PhD Thesis

Automating and Assisting Image Segmentation with Decision Forests

Loïc Peter

Technische Universität München
Department of Computer Science
Chair for Computer Aided Medical Procedures and Augmented Reality

TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Automating and Assisting Image Segmentation with Decision Forests

Loïc Franz Christophe Peter

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des Akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. Julien Gagneur
Prüfer der Dissertation:	1.	Prof. Dr. Nassir Navab
	2.	Prof. Dr. Vincent Lepetit

Die Dissertation wurde am 15.02.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 01.07.2017 angenommen.

Abstract

The task of semantic image segmentation, i.e. the partitioning of an image into regions whose content is identified, is a fundamental objective in computer vision and medical image analysis. The need for segmentation algorithms can be encountered in diverse real-world scenarios. In some cases, the labeling of a scene must be entirely *automated*, for example to act as a scene understanding tool to help the decision-making of an autonomous system such as a robot or a self-driving vehicle. In other situations, a segmentation algorithm *assists* a human user in labeling an image, for instance to facilitate in a clinical context the estimation of the volume of an organ or of the area of an anatomical region. In the latter scenario, the fully-automated aspect is no longer crucial and can be relaxed in favor of an improved flexibility via interactions with the user.

In this thesis, we demonstrate how the statistical learning framework of decision forests can be exploited for these two application cases. Our first contribution is a scale-adaptive training of random forests based on Haar-like features. With a simple modification of the standard training, we show that the visual information around a pixel can be effectively learned at both local and global scales while keeping the computational time unchanged at training and prediction times. In a second contribution, we demonstrate how such an automated forest-based segmentation model can be used to guide a user through large visual data. More precisely, we consider the case of large digital slides in histopathology and introduce an interactive framework where the segmentation task is jointly performed with an exploration phase which invites the user to look at regions of interest within the high-resolution image. Online real-time corrections can be made after each region suggestion to adapt the prediction model to the specific properties of the image at hand. Going towards hands-free interactions, our third contribution considers the novel case where a human user only provides binary "*Yes/No*" inputs to guide a segmentation. This scenario is seen as an instance of the Twenty Questions game, where the machine must ask informative questions to guess the correct segmentation expected by the user. We apply this setting both for generic interactive segmentation and for the refinement of an existing forest-based prediction.

Overall, we hope that our work demonstrates the effectiveness of decision forests for automated segmentation in diverse application domains and offers with our scale-adaptive scheme a generic contribution towards their improvement. Moreover, we wish to emphasize the relevance of interactive settings and to introduce practical solutions where learned models can assist manual annotations in cases where standard interactive techniques are not directly applicable.

Kurzfassung

Semantische Bildsegmentierung ist ein grundlegendes Ziel in den Bereichen Computer Vision und medizinischer Bildanalyse. Sie besteht darin, ein Bild in Regionen aufzuteilen und ihren individuellen Inhalt gleichzeitig zu interpretieren. Segmentierungsalgorithmen werden in diversen praktischen Situationen benötigt. In manchen Fällen muss die Segmentierung eines Bildes voll *automatisiert* werden. Eine mögliche Anwendung ist das automatische Verstehen der Umgebung eines autonomen Systems, auf dem seine Entscheidungen basieren. In anderen Situationen eignet sich ein Algorithmus eher dazu, einen Menschen dabei zu *unterstützen*, ein Bild zu markieren. Dieses Szenario wird beispielsweise in einem klinischen Kontext beobachtet, in dem ein Arzt das Volumen eines Organs einschätzen möchte, was aufwendige Markierungen eines 3D-Bildes erfordert. In diesem Fall kann der voll automatisierte Aspekt des Algorithmus gegen eine höhere Flexibilität ausgetauscht werden, die durch Interaktionen mit dem Benutzer ermöglicht wird.

In dieser Dissertation zeigen wir wie das Random Forest Framework für diese beiden Anwendungen genutzt werden kann. Unser erster Beitrag ist eine neue auf Haar-Features basierende Trainingsmethode, die informative visuelle Spannbreiten automatisch und ohne Erhöhung des Rechenaufwandes findet. Als zweiten Beitrag stellen wir eine interaktive Methode vor, die die Vorhersage eines im Voraus erlernten Random Forest Modells nutzt, um die Beobachtung sehr großer histologischer Schnitte zu erleichtern und Bildregionen anzuzeigen, die klinisch relevante Objekte enthalten. Nach jedem Vorschlag kann die Relevanz dieser Auswahl interaktiv bestätigt oder widerrufen werden, um die zukünftigen Entscheidungen durch inkrementelles Lernen anzupassen. Im Bezug auf handfreie Interaktionen schlagen wir schließlich ein neues Szenario vor, in dem der Benutzer nur durch eine binäre “*Ja / Nein*” Eingabe mit der Maschine kommunizieren kann. Dabei wird Objektsegmentierung als ein “*Wer bin ich?*”-Spiel zwischen Mensch und Computer aufgefasst, in dem der Computer informative Fragen stellen muss, um die vom Benutzer erwartete Segmentierung zu erraten. Darüber hinaus zeigen wir wie dieses Szenario sich für die Verfeinerung einer existierenden Random Forest Segmentierungsvorhersage eignet.

Mit diesen Beiträgen hoffen wir, die Effektivität von Random Forests für automatisierte Segmentierung in verschiedenen Anwendungsfällen zu veranschaulichen und mit unserer neuen Trainingsmethode eine Verbesserung des klassichen Lernverfahrens zu erzielen. Zudem wollen wir die Relevanz von Mensch-Computer-Interaktion unterstreichen und praktische Lösungen einführen, in denen vorgelernte Modelle manuelle Markierungen unterstützen können, wenn klassische interaktive Methoden nicht direkt anwendbar sind.

Acknowledgments

As the PhD adventure comes to an end, I would like to thank here the numerous people who contributed directly or indirectly to this thesis and who made these years of research such an exciting time.

First and foremost, I would like to express my deep gratitude to my PhD advisor Nassir Navab for his supervision and for always being very trusting and supportive in terms of science and career decisions during and after the PhD. I remember the chair as a fantastic research environment, with an extraordinary amount of trust and freedom in pursuing our own ideas and scientific collaborations. I am very grateful to Nassir for creating such an environment and giving me the opportunity to be part of it. It was a real chance and pleasure to spend these years there.

