database is.

7.4.2 Discussion of the results

We have seen that the SAF approach performs well on the three databases and reaches or outperforms the specific segmentation methods. Compared to other general segmentation approaches, $\mathscr{P}_{pix}$ and $\mathscr{P}_{win}$, it always offers a better spatial coherence of detected objects, which is advantageous for segmentation purposes. The pertinence of using a support which is large enough and adapts to the image content is confirmed on images where the information provided by the pixel alone is not enough to assign this one to a class or an other (for example when texture is a de-

(a) GT of im1 (b) MSER for im1 (c) Serna for im1 (d) SAF for im1

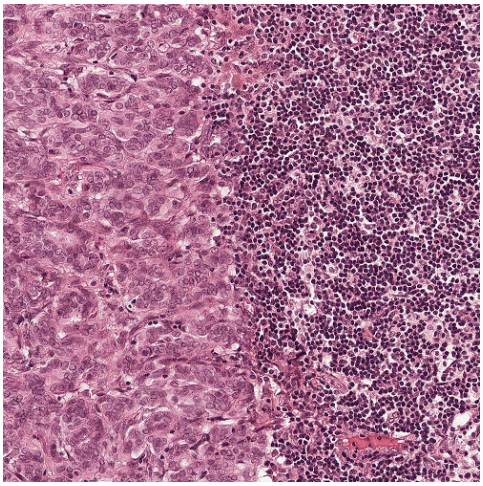
Figure 7.5: Classification results for im1 of $\mathscr{D}_1$: comparison between specific methods (Serna et al. [2014] and MSER) and the general method we propose (fourth row, probability map thresholded at 0.7).



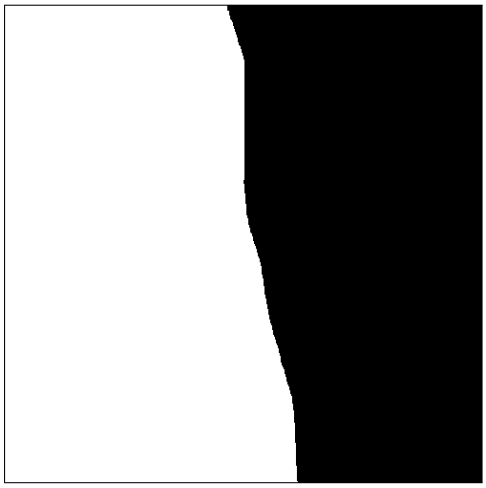
(a) Original image (b) GT contours (red) superimposed on thresholded (a) (white)

Figure 7.6: Groundtruth imprecision for $\mathscr{D}_3$ database: illustration on a cell nucleus contour. Image (a) has been thresholded at 1 (lowest possible threshold) and appears in white in image (b). The GT should be equal or smaller than this white region. However, we can see the the GT contours (shown in red) exceed this area by some pixels.

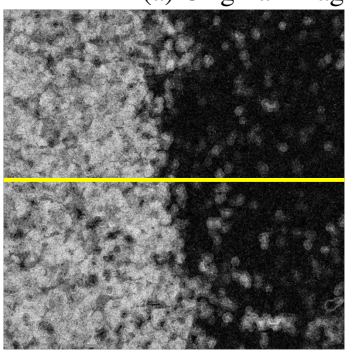
terminant feature as in $\mathscr{D}_2$). Otherwise, the three pipelines should give equivalent classification results, as for $\mathscr{D}_1$. The case of $\mathscr{D}_3$, where SAF performances are slightly lower, is interesting, as we could have expected them to show the same behavior as for $\mathscr{D}_1$. Different reasons could be evoked to explain this phenomenon. First, there may be errors on waterpixels' contour (overlapping rather than adhering to contours) due to the weakness of the gradient signal. To check this potential source of error, one should perform the same experiments with ideal superpixels (for example an over-segmentation of the groundtruth itself). In any case, using partitions created by different superpixel generation methods (based on different principles) instead of only one type could make SAF more robust to the change of database. Another possibility, if superpixels adhere well to contours, would be that their classification by random forests is not entirely efficient (with a potential over-fitting). A third possibility resides in the fact that our evaluation may be biased by the imprecision of the groundtruth. We have seen it for the $\mathscr{D}_1$ database (see Fig. 7.4), but it is all the more true for the $\mathscr{D}_3$ database where the groundtruth systematically integrates, in its objects, pixels from the background with value 0 in the original image (see Fig. 7.6). Experiments should be performed again allowing an error margin of some pixels, as we did for superpixel evaluation on the Berkeley segmentation database (see Chapt. 4). This margin should be set according to the imprecision of the GT defined for each database (around 2 pixels in the case of $\mathscr{D}_3$ for example).



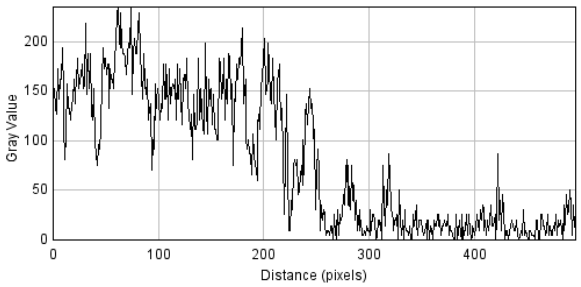
(a) Original image



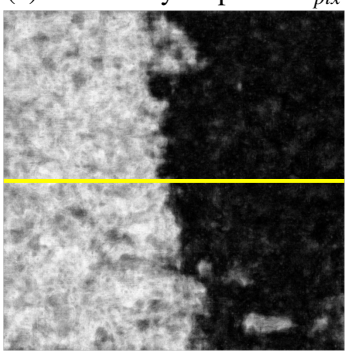
(b) GT of (a)



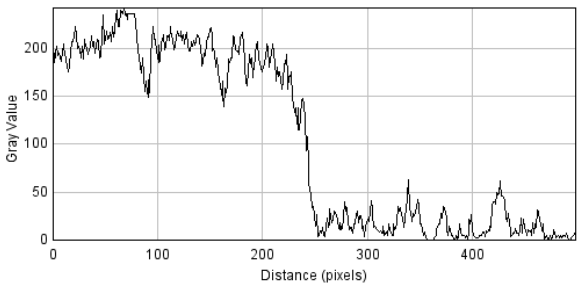
(c) Probability map for $\mathscr{P}_{pix}$