I would like to thank Vincent Lepetit and Julien Gagneur for having accepted to be part of my PhD committee, for having taken the time to read this thesis and for the excellent comments and discussion during the defense. I am also grateful to Nikos Paragios and Willem Ouwehand for their support at the early stages of the PhD. I also would like to thank Guillaume Charras for offering me a very positive first experience of academic research, which definitely contributed to my decision to continue along this path.

I owe a lot to Olivier Pauly, Diana Mateus and Max Baust, three senior researchers at the chair with whom I had the pleasure to work in close collaboration. The three of them played an essential role in this PhD and I would like to thank them for having been there when needed, both for exchanges of scientific ideas but also for personal discussions. Olivier's methodological advices during the first year contributed greatly to my personal development as a researcher and still have nowadays a strong influence on the way I approach research. Diana acted as a mentor all along the PhD: she was not only always available and supportive at personal and scientific levels, but her trust also gave me the opportunity to be regularly involved in 'senior activities' such as teaching, peer reviewing and the organisation of a workshop which all were a great learning experience. Finally, Max's regular support and encouragement towards the end of the PhD gave me a lot of energy to complete the final steps and his advices about how to approach the next career step after the PhD were a fantastic help.

I would like to thank Nicolas for his supervision while working on the master's project which led to the PhD and for the countless meals and discussions afterwards, and Tingying, not only for challenging my beliefs about research but also for her regular interaction with clinicians which made possible the project on histology presented in this thesis. More generally, I would like to thank all the colleagues at the chair who make this place such an amazing environment. I still remember the machine learning meetings with Olivier, Pierre, Christian, Joé and Vasilis as the most constructive and exciting scientific meetings I had the chance to attend. However, discussions about science were definitely not limited to the meeting room: it was always a pleasure to join Pierre for his daily walks, to have a tea break with Benjamin and to engage in office debates with Christian and Sebastian about the goal of academic research and the right way to do a nested cross-validation. Many scientific contributions resulted from the discussions and the collaborative environment at the chair, and it was overall a real pleasure to work together with Olivier, Diana, Max, Nicolas, Pierre, Christian, Benjamin, Sailesh, Ralf, Kanishka, Marie and Florian, as well as with our clinical partners Noemi Schworm, Stefan Stangl and Gabriele Multhoff. Finally, I am really happy and proud of having managed a healthy compromise between research and physical activities such as Kicker, cricket, pétanque, karaoke, climbing, (fantasy) football, Schafkopf, board games and golf tournaments. Thanks to Pierre, Benjamin, Christian, Sebastian, Robert, Christian, Tobias, Ralf, Iro, Federico, Richard, Séverine, Felix and many others for all these great moments out of the university. In addition, I would like to thank once more Christian for his help on the day of the defense, Esther for offering me a room the night before the defense, and Benjamin for his help regarding the final steps of the PhD and for offering me to stay at his country house when I was back in Munich.

The perseverance that is required to complete a PhD is certainly not limited to the scientific part. I cannot be grateful enough to Martina Hilla, Manuela Fischer and Inga Weise for their frequent and precious help regarding all administrative issues, and most notably for the preparation of the PhD defense and the recognition of my French degree by the university. Many thanks to Martin and Dennis as well for their great help running the infrastructure at the chair.

The PhD defense took place as I had already left Munich for London. I would like to thank Tom Vercauteren and Sébastien Ourselin for welcoming me in the department at UCL and for their support before the defense. I would like to thank my new colleagues at UCL who made me feel comfortable very quickly in this new environment. Many thanks to Jonas, Michael, Maria, Kerstin, Kostas, Marta, Liane, François, João and Stefano for their friendship, for the kind messages on the day of the defense and for the celebrations following it. I also would like to address special thanks to Kerstin for taking the time to proofread the Kurzfassung on page iii.

It is impossible to conclude these acknowledgments without mentioning a few other people who, although they were not part of the immediate research environment, played nevertheless an essential role at a more personal level through their constant support and affection. I am extremely grateful to Christina for her unconditional support and patience during the most important part of the PhD, for being the best possible teacher of the German culture and language, and of course more generally for all the great times we had together. Thanks to Guillaume, Yoann and Romain with whom I shared amazing moments in Munich but also in London and in Paris. Thanks to Marthe for her precious support in Berlin in particular. And finally, thanks to my old friends Jean, Marco and JS, and to Stéphane, for his exceptional friendship (of course!) but also for always inspiring discussions and refreshing points of view about science. Finally, this thesis (and most of the preceeding steps!) would not have been possible without the constant support of my family, especially Laurine and my parents. Thanks to all of you.

Contents

Introduction

By definition, the task of image segmentation consists in parsing a given image into semantically meaningful regions by associating a label to each pixel. The segmentation objective is a fundamental building-block of image analysis algorithms, as it typically bridges the gap between low-level pixel information and high-level image content. The use of a segmentation algorithm is, for instance, relevant every time the size or the shape of a given object has to be quantitatively estimated. The general definition of segmentation encompasses in fact slightly different scenarios (Fig. 1) such as:

- Given an image, can we provide a binary segmentation separating background and foreground? Binary segmentation can, for example, be used as a first step to extract objects in a scene before further analysis or to conduct image editing tasks.
- More generally, one can desire to divide an image into regions that are considered homogeneous in some sense. This scenario is often ill-posed and several persons can have very different expectations about what constitutes a good parsing. The image parsing problem is strongly related to edge detection where one tries to define edges that are separating meaningful objects. It can be used as a pre-processing tool to reduce the complexity of an image, moving from pixels to superpixels or to objects representing higher-level information.
- The semantic image segmentation scenario is an instance of the image parsing objective where each region is assigned a semantic label indicating its content such as `Sky` or `Horse`. These labels are usually chosen among a predefined list of possibilities. With the introduction of semantic labels, the ill-defined aspect of image parsing is mitigated and the goal of understanding the content of the image is explicitly introduced.

Most segmentation tasks encountered in practice could be conducted by a human person with enough time and training, although the result may show inter-segmenter variability depending on how challenging and well-posed the task is. However, a considerable part of research in computer vision and medical image analysis consists in finding algorithms to facilitate segmentation procedures. We can distinguish two general situations where a segmentation algorithm can be of interest:

1. **Automating segmentation**, where the segmentation of an image must be performed without human intervention. A typical example would be the scenario of a self-driving vehicle, which has to automatically interpret the surrounding environment

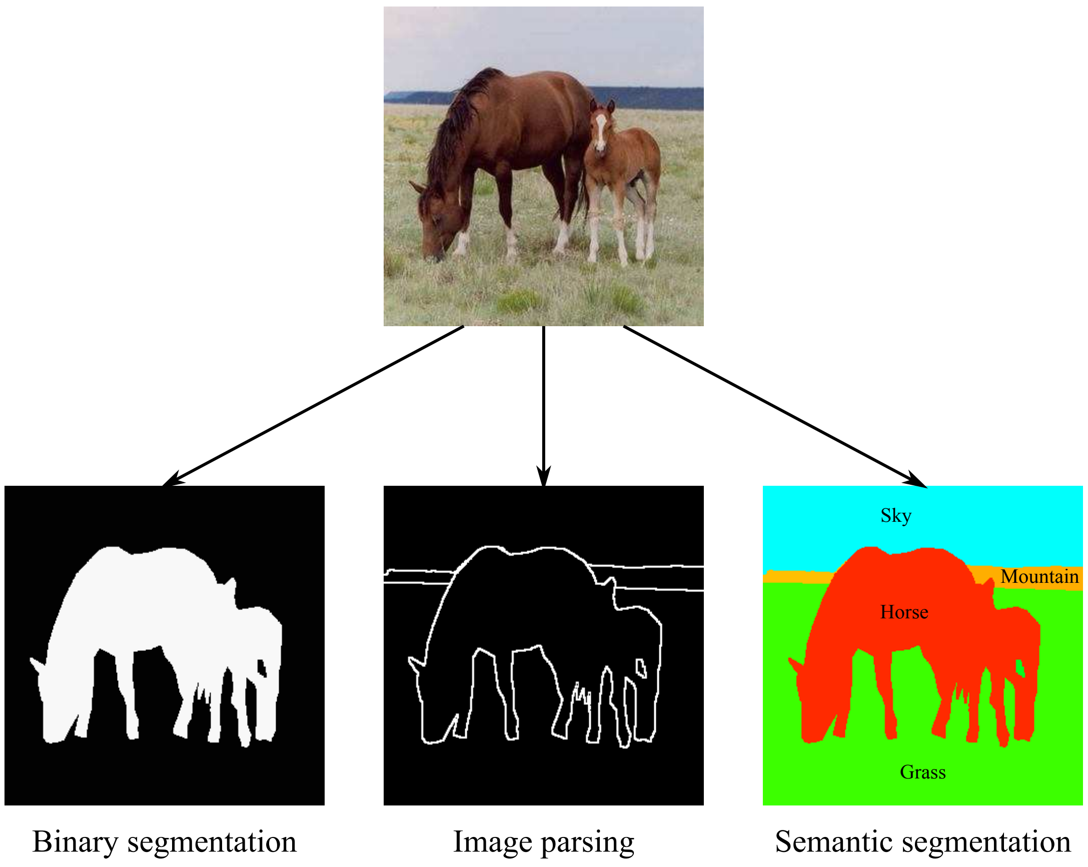


Figure 1: **Different types of image segmentation.**

to make decisions. If the sensors built in the vehicle are cameras, a semantic image segmentation of each frame would inform about the presence or absence of various kind of objects. Since the segmentation is part of an entirely automated system, it is impossible, from an application point of view, to allow a human intervention in the process. Therefore, the segmentation task must be left to an algorithm under presumably strong constraints in terms of accuracy and reliability.

2. **Assisting segmentation**, where the labeling process would take too long for the human sitting behind a computer. Here, we can imagine a medical scenario where a clinician would be interested in the volume of an organ in a 3D body scan. While the user could label manually each 2D slice, an algorithmic solution could be of great help to facilitate this otherwise tedious task. A fully-automated segmentation algorithm as mentioned above remains a natural option. However, unlike in the first scenario, we have here the additional opportunity to involve the clinician in the segmentation process, for instance to correct some mistakes of an automated approach. If the requested user intervention is lightweight enough, an interactive solution can still be very beneficial in comparison to a manual segmentation. Moreover, the clinician retains a visual control over the segmentation output, allowing to discard or refine the result if it is not satisfying enough.

This thesis proposes contributions for these two scenarios. The first two chapters focus on the automation of a semantic image segmentation objective with statistical learning methods. In Chapter 1, we expose the general framework of semantic image segmentation and describe its traditional formulation as a supervised learning method. In other words, we explain how a fully automated semantic image segmentation strategy can be learned from true examples labeled by a human. Chapter 2 focuses in details on a specific class of supervised learning models popular for segmentation in computer vision and medical applications: the framework of decision forests. After a detailed presentation of this formalism both as a generic supervised learning model and in the case of segmentation, we introduce our first contribution, i.e. a scale-adaptive forest training technique which improves segmentation accuracy without increase in computational cost. Chapter 3 operates as a transition chapter between automated and interactive segmentation approaches, where we discuss the possible limitations of supervised learning techniques and give a quick overview about classical approaches for interactive segmentation. In the last two chapters, we introduce two contributions for interactive segmentation where standard settings do not apply and where automated methods can be exploited without losing the flexibility brought by the user intervention. In Chapter 4, we demonstrate how to handle large digital slides where objects of interests are difficult to locate in the first place. Based on a random forest model, the semantic segmentation is jointly tackled with an exploration objective with relevance feedback capabilities. In Chapter 5, we consider the case of limited input where a user can only answer by *Yes* or *No* to questions asked by the computer. The segmentation is modeled as a Twenty Questions game between the human and the machine, where the most appropriate questions are asked to guess the object of interest to the user or to refine the prediction of a learned model.