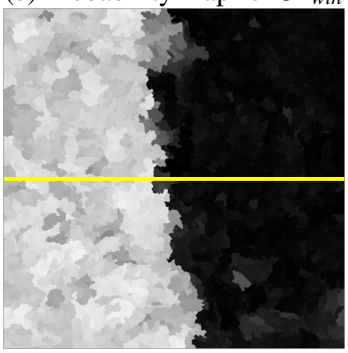
(d) Profile for $\mathscr{P}_{pix}$



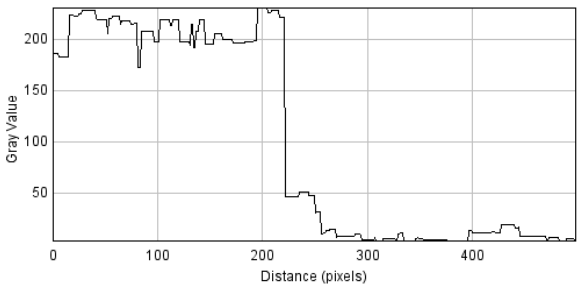
(e) Probability map for $\mathscr{P}_{win}$



(f) Profile for $\mathscr{P}_{win}$

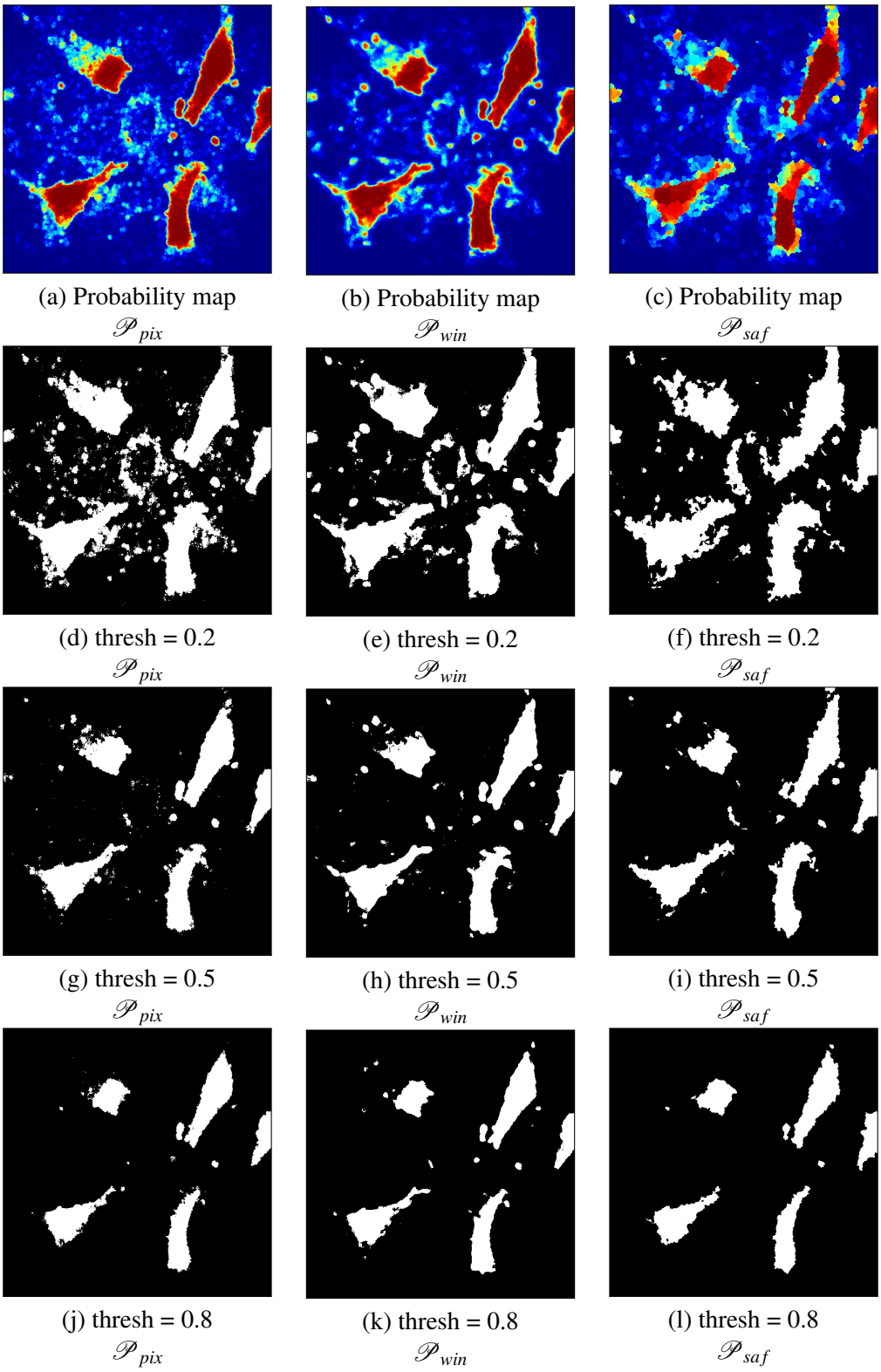


(g) Probability map for $\mathscr{P}_{saf}$



(h) Profile for $\mathscr{P}_{saf}$

Figure 7.7: **Final classification results an image of $\mathscr{D}_2$.** Grey level profiles are evaluated on the yellow line.



(a) Probability map $\mathscr{P}_{pix}$ (b) Probability map $\mathscr{P}_{win}$ (c) Probability map $\mathscr{P}_{saf}$

(d) thresh = 0.2 $\mathscr{P}_{pix}$ (e) thresh = 0.2 $\mathscr{P}_{win}$ (f) thresh = 0.2 $\mathscr{P}_{saf}$

(g) thresh = 0.5 $\mathscr{P}_{pix}$ (h) thresh = 0.5 $\mathscr{P}_{win}$ (i) thresh = 0.5 $\mathscr{P}_{saf}$

(j) thresh = 0.8 $\mathscr{P}_{pix}$ (k) thresh = 0.8 $\mathscr{P}_{win}$ (l) thresh = 0.8 $\mathscr{P}_{saf}$