Chapter 1

Semantic Image Segmentation: A Supervised Learning Task

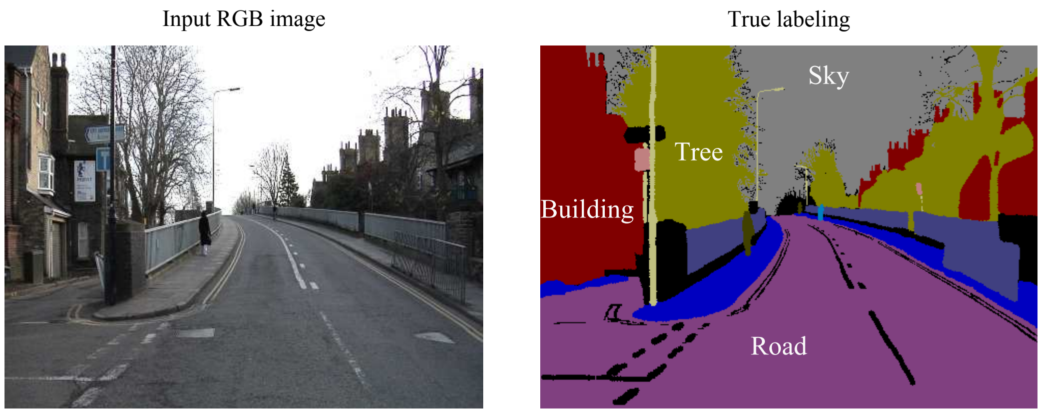
We start this thesis with an overview of the objective of semantic image segmentation and demonstrate how it can be modeled as a supervised learning task, i.e. how a segmentation algorithm can be automatically inferred from a set of labeled images. We first provide general considerations both on semantic image segmentation (Sec. 1.1) and supervised learning (Sec. 1.2). The connection between the two frameworks is then made in Sec. 1.3. This first chapter serves as an introduction to the central concepts used in this thesis and will be followed by a detailed treatment of a specific class of learning models in Chapter 2 where our first methodological contribution will be introduced.

1.1 Problem Statement

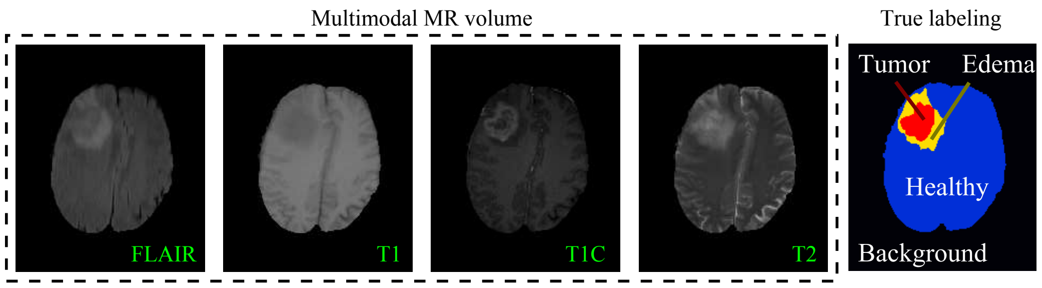
1.1.1 Images and Labelings: Definitions

Performing the semantic segmentation of an image consists in both parsing this image into regions and assigning to each region an interpretation of its content. Since there is neither a single way to parse an image nor a single way to interpret the semantic content of a region, the design of a semantic segmentation algorithm for a given application usually requires the explicit definition of a set $\mathcal{Y}$ of candidate region labels which represents the kind of objects one is interested to segment. For instance, in a road scene understanding context where images are taken by a camera mounted on a car (Fig. 1.1a), a plausible set of candidate labels could be $\mathcal{Y} = \{$`Road`, `Building`, `Sky`, `Tree`, `Sidewalk`, `Car`, `Column pole`, `Sign symbol`, `Fence`, `Pedestrian`, `Bicyclist`$\}$. One can recognize in this set of labels the most common entities appearing in road scenes and the objects that we are interested in identifying correctly, from an application of point of view, in this type of images.

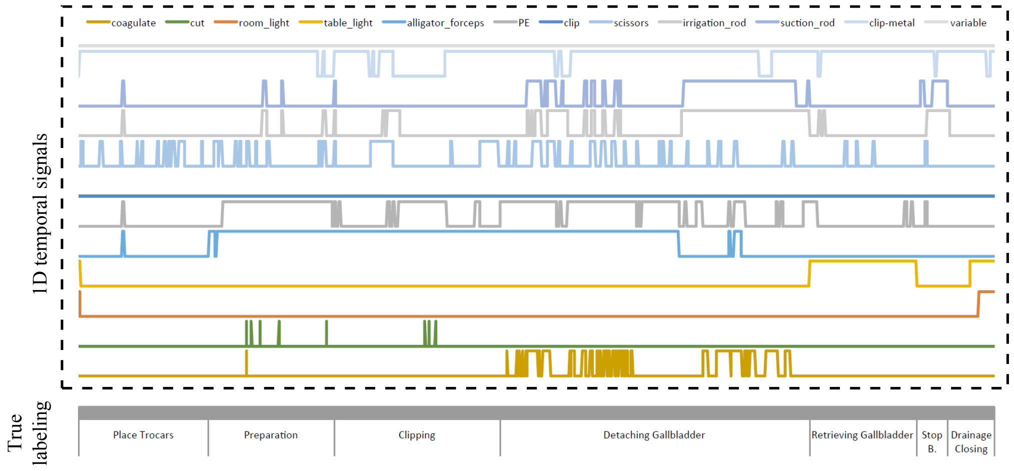
We define formally an image as a function $I : \Omega_I \to \mathbb{R}^{n_{\text{channels}}}$. Ω_I is a multidimensional grid of dimension d representing the finite set of locations where an image value is observed. In more explicit terms, Ω_I is the set of pixel locations over which I is defined. Without loss of generality, we assume Ω_I to be fully defined by d positive integers



(a) Road scene understanding (CamVid dataset [Brostow et al., 2008a]).



(b) Brain tumor segmentation in 3D MR volumes (BRATS dataset [Menze et al., 2015]).



(c) Phase recognition in surgical workflow [Stauder et al., 2014].

Figure 1.1: **Applications of semantic segmentation.** A segmentation method for road scene understanding is a building block for the development of a self-driving car, while segmenting a brain tumor can allow a quantitative measurement of its volume for clinical studies. Although the case will not be considered further in this thesis, we also mention that a temporal signal can be seen as a one-dimensional image. The recognition of the phase of a surgical workflow given instrument usage data is an example of semantic classification in this context. Further examples of applications of semantic image segmentation will be encountered in Chapter 2 and Chapter 4.

$(s_i(I))_{1\leq i\leq d} \in (\mathbb{N}^*)^d$ which encode the size of the image, so that

$$\Omega_I = \Omega\left(s_1(I), \dots, s_d(I)\right) = \prod_{i=1}^{d} \left\{1, \dots, s_i(I)\right\}, \tag{1.1}$$

where $\prod$ denotes the Cartesian product. For example, in the case $d = 3$ of a volume, $s_1(I)$, $s_2(I)$ and $s_3(I)$ are usually called height, width and depth of the image I respectively. At each location $\mathbf{p} \in \Omega_I$, the image value $I(\mathbf{p})$ is a n_{channels}-dimensional vector which corresponds to the n_{channels} signals observed at this location. For example, a color image taken with a camera has typically 3 channels, which encode the respective amounts of red, green and blue contributing to the color observed visually. Similarly, in the medical domain, a magnetic resonance volume is sometimes composed of several modalities such as `T1`, `T2` and `FLAIR` (Fig. 1.1b). For the convenience of notation, an image can also be seen as a function $I : \Omega_I \times \Gamma \to \mathbb{R}$ where Γ is the set of channels of the image, with $|\Gamma| = n_{\text{channels}}$. Given a location $\mathbf{p} \in \Omega_I$ and a color channel $c \in \Gamma$, we then denote $I(\mathbf{p}, c)$ the measured value at the location $\mathbf{p}$ in the color channel c. For a given application case (such as the ones shown in Fig. 1.1), the images to segment must share some common properties: their dimensionality d, the set Γ of color channels (or modalities) and the set of target labels $\mathcal{Y}$ are fixed. However, we do not need to assume that the images are all of the same size or acquired at the same resolution.

Given the set of candidate labels $\mathcal{Y}$ and an image I, a labeling of I is similarly defined as a function $L(I) : \Omega_I \to \mathcal{Y}$ defined over the same lattice as I and assigning to each location of Ω_I a semantic label representing the nature of the underlying object. We assume that there exists, for each image I, a unique[1] true labeling $\hat{L}(I)$. The goal of the semantic segmentation task is to predict, for a given image I, a labeling $L(I)$ as close as possible to the true segmentation $\hat{L}(I)$.

1.1.2 Assessing the Quality of a Labeling

Evaluating the performance of a semantic image segmentation algorithm requires a quantitative measure of the quality of a predicted labeling $L(I)$ given the expected true labeling $\hat{L}(I)$. The definition of a good measure may depend on the application at hand.

Binary case For simplicity, we start with a binary segmentation case, where one particular object of interest known in advance has to be segmented (for instance an organ). In this case, the set of labels is of the form $\mathcal{Y} = \{\texttt{Background}, \texttt{Foreground}\}$ where `Background` denotes the label of all pixels that do not belong to the target object (labeled as `Foreground`). In these conditions and once a prediction has been made, all pixels $\mathbf{p}$ can be partitioned into exactly 4 categories depending on the respective value of $L(I, \mathbf{p})$ and of $\hat{L}(I, \mathbf{p})$. We define accordingly the number of True Positives (TP), False Positives (FP), False Negatives (FN) and True Negatives (TN) as:

$$\text{TP} = \left|\left\{\mathbf{p} \in \Omega_I | L(I, \mathbf{p}) = \texttt{Foreground} \text{ and } \hat{L}(I, \mathbf{p}) = \texttt{Foreground}\right\}\right|, \tag{1.2}$$

[1]To avoid ambiguities, this assumption may require to extend $\mathcal{Y}$ with one additional label `Other` assigned to each object not belonging to the labels of interest.

$$\text{FP} = \left|\left\{\mathbf{p} \in \Omega_I | L(I,\mathbf{p}) = \texttt{Foreground} \text{ and } \hat{L}(I,\mathbf{p}) = \texttt{Background}\right\}\right|, \tag{1.3}$$

$$\text{FN} = \left|\left\{\mathbf{p} \in \Omega_I | L(I,\mathbf{p}) = \texttt{Background} \text{ and } \hat{L}(I,\mathbf{p}) = \texttt{Foreground}\right\}\right|, \tag{1.4}$$

$$\text{TN} = \left|\left\{\mathbf{p} \in \Omega_I | L(I,\mathbf{p}) = \texttt{Background} \text{ and } \hat{L}(I,\mathbf{p}) = \texttt{Background}\right\}\right|. \tag{1.5}$$

A first intuitive metric is the accuracy, which quantifies the proportion of pixels in the image I that have been correctly predicted, i.e.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{|\Omega_I|} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}. \tag{1.6}$$

However, if we are in an asymmetric situation where the `Foreground` label is considered more important, the accuracy is not an appropriate measure. This can be easily explained by a possible overwhelming number of true negatives masking the real relevance of the obtained segmentation. Intuitively, if the aim is for instance to segment the liver in a body scan, we do not want the quality measure of the segmentation to be influenced by the amount of pixels classified correctly in the rest of the body or, even worse, the amount of pixels outside the body (which can be made arbitrarily large). Therefore, overlap measures between the predicted foreground segmentation and the true foreground segmentation are a common choice, such as the Dice score

$$\text{Dice} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{1.7}$$

or the Jaccard index

$$\text{Jaccard} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}. \tag{1.8}$$

These quantities are between 0 and 1, respectively corresponding to an empty overlap and to a complete overlap. As recommended, they also do not depend on the number of true negatives TN. In the case of non-empty and non-perfect overlap, these global measures do not tell whether the prediction was biased towards `Foreground` (too many pixels were predicted as belonging to the object of interest, i.e. too many false positives) or biased towards `Background`, with a tendency to miss parts of the object of interest (i.e too many false negatives). To quantify these two aspects respectively, the Precision and the Recall are defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1.9}$$

and

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{1.10}$$

It can be seen that the Dice score, sometimes also called F-Score, is the harmonic mean of the Precision and the Recall, i.e.

$$\frac{1}{\text{Dice}} = \frac{1}{2}\left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}\right). \tag{1.11}$$

As such, the Dice score is always comprised between the Precision and the Recall, but biased towards the lower value of the two. This is thus a more pessimistic estimate than the arithmetic mean. Although we will not encounter them in this thesis, we finally mention that in the medical domain, boundary-based measures such as the Hausdorff distance can also be used to quantify the distance between the two segmentation contours. These metrics put a higher importance on the smoothness of the segmentation output.