Figure 7.8: **Final classification results on im1 of $\mathscr{D}_1$.**

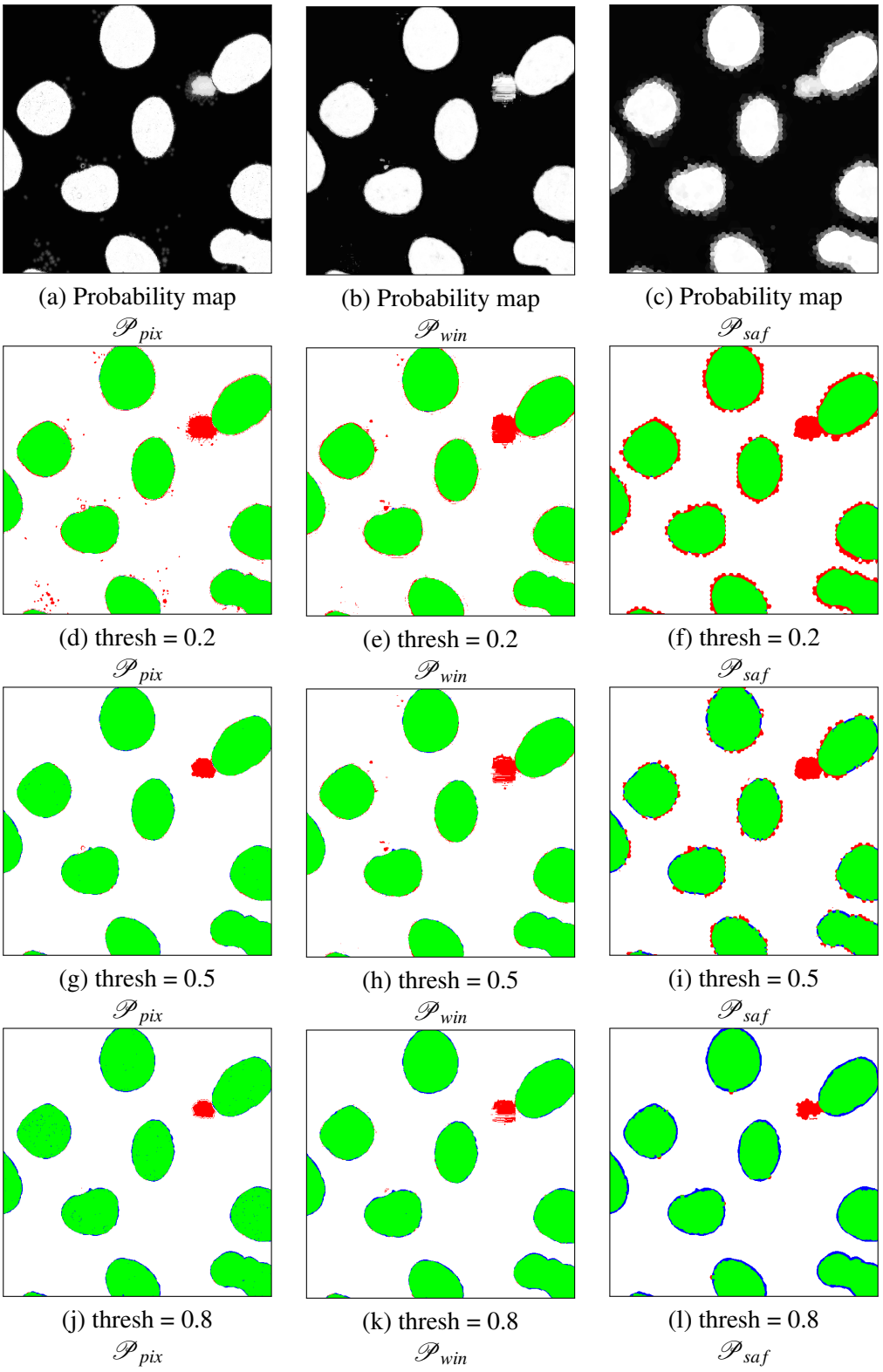


Figure 7.9: **Final classification results on a crop of image *dna-30* of $\mathscr{D}_3$.** Legend: TP, TN, FP and FN are shown in green, white, red and blue respectively.

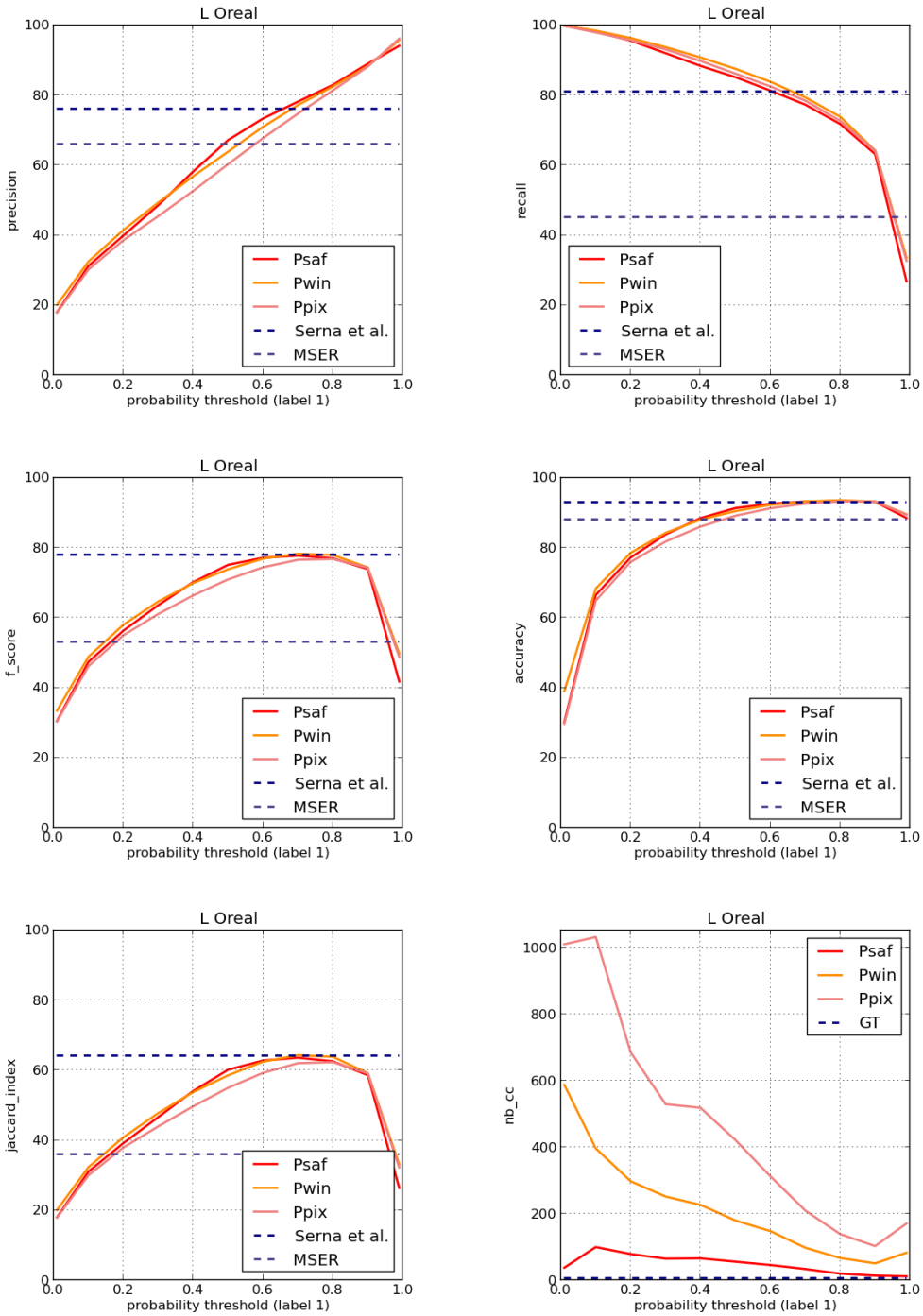


Figure 7.10: Quantitative results for $\mathscr{D}_1$.

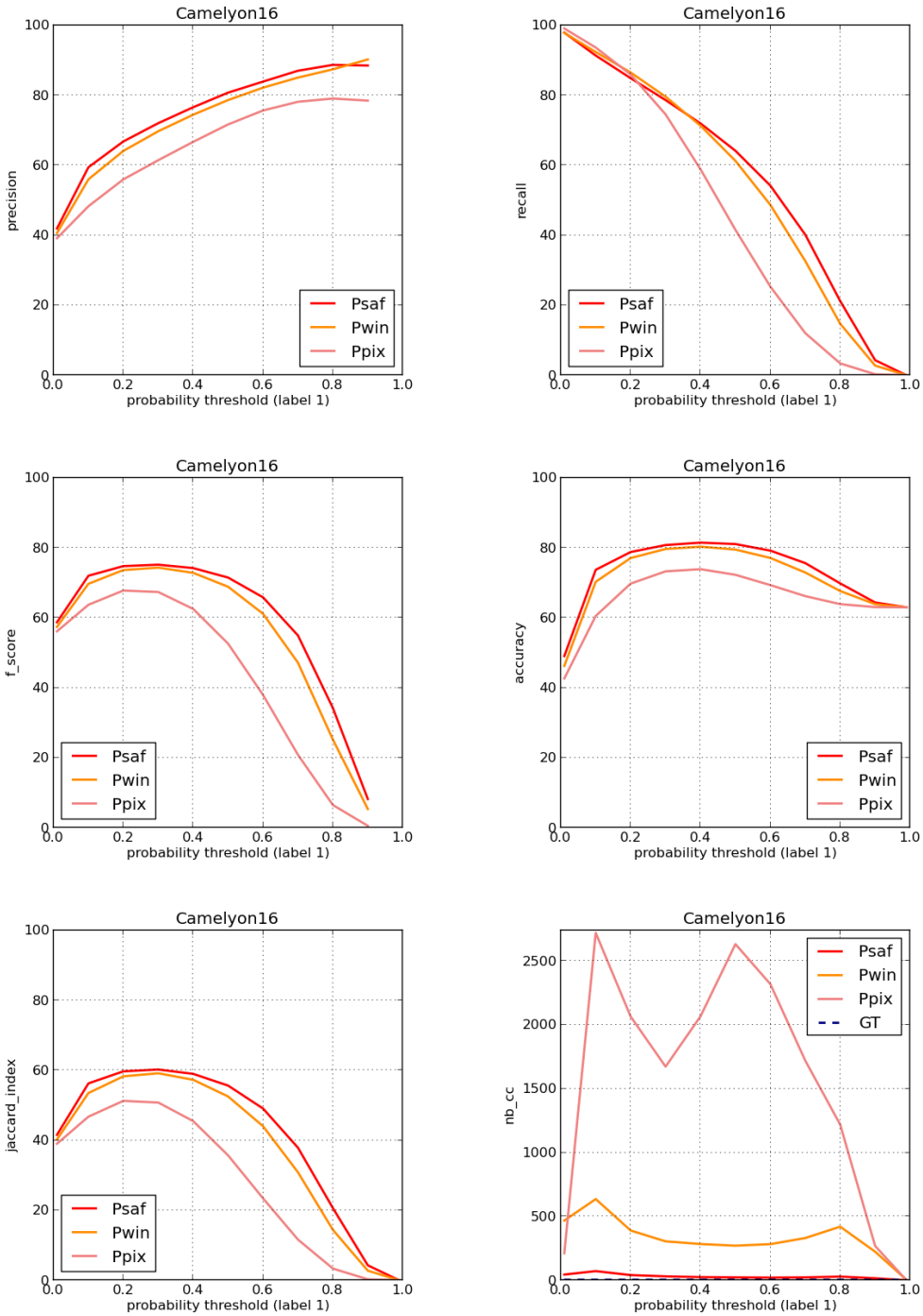


Figure 7.11: Quantitative results for $\mathscr{D}_2$.

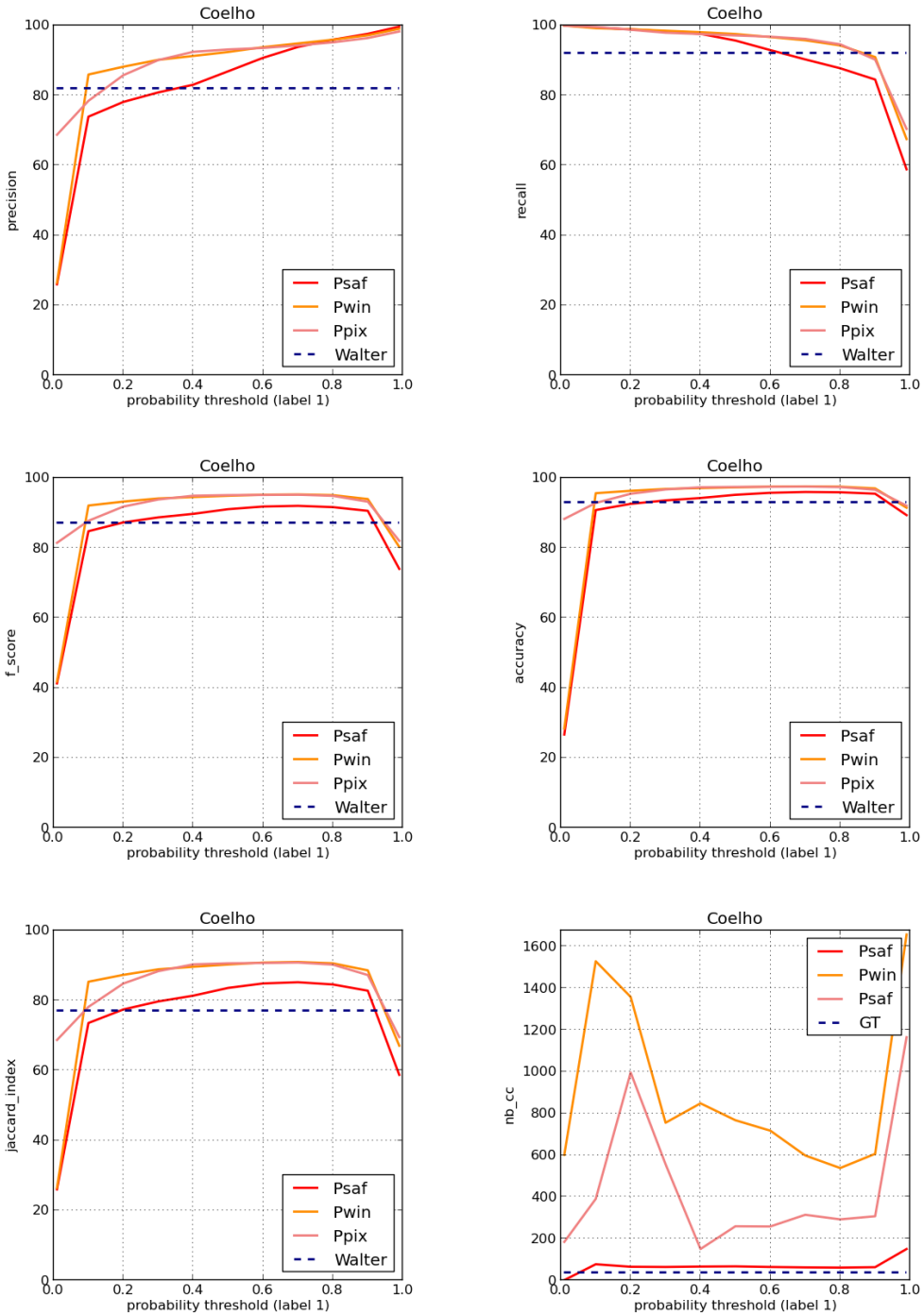
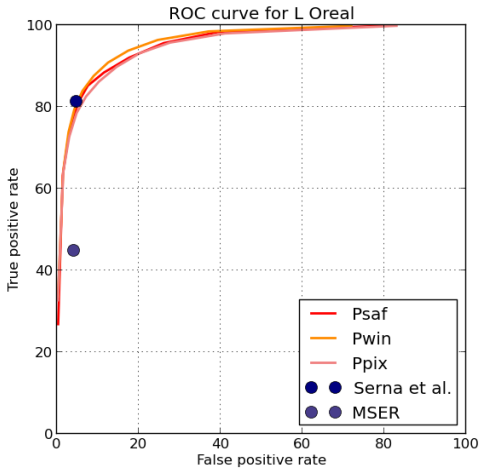
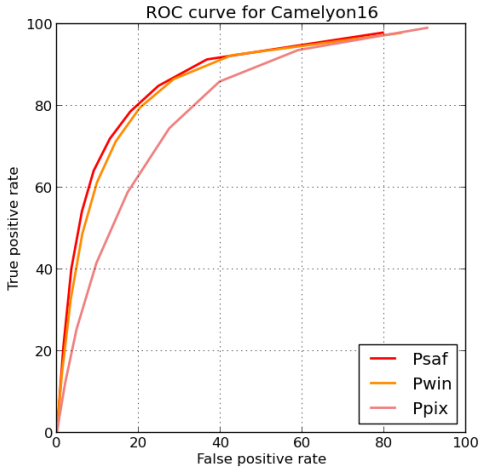