Multi-Class Case In a multi-class scenario, where $|\mathcal{Y}| > 2$, the Accuracy measure is still applicable as the proportion of pixels that have been classified correctly. Thus, each pixel has an equal contribution in the computation of the Accuracy. Sometimes, we would rather give an equal importance to each label instead (see Sec. 2.1.7) and employ classwise averages of the metrics defined in the binary case. More precisely, for each label $c \in \mathcal{Y}$, we can define TP$_c$, FP$_c$ and FN$_c$ in a one-vs-all fashion, i.e. by considering temporarily all other labels as `Background` and the label c as `Foreground`. This allows us to define in a classwise manner all measures defined for the binary case, which can then be averaged over classes to provide a final quality measure.

1.2 The Supervised Learning Framework

We would like to create, for a given application, an algorithm able to predict an accurate semantic labeling $L(I)$ of an image I. A first solution would be to handcraft decision rules based on human knowledge and to implement manually a series of decisions to label each pixel. Back to the example of road scene understanding, we could explicitly define a range of color intensity which corresponds to the color of the sky commonly observed. However, the manual definition of these rules would be very tedious and limited to a given application. Instead, we can create a dataset of labeled images, i.e. of couples $\big(I_1, \hat{L}(I_1)\big), \ldots, \big(I_N, \hat{L}(I_N)\big)$. From this set of labeled images for which the 'correct answer' is known, we implement an algorithm which *learns* the relationship between the visual appearance of an image I and its correct semantic labeling $\hat{L}(I)$. In a sense, we thereby automate the design of a handcrafted algorithm by exploiting the labeled set of data, which allows to replace the complex task of designing an algorithm for a given application by the easier task of creating labeled data. The transposition of a learning algorithm from an application to another is then made easier: although there are possibilities to encode domain knowledge, most of the learning task is expected to be similar for a new application.

The field of supervised learning concerns the general objective of learning a predictive model from observations for which the correct label is known. The learned model is then expected to be deployed on new observations and to perform predictions as close as possible from the truth. This formalism is at the core of the semantic image labeling problem, among many other applications going well beyond image analysis. This section aims at introducing the general notions of supervised learning which will be encountered in the rest of this thesis in the context of semantic segmentation.

1.2.1 An Introductory Example

Before exposing a mathematical formulation of supervised learning, we start with an example to illustrate informally many of the problems and tradeoffs encountered when a problem has to be solved with machine learning. Let us consider the scenario of spam detection. How can we define an algorithm which, from the content of an e-mail, automatically decides whether it is a spam? When defined as a supervised learning task, we assume available a set of observations (e-mails) for which the label (`spam` or `no spam`) has been communicated by a human user. We then use this acquired knowledge to generate a decision rule which minimizes the expected error rate on future unseen data. We will expose this formally in Sec. 1.2.2. A standard application of supervised learning involves two steps. First, we have to find a way to create a clear quantitative representation of an e-mail. An e-mail is indeed fundamentally a string of characters, which is a very unstructured entity for a mathematical treatment. A so-called *feature extraction* step is thus needed, which transforms an e-mail into a (possibly high-dimensional) vector which encodes quantitatively the information used for the prediction task (Sec. 1.2.3). Once this conversion has been made, a *learning method* must be chosen which, from statistical observations over the set of training vectors and their attached label, derives a prediction rule (Sec. 1.2.4).

When choosing a learning model, the allowed complexity of the model must be carefully adjusted to the complexity of the task. The drawback of too simplistic models is intuitively clear: it is for instance not enough to use the presence of one given word only, such as 'lottery', to identify accurately a spam. While this spam detector would probably be successful in some cases, many non lottery-related spams would be missed, and perhaps some true e-mails would be misclassified as well. However, a model that is too complex is also prone to failures, in the sense that it would be made too dependent on the available training data, finding arbitrary rules that work well on the given training data. The difficulty of picking a model with the right complexity is a fundamental problem in supervised learning known as the bias-variance tradeoff (Sec. 1.2.5), which can be mitigated by holding out data for model validation or by combining several models (Sec. 1.2.7).

In the rest of this section, we propose a more formal presentation of the intuitions we have just exposed in the case of the spam detector.

1.2.2 Mathematical formulation

The general mathematical formalism of supervised learning assumes that we are given:

- An observation space $\mathcal{X}$ denoting the set of all possible observations.

- A label space $\mathcal{Y}$ which is the set of possible predictions for an observation. Two common cases are classification where $\mathcal{Y}$ is a finite set and regression where $\mathcal{Y} = \mathbb{R}^m$. In this work, we will focus largely on classification which is closer to the nature of the semantic segmentation task.

- A joint random variable (X, Y) taking values in $\mathcal{X} \times \mathcal{Y}$ which encodes the sampling process. Any new observation is a realization of the random variable (X, Y) sampled from its distribution $P_{(X,Y)}$.

- A loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ which computes the cost $l(y, \hat{y})$ of predicting a label y if the true label is $\hat{y}$. Typically, the loss satisfies $l(\hat{y}, \hat{y}) = 0$ and $l(y, \hat{y})$ becomes greater the 'further' y is from $\hat{y}$.

- A training set $\mathcal{S} = \{(x_i, y_i), 1 \leq i \leq N\}$ of observations sampled independently from the distribution $P_{(X,Y)}$.

In a supervised learning scenario, our objective is to use the knowledge present in the training set $\mathcal{S}$ to learn a decision rule $f : \mathcal{X} \to \mathcal{Y}$ which minimizes the error on future observations, i.e minimizes the expected prediction error

$$\text{EPE}(f) = \mathrm{E}\left[l(f(X), Y)\right] = \int_{\mathcal{X}\times\mathcal{Y}} l(f(x), y) P_{(X,Y)}(x, y)\mathrm{d}x\mathrm{d}y. \tag{1.12}$$

The minimization is performed over a set of hypotheses $\mathcal{H}$ which encodes the shape of possible functions f allowed in the chosen training model. Learning a function f which outputs a label given an observation can be modeled in a probabilistic sense by learning P such that

$$f(x) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\, P(Y = y|X = x) \tag{1.13}$$

for any observation $x \in \mathcal{X}$. This approach is said to be discriminative: the decision process is made dependent on the observed realization of the random variable X without consideration about how likely this event was. While discriminative models are sufficient from a prediction point of view, a more general and complex approach is called generative and consists in learning directly the joint law $P_{(X,Y)}$, which amounts to acquiring both knowledge on the prediction task and on the underlying mechanism that generates observations. Unlike the discriminative case, the generative case also learns the probability P_X according to which observations are sampled. Knowing the joint distribution allows to estimate the expectation in Eq. 1.12 and does not remove any discriminative capabilities: if the joint distribution is known, the probability in the optimization problem shown in Eq. 1.13 can be written[2]

$$P(Y = y|X = x) = \frac{P(X = x, Y = y)}{\sum_{t\in\mathcal{Y}} P(X = x, Y = t)}. \tag{1.14}$$

Unfortunately, learning the joint distribution is often a complex task, especially for high-dimensional set of observations. In this thesis, the developed learning approaches based on decision forests will be discriminative. However, Chapter 5 will also address, in another context, the topic of density estimation over a space of segmentations.