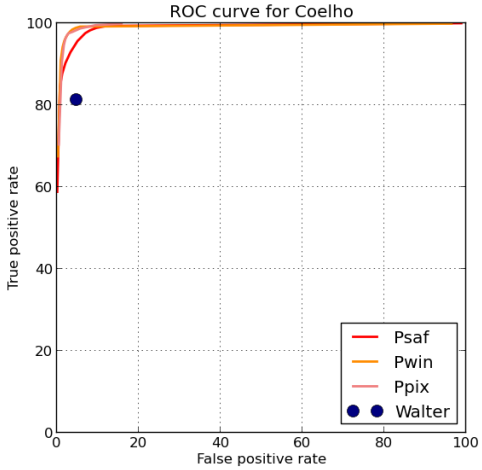
Figure 7.12: Quantitative results for $\mathscr{D}_3$.



(a) ROC curve for $\mathscr{D}_1$



(b) ROC curve for $\mathscr{D}_2$



(c) ROC curve for $\mathscr{D}_3$

Figure 7.13: ROC curves for the three databases.

7.5 Conclusion and prospects on SAF

7.5.1 Conclusion

In conclusion, we have introduced a novel general segmentation method consisting in performing pixel classification using features computed on superpixels: the **Superpixel-Adaptive Features (SAF)**. The advantage over the window approach is to use a type of computational support which adapts to the image content, capturing more accurately the properties of each class. The advantage over direct superpixel classification is the possibility to use multiple partitions (such as, but not limited to, multiple scales). Compared to pixel-CS and window-CS approaches, classification results are rather equivalent but with a much better spatial coherence of detected objects, which is desired for segmentation purposes. SAF also succeed in reaching state-of-the-art methods specifically designed for a given database, and without requiring any adaptation/tuning step from the user (apart from manually segmenting some examples for the training set), which is very promising for this general method.

7.5.2 Prospects

This work offers many prospects of improvements, but also of applications. Here are two examples:

Generalization: The SAF used for *pixel* classification could be generalized to SAF for *superpixel* classification. Note that we prefer not to build a hierarchy between the SP partitions of different scales in this application (to avoid error propagation as well as to integrate other different kinds of information rather than just different scales). Hence, the new superpixel-UC (*unit of classification*) can be overlapped by more than one superpixel in a given partition of superpixel-supports. The rules to assign a superpixel support to a superpixel-UC should hence be further investigated (ex: taking the one which overlaps the most the UC, or averaging over the ones which overlap at least 90% of the UC area, ...). The advantage would be to reduce the number of UCs (superpixels instead of pixels) while computing features on different superpixels coming from a wide range of different partitions.

Other combinations of multiple partitions: There are two ways to consider SAF. Either we see SAF as features computed on special adaptive structuring elements (V. Morard et al. [2011]), with an advantage on computation complexity. Or we can consider them as a special kind of *ensemble clustering*, a sub-field of data/image analysis which aims at combining different partitions of the same set/image to obtain a unique final segmentation. Indeed, we use different partitions of SPs as well as the partition into pixels (to define the UC). Thus, it can be seen as an ensemble clustering where one partition (the partition into pixel-UC) plays a special part in the combination. This observation leads us to many prospects as other ensemble clustering methods could be used from the abundant corresponding literature to obtain the final segmentation instead of using a classification pipeline.

8 Conclusion and Prospects

Que sera sera,
What ever will be, will be.
Doris Day, The Man Who Knew Too Much.

Résumé

Ce chapitre conclut sur les waterpixels et leur pertinence dans la chaîne de classification appliquée à la segmentation d'image. Il présente également quelques perspectives pour continuer ces travaux.

*****

This chapter is dedicated to the conclusion of the PhD and presents some prospects for future work.

8.1 Conclusion

The aim of this work was to provide a general segmentation method yielding good performance on any image database without needing to be modified each time by the user (expert or not in image analysis), as far as the latter could provide some examples already segmented by hand within the database to be segmented. Indeed, the goal was to alleviate the tediousness of whole-database manual segmentation (later used for individual inspection of objects) by automating it with a single method, in order to ease and speed up the often difficult task of image content understanding. In this PhD, we have proposed a general segmentation method based on a classification pipeline and confirmed the pertinence of inserting superpixels in such workflow.

Three main methodological contributions can also be highlighted. Firstly, we have designed a new superpixel generation method based on the watershed transformation: the **waterpixels**. These superpixels are efficient in terms of quality as well as computation time. Moreover, they are easy to handle for the user and a fast implementation is available online. Secondly, we have introduced the concept of **Superpixel-Adaptive Features (SAF)** which are features computed on superpixels to classify pixels. We have confirmed that they constituted pertinent supports, with, besides, a possibility to capture richer information thanks to multiple partitions. In practice, they also offer a

better spatial coherence of detected objects than classic pipelines. Thirdly, we have shown that the **spatial repulsion between markers** proposed in the waterpixel method could, by itself, improve the watershed segmentation performance.

The PhD topic raised many challenges. The first one was to ensure the *general behavior* of the pipeline so that it could adapt to each database without requiring from the user any knowledge in the field of image analysis. Each element/step of the proposed workflow (classification process, waterpixels, operators, SAF, Random Forest) is guaranteed to stay general or to use the same procedure to automatically adapt itself to each database. The second difficulty was to build a paradigm with *low computation time* so that the segmentation automation corresponds also to the user's needs in everyday life (biologists, etc). The waterpixel generation step shows low complexity and we have provided a fast implementation on our website. The other steps of the pipeline are not optimized yet but their computation time remains reasonable. Last but not least, evolving in the *competitive field* of superpixels, which required to outperform the performance of an increasing number of competing methods with a challenging and thorough evaluation process, constituted one of the main challenges. Our contributions have been benchmarked against state-of-the-art methods and published in the community.

The framework of the proposed general method is defined as follows. *Due to supervised learning*, it must be applied on a database (not on a single isolated image) and some examples of already labeled pixels from each class (groundtruth) must be available for the training phase. *Due to the region approach embodied by superpixels*, objects should span at least some pixels as they are to be split into different superpixels. Moreover, as far as features are concerned, objects should be differentiable from one another without contour information (*e.g.* thanks to different colors, textures, etc). It would not be well suited for objects which can only be discriminated by their contours. Besides, the groundtruth must be provided as a partition into regions (and not into contour/non-contour pixels). Finally, *due to waterpixels*, we need a good contour signal in the gradient to obtain a good adherence to object boundaries.

8.2 Prospects

In this section, we propose some prospects for waterpixels and their application to image segmentation learning.

On Waterpixels: Waterpixels offer many prospects. Firstly, the different steps of the generation method could be further improved. As previously said, the optimization of the marker selection constitutes one of the most promising leads (see Chapt. 5). Secondly, they could be used in many applications as superpixels, as regions or simply as interesting computation supports. Finally, even though we adopted a different strategy afterwards with SAF, it is interesting to note that a hierarchy of partitions can be easily built while computing the watershed (iterative fusion of regions, based on a unique partition of waterpixels), which could be advantageous in other applications.

On SAF: As far as the PhD topic is concerned, the classification performance could be further improved by enriching the feature set (using other operators, other SP partitions, etc). Going beyond, it would also be interesting to build SAF of superpixels and exploit the ensemble clustering framework they can offer (see Sec. 7.5.2).

On Machine Learning and Mathematical Morphology: This work confirms once again that Machine Learning and Mathematical Morphology are two powerful fields which could benefit from each other in order to go one step further towards efficient and automatic object recognition.

This link could be further investigated, in particular in the context of the promising deep learning framework. For example, it would be interesting to consider the replacement of window patches by waterpixels in the construction of convolutional neural networks, or the insertion of SAF features at a certain intermediary step of the network.

A

Appendices

Database	Pipeline	*n_estimators*	*min_samples_leaf*
$\mathscr{D}_1$	$\mathscr{P}_{pix}$	200	10
	$\mathscr{P}_{win}$	200	10
	$\mathscr{P}_{saf}$	100	100
$\mathscr{D}_2$	$\mathscr{P}_{pix}$	50	1
	$\mathscr{P}_{win}$	200	1
	$\mathscr{P}_{saf}$	200	10
$\mathscr{D}_3$	$\mathscr{P}_{pix}$	200	1
	$\mathscr{P}_{win}$	200	1
	$\mathscr{P}_{saf}$	50	10

Table A.1: Parameter values of random forests for the three databases and the three pipelines.

A.1 Optimization of Random Forest parameters

As explained in Chapt.2, paragraph 2.3.2, the parameters of the random forests have to be tuned for each database and each pipeline in order to provide the best classification performance as possible. This is done by cross-validation (3-folds) on the training set. Different values of the number of trees and the minimum number of data samples in a leaf for the latter to be kept while building the tree during training (respectively $n\_estimators \in \{50, 100, 200\}$ and $min\_samples\_leaf \in \{1, 10, 100\}$) are tested and the pair of values which offers the best f-score (see Chapt. 2) is chosen for final training/prediction of the database. Table A.1 presents the values used for the three databases and three pipelines.

Remark: We should perform a *nested* cross-validation for $\mathscr{D}_1$, *i.e.* a cross-validation for each of the 8 trainings done in the leave-one-image-out procedure. To simplify this process, we perform a simple cross-validation on three of the 8 trainings independently, and then average the parameter values found to use them as unique values for all the 8 trainings in the final phase.

A.2 Mean mismatch factor definition

Let $\{s_i\}_{1\leq i\leq N}$ be a set of superpixels. The centered version s_i^* of s_i is obtained by translating s_i so that its barycenter is the origin of the coordinate system.

The average shape $\widehat{s}^*$ of the $\{s_i\}$ is computed as follows. Let us first define a function S:

$$S : \left| \begin{array}{ccc} D & \longrightarrow & \mathbb{N} \\ x_p & \longmapsto & \sum_{i=1}^{N} 1_i(x_p) \end{array} \right. \tag{A.1}$$

where 1_i is the indicator function of s_i^*. Thus, the image S corresponds to the summation image of all centered superpixels. Let furthermore $\mu_A = 1/n \sum_{i=1}^{N} |s_i|$ be the average area of the considered superpixels, and let S_t be the threshold of S at level t: $S_t(x) = \{x_p \in D \,\big|\, |S(x_p)| \geq t\}$.

The average centered shape $\widehat{s}^*$ is then the set S_{t_0}, where t_0 is the maximal threshold value which enables $\widehat{s}^*$ to have an area greater than or equal to μ_A:

$$t_0 = \max\{t \,\big|\, |S_t| \geq \mu_A\} \tag{A.2}$$

$$\widehat{s}^* = S_{t_0} \tag{A.3}$$

Finally, the mean mismatch factor of superpixels $\{s_i\}_{1\leq i\leq N}$ is:

$$MF = \frac{1}{N} \sum_{i=1}^{N} mf(s_i^*, \widehat{s}^*) \tag{A.4}$$

B

Bibliography

Bibliography

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

B. Andres, U. Köthe, M. Helmstaedter, W. Denk, and F. Hamprecht. Segmentation of SBFSEM Volume Data of Neural Tissue by Hierarchical Classification. *Pattern Recognition*, D:142–152, 2008.