[2]The sum in Eq. 1.14 can be replaced by an integral if $\mathcal{Y}$ is continuous, in which case P is a probability density function instead.

1.2.3 Feature Extraction

A learning model is a strategy to generate a prediction rule from labeled examples. In Sec. 1.2.2, we have not made any assumption about the nature of the space of observations $\mathcal{X}$. In fact, most learning procedures assume that the observation space has a mathematically convenient structure, for instance Euclidean. However, this assumption is not straightforward in practice, as the collected observations in their raw form are usually more abstract. As mentioned in Sec. 1.2.1, e-mails are observations for the task of spam classification but they do not come readily as a vector of predefined fixed size. For this reason, the practical deployment of a supervised learning algorithm implies first a transformation of the abstract observation space $\mathcal{X}$ into a more convenient space in which the learning and predictions will be performed. This stage is called feature extraction. In this thesis, we propose to formalize this aspect by keeping the observation set $\mathcal{X}$ abstract and considering a collection of feature maps $(\phi_{\boldsymbol{\lambda}})_{\boldsymbol{\lambda}\in\Lambda}$, where each $\phi_{\boldsymbol{\lambda}} : \mathcal{X} \to \mathbb{R}$ quantifies a certain kind of property of an observation. For instance, back to the example of the e-mail classification task, we could define Λ as the set of words in the English language, and each $\phi_{\boldsymbol{\lambda}}(x)$ would represent the number of times the word $\boldsymbol{\lambda}$ appears in the e-mail x. Prediction models are only able to access information about x via these feature maps. If Λ is finite, this formulation consists in learning in an Euclidean space, seeing each observation x as a vector $(\phi_{\boldsymbol{\lambda}}(x))_{\boldsymbol{\lambda}\in\Lambda}$.

1.2.4 First Examples of Prediction Models

After having equipped the observation space with suitable feature maps for the application at hand, the goal of the learning step is to find an effective way to combine these feature maps to perform accurate predictions. We review in this section some examples of learning models.

Linear Models

We first assume that the number $|\Lambda|$ of features is finite. Model-based learning algorithms consist in defining explicitly a hypothesis space $\mathcal{H}$ independent of the training data. If the label space is $\mathcal{Y} = \mathbb{R}$, an example of model-based algorithm is linear regression, where one seeks a function f of the form

$$f_{\mathbf{w}}(x) = \sum_{\boldsymbol{\lambda}\in\Lambda} w_{\boldsymbol{\lambda}}\phi_{\boldsymbol{\lambda}}(x), \tag{1.15}$$

where $\mathbf{w} = (w_{\boldsymbol{\lambda}})_{\boldsymbol{\lambda}\in\Lambda}$ is a vector of weights. Given a training set $\mathcal{S}$ which provides limited information about the underlying distribution of observations and labels, the goal of the training procedure is to find an appropriate vector $\mathbf{w}$ which eventually leads to an expected prediction error as small as possible according to a predefined loss l. To do so, we can for example see the samples contained in $\mathcal{S}$ as an approximation of the true distribution and minimize the error on the training set (also called empirical risk), i.e.

$$\mathrm{ER}(f, \mathcal{S}) = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{N} l(f(x_i), y_i). \tag{1.16}$$

In the case of a linear regression model with square loss defined as $L(y, y') = (y - y')^2$, this can even be solved in closed form.

In the case of classification (i.e. of a finite label space $\mathcal{Y}$), a related model is called logistic regression. For simplicity, we consider the case $\mathcal{Y} = \{0, 1\}$ of two labels only, although multi-class extensions exist. The decision function is, this time, of the form $f_{\mathbf{w}}(x) = \operatorname{argmax}_{y \in \mathcal{Y}} P(Y = y|X = x; \mathbf{w})$ where

$$P(Y = 1|X = x; \mathbf{w}) = \sigma\left(\sum_{\boldsymbol{\lambda} \in \Lambda} w_{\boldsymbol{\lambda}} \phi_{\boldsymbol{\lambda}}(x)\right). \tag{1.17}$$

$\sigma : \mathbb{R} \to]0, 1[$ is, here, the sigmoid function defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{1.18}$$

$P(Y = 0|X = x; \mathbf{w})$ is naturally defined as $P(Y = 0|X = x; \mathbf{w}) = 1 - P(Y = 1|X = x; \mathbf{w})$. In other words, a logistic regression uses an underlying linear model passed into a sigmoid function and sees this output as a probabilistic estimate of the label to be 1. As in the linear regression case, the training set $\mathcal{S}$ is used to find the model parameters $\mathbf{w}$ minimizing the error on $\mathcal{S}$, expecting a good generalizability to new data. In this case, the lack of closed form requires to resort to numerical optimization techniques such as the Newton-Raphson algorithm. Model-based approaches are especially appropriate if there is a known (here, linear) relationship between observation and labels. Thereby, the interpretation of their decision rule may also be facilitated since the contribution of each feature is directly encoded by the corresponding weight.

The k-nearest neighbors algorithm

The learning methods described above are said model-based, where a function is entirely defined by a set of parameters and where the training set is used to find appropriate values of these parameters. We present here a more flexible learning algorithm which is said instance-based: the k-nearest neighbors algorithm. First, a distance between observations must be defined. If the set of feature maps Λ is finite, the simplest example of distance is the Euclidean one defined as

$$d(x, x') = \sqrt{\sum_{\boldsymbol{\lambda} \in \Lambda} \left(\phi_{\boldsymbol{\lambda}}(x) - \phi_{\boldsymbol{\lambda}}(x')\right)^2}. \tag{1.19}$$

Given any $x \in \mathcal{X}$, we define the subset $\mathcal{S}^k_{\text{neighbors}}(x)$ of $\mathcal{S}$ as the set of the k training examples that are the closest to x. The prediction of a k-nearest neighbors model is then obtained by averaging the labels of these k neighbors. In classification, we obtain $P(Y = y|X = x)$ as the empirical class distribution within $\mathcal{S}^k_{\text{neighbors}}(x)$. In the case of regression, one performs the output average by also weighting neighbors inversely proportionally to their distance, giving more weight to close examples.