Bjoern Andres, Ullrich Koethe, Thorben Kroeger, Moritz Helmstaedter, Kevin L. Briggman, Winfried Denk, and Fred A. Hamprecht. 3d segmentation of sbfsem images of neuropil by a graphical model over supervoxel boundaries. *Medical Image Analysis*, 16(4):796–805, 2012.

David Ascher, Paul F. Dubois, Konrad Hinsen, James Hugunin, and Travis Oliphant. *Numerical Python*. Lawrence Livermore National Laboratory, Livermore, CA, ucrl-ma-128569 edition, 1999.

Wanda Benesova and Michal Kottman. Fast superpixel segmentation using morphological processing. *Proceedings of the International Conference of Machine Vision and Machine Learning*, (67), 2014.

Serge Beucher. Watershed, hierarchical segmentation and waterfall algorithm. In Jean Serra and Pierre Soille, editors, *Mathematical Morphology and its applications to signal processing (Proceedings ISMM'94)*, pages 69–76, Fontainebleau, France, September 1994. Kluwer Academic Publishers.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Y. Chai, V. Lempitsky, and A. Zisserman. Bicos: A bi-level co-segmentation method for image classification. In *IEEE International Conference on Computer Vision*, 2011.

Luís Pedro Coelho. Mahotas: Open source software for scriptable computer vision. *CoRR*, abs/1211.4907, 2012.

Luís Pedro Coelho, Aabid Shariff, and Robert F Murphy. Nuclear segmentation in microscope cell images: a hand-segmented dataset and comparison of algorithms. In *Biomedical Imaging: From Nano to Macro, 2009. ISBI'09. IEEE International Symposium on*, pages 518–521. IEEE, 2009.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, May 2002.

Pierre-Henri Conze, François Rousseau, Vincent Noblet, Fabrice Heitz, Riccardo Memeo, and Patrick Pessaux. *Semi-automatic Liver Tumor Segmentation in Dynamic Contrast-Enhanced CT Scans Using Random Forests and Supervoxels*, pages 212–219. Springer International Publishing, 2015.

Pierre-Henri Conze, Vincent Noblet, F. Rousseau, Fabrice Heitz, Riccardo Memeo, and Patrick Pessaux. Random forests on hierarchical multi-scale supervoxels for liver tumor segmentation in dynamic contrast-enhanced CT scans. In *13th IEEE International Symposium on Biomedical Imaging, ISBI 2016, Prague, Czech Republic, April 13-16, 2016*, pages 416–419, 2016.

Liuyun Duan and Florent Lafarge. Image partitioning into convex polygons. In *CVPR*, pages 3119–3127. IEEE Computer Society, 2015.

Paul F. Dubois, Konrad Hinsen, and James Hugunin. Numerical python. *Computers in Physics*, 10(3), May/June 1996.

Erhan Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46 (1):48–60, May 1990.

Matthieu Faessel and Michel Bilodeau. Smil: Simple morphological image library. In *Séminaire Performance et Généricité, LRDE*, 2013.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

Ezequiel Geremia, Bjoern H. Menze, and Nicholas Ayache. Spatially Adaptive Random Forest. In *2013 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1332–35, San Francisco, CA, United States, 2013. IEEE.

C. Gini. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.* Tipogr. di P. Cuppini, 1912.

Allan Hanbury. *How Do Superpixels Affect Image Segmentation?*, pages 178–186. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

Robert M. Haralick, K. Sam Shanmugam, and Its'hak Dinstein. Textural features for image classification. *IEEE Trans. Systems, Man, and Cybernetics*, 3(6):610–621, 1973.

Zhongwen Hu, Qin Zu, and Qingquan Li. Watershed superpixel. *IEEE International Conference on Image Processing*, September 2015.

P. Jaccard. Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

Pavel Kalinin and Aleksandr Sirota. A graph based approach to hierarchical image oversegmentation. *Computer Vision and Image Understanding*, 130:80–86, January 2015.

J.-K. Kamarainen. *Feature Extraction Using Gabor Filters*. PhD thesis, Lappeenranta University of Technology, 2003.

Serge Koudoro, Matthieu Faessel, and Michel Bilodeau. Morph-m: Image processing library specialized in mathematical morphology. *Image Processing On Line Journal*, 2012.

Ullrich Köthe. *Generische Programmierung für die Bildverarbeitung*. PhD thesis, University of Hamburg, 2000.

A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. TurboPixels: Fast Superpixels Using Geometric Flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, 2009.

Alex Levinshtein, Cristian Sminchisescu, and Sven Dickinson. *Optimal Contour Closure by Superpixel Grouping*, pages 480–493. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Graham Knott, and Pascal Fua. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE Trans. Med. Imaging*, 31(2):474–486, 2012.

V. Machairas, E. Decencière, and T. Walter. Waterpixels: superpixels based on the watershed transformation. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 4343–4347, October 2014.

V. Machairas, M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter, and E. Decencière. Waterpixels. *IEEE Transactions on Image Processing*, 24(11):3707–3716, November 2015. ISSN 1057-7149.

Vaïa Machairas, Thérèse Baldeweck, Thomas Walter, and Etienne Decencière. New general features based on superpixels for image segmentation learning. In *13th IEEE International Symposium on Biomedical Imaging, ISBI 2016, Prague, Czech Republic, April 13-16, 2016*, pages 1409–1413, 2016.

Marco Manfredi, Costantino Grana, and Rita Cucchiara. Learning superpixel relations for supervised image segmentation. In *2014 IEEE International Conference on Image Processing, ICIP 2014, Paris, France, October 27-30, 2014*, pages 4437–4441, 2014.

B. Marcotegui and F. Meyer. Bottom-up segmentation of image sequences for coding. *Annales Des Télécommunications*, 52(7-8):397–407, 1997. ISSN 0003-4347, 1958-9395.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Int. Conf. on Computer Vision*, volume 2, pages 416–423, July 2001.

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, pages 36.1–36.10. BMVA Press, 2002.

Fernand Meyer. Minimal spanning forests for morphological segmentation. In Jean Serra and P. Soille, editors, *Mathematical Morphology and its applications to signal processing (Proceedings ISMM'94)*, pages 13–14, Fontainebleau, France, September 1994. Kluwer Academic Publishers.

Fernand Meyer. An overview of morphological segmentation. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(7):1089–1118, 2001.

Branislav Micusik and Jana Kosecka. Semantic segmentation of street scenes by superpixel co-occurrence and 3d geometry. In *IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, Sept 2009.

O. Monga. An optimal region growing algorithm for image segmentation. *International Journal of Pattern Recognition and Artificial Intelligence*, 01(03n04):351–375, 1987.

Peer Neubert and Peter Protzel. Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms. In *2014 IEEE International Conference on Pattern Recognition (ICPR)*, pages 996–1001, August 2014.

Travis E. Oliphant. *Guide to NumPy*. Provo, UT, March 2006.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011a.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011b.

Loïc Peter, Olivier Pauly, Pierre Chatelain, Diana Mateus, and Nassir Navab. Scale-adaptive forest training via an efficient feature sampling scheme. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *MICCAI (1)*, volume 9349 of *Lecture Notes in Computer Science*, pages 637–644. Springer, 2015.

X. Ren and J. Malik. Learning a classification model for segmentation. In *International Conference on Computer Vision, 2003*, pages 10–17 vol.1, 2003.

Alexander Schick, Mika Fischer, and Rainer Stiefelhagen. An evaluation of the compactness of superpixels. *Pattern Recognition Letters*, 43(0):71 – 80, 2014. {ICPR2012} Awarded Papers.

Andrés Serna, Beatriz Marcotegui, Etienne Decencière, Thérèse Baldeweck, Ana-Maria Pena, and Sébastien Brizion. Segmentation of elongated objects using attribute profiles and area stability: Application to melanocyte segmentation in engineered skin. *Pattern Recognition Letters*, 47: 172–182, 2014.

Jianbing Shen, Yunfan Du, Wenguan Wang, and Xuelong Li. Lazy random walks for superpixel segmentation. *IEEE Trans. Image Processing*, 23(4):1451–1462, 2014.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000.

Pierre Soille. *Morphological image analysis: principles and applications*. Springer-Verlag New York, Inc., 2003.

C. Sommer, C. Strähle, U. Köthe, and F. A. Hamprecht. ilastik: Interactive learning and segmentation toolkit. *Eighth IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 230–233, 2011.

J. Stawiaski and E. Decencière. Interactive liver tumor segmentation using watershed and graph cuts. In *Segmentation in the clinic : A grand Challenge II (MICCAI 2008 workshop)*, New York, USA, 2008.

N.J.C. Strachan, P. Nesvadba, and A.R. Allen. Fish species recognition by shape analysis of images. *Pattern Recognition*, 23(5):539 – 544, 1990.

Ngan Meng Tan, Yanwu Xu, Wooi-Boon Goh, and Jiang Liu. Robust multi-scale superpixel classification for optic cup localization. *Comp. Med. Imag. and Graph.*, 40:182–193, 2015.

V. Morard, E. Decencière, and P. Dokladal. Region growing structuring elements and new operators based on their shape. In *Signal and Image Processing (SIP 2011)*, Etats-Unis, 2011. ACTA Press.

C. Vachier and Fernand Meyer. Extinction values: A new measurement of persistence. *IEEE Workshop on Non Linear Signal/Image Processing*, page 254–257, 1995.

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. ISSN 2167-8359.

O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Eur. Conf. on Computer Vision*, pages 211–224, 2010.

Jie Wang and Xiaoqiang Wang. VCells : Simple and Efficient Superpixels Using Edge-Weighted Centroidal Voronoi Tessellations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1241–1247, 2012.

W. Wu, A. Y. C. Chen, L. Zhao, and **J. J. Corso**. Brain tumor detection and segmentation in a CRF (conditional random fields) framework with pixel-pairwise affinity and superpixel-level features. *International Journal of Computer Aided Radiology and Surgery*, 9(2):241–253, 2014.

Yanwu Xu, Jiang Liu, Jun Cheng, Fengshou Yin, Ngan Meng Tan, Damon Wing Kee Wong, Ching Yu Cheng, Yih Chung Tham, and Tien Yin Wong. Efficient optic cup localization based on superpixel classification for glaucoma diagnosis in digital fundus images. In *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*, pages 49–52, 2012.

G. Zeng, P. Wang, J. Wang, R. Gan, and H. Zha. Structure-sensitive superpixels via geodesic distance. *Int. Conf. on Computer Vision*, 1(c):447–454, 2011.

Gang Zhou, Yuehu Liu, and Zhiqiang Tian. Scene text detection with superpixels and hierarchical model. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, Sept 2012.

Résumé

L'objectif de ces travaux est de fournir une méthode de segmentation sémantique qui soit générale et automatique, c'est-à-dire une méthode qui puisse s'adapter par elle-même à tout type de base d'images, afin d'être utilisée directement par les non-experts en traitement d'image, comme les biologistes.

Pour cela, nous proposons d'utiliser la classification de pixel, une approche classique d'apprentissage supervisé, où l'objectif est d'attribuer à chaque pixel l'étiquette de l'objet auquel il appartient. Les descripteurs des pixels à classer sont souvent calculés sur des supports fixes, par exemple une fenêtre centrée sur chaque pixel, ce qui conduit à des erreurs de classification, notamment au niveau des contours d'objets. Nous nous intéressons donc à un autre support, plus large que le pixel et s'adaptant au contenu de l'image : le *superpixel*.

Les superpixels sont des régions homogènes et régulières, issues d'une segmentation de bas niveau. Nous proposons une nouvelle façon de les générer grâce à la ligne de partage des eaux, les *waterpixels*, méthode rapide, performante et facile à prendre en main par l'utilisateur. Ces superpixels sont ensuite utilisés dans la chaîne de classification, soit à la place des pixels à classer, soit comme support pertinent pour calculer les descripteurs, appelés SAF (*Superpixel-Adaptive Features*). Cette seconde approche constitue une méthode générale de segmentation dont la pertinence est évaluée qualitativement et quantitativement sur trois bases d'images provenant du milieu biomédical.

Mots Clés

segmentation, morphologie mathématique, apprentissage statistique, superpixels, ligne de partage des eaux

Abstract

In this work, we would like to provide a general method for automatic semantic segmentation, which could adapt itself to any image database in order to be directly used by non-experts in image analysis (such as biologists).

To address this problem, we first propose to use pixel classification, a classic approach based on supervised learning, where the aim is to assign to each pixel the label of the object it belongs to. Features describing each pixel properties, and which are used to determine the class label, are often computed on a fixed-shape support (such as a centered window), which leads, in particular, to misclassifications on object contours. Therefore, we consider another support which is wider than the pixel itself and adapts to the image content: the *superpixel*.

Superpixels are homogeneous and rather regular regions resulting from a low-level segmentation. We propose a new superpixel generation method based on the watershed, the *waterpixels*, which are efficient, fast to compute and easy to handle by the user. They are then inserted in the classification pipeline, either in replacement of pixels to be classified, or as relevant supports to compute the features, called *Superpixel-Adaptive Features* (SAF). This second approach constitutes a general segmentation method whose pertinence is qualitatively and quantitatively evaluated on three databases from the biological field.

Keywords

segmentation, mathematical morphology, machine learning, superpixels, watershed