As an instance-based approach, the k-nearest neighbors algorithm has a different philosophy than model-based techniques. Instead of using the training set to find model

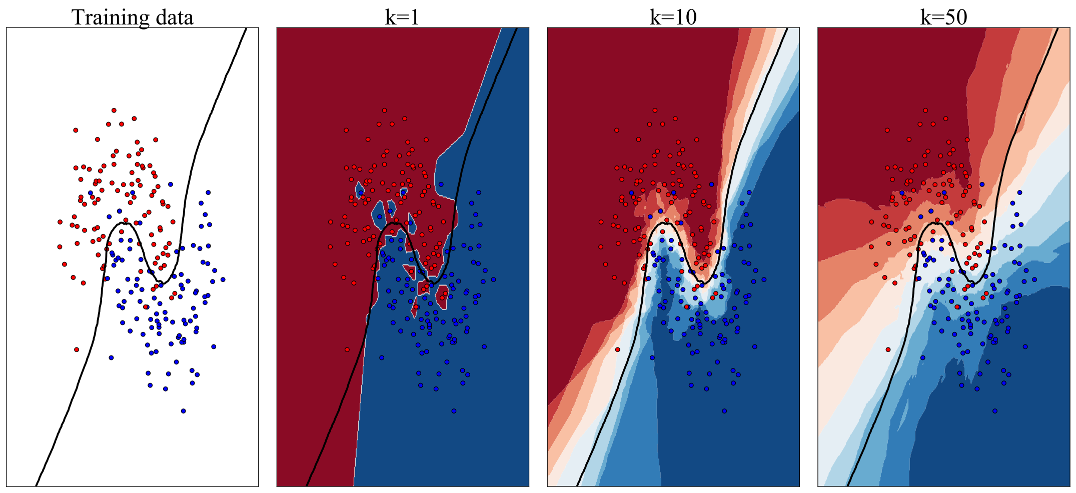


Figure 1.2: **Behavior of a k-nearest neighbor classifier when k varies.** The black line denotes the optimal decision boundary between red and blue samples. The intensity of the background color denotes the confidence in the prediction. A small value ($k = 1$) results in overfitting with highly confident predictions around outlier samples. Increasing k results in smoother predictions until the model is no longer complex enough to fit the underlying distribution accurately. $k = 10$ seems, here, to achieve a good tradeoff. These synthetic examples were created with the scikit-learn library [Pedregosa et al., 2011].

parameters, it is in a sense used as a look-up table: given a new observation, we retrieve the most similar training examples and examine their labels to form the prediction. It is therefore very flexible and can accommodate arbitrarily complex distributions. However, its accuracy relies heavily on the relevance of the chosen distance. Another drawback of the k-nearest neighbors algorithm is the fact that the training set has to be available at prediction time. Decision trees and random forests, the central learning algorithms used in this thesis, are another example of instance-based learning techniques which overcome these limitations (Chapter 2).

As a learning procedure itself, i.e. as soon as one assumes that the feature extraction and definition of the associated distance are done, the prediction rule given by the k-nearest neighbor algorithm still depends on the hyperparameter k. We illustrate in Fig. 1.2 the influence of the value of k on the prediction. With $k = 1$, one has an extremely flexible model leading to fine-grained decisions and which offers, by definition, perfect predictions on the training data itself. However, this fidelity to the training data is in fact counter-productive: every training point, even the noisy ones, are contributing to the decision, which in return deviates from the true distribution. Increasing k smoothes predictions which become more robust, until they become 'too smooth' when k is too high, no longer matching the complexity of the underlying distribution. The value of k must be carefully adjusted to realize a tradeoff between these two phenomena called respectively overfitting and underfitting: this is the bias-variance tradeoff.

1.2.5 Bias-Variance Tradeoff

Considering the regression case $\mathcal{Y} = \mathbb{R}$ using the square loss, we can bring a formal insight on the intuitive behavior observed above with the k-nearest neighbor predictions. Let us assume that the observation variable X and output variable Y are linked by $Y = \hat{f}(X) + \epsilon$ where ϵ is a random variable modeling the observed noise, which we will assume centered on 0, of standard deviation σ and independent of the observation variable X. A learning algorithm approximates $\hat{f}$ with a function f created from a set of labeled observations $\mathcal{S}$ independently drawn from the joint probability $P_{(X,Y)}$. In this section, we make explicit the dependency of f on $\mathcal{S}$ and write $f(x, \mathcal{S})$ instead of $f(x)$. If we write the expected prediction error (Eq. 1.12), we obtain [Hastie et al., 2009]

$$\begin{aligned}
\text{EPE}(f) &= \mathrm{E}_{(X,\epsilon)}\,\mathrm{E}_{\mathcal{S}}\left[(Y - f(X,\mathcal{S}))^2\right] \\
&= \mathrm{E}_{(X,\epsilon)}\left[Y^2 + \mathrm{E}_{\mathcal{S}}\left[f^2(X,\mathcal{S})\right] - 2\hat{f}(X)\,\mathrm{E}_{\mathcal{S}}\left[f(X,\mathcal{S})\right]\right] - 2\,\mathrm{E}_{\mathcal{S}}\underbrace{\mathrm{E}_{(X,\epsilon)}\left[\epsilon f(X,\mathcal{S})\right]}_{=\mathrm{E}[\epsilon]\,\mathrm{E}_X[f(X,\mathcal{S})]=0} \\
&= \sigma^2 + \mathrm{E}_X\left[\mathrm{Var}_{\mathcal{S}}\left[f(X,\mathcal{S})\right] + \mathrm{E}^2_{\mathcal{S}}\left[f(X,\mathcal{S})\right] + \hat{f}^2(X) - 2\hat{f}(X)\,\mathrm{E}_{\mathcal{S}}\left[f(X,\mathcal{S})\right]\right],
\end{aligned} \tag{1.20}$$

where we used the fact that $\mathrm{E}_{(X,\epsilon)}\left[Y^2\right] = \sigma^2 + \mathrm{E}_X\left[\hat{f}^2(X)\right]$ by independence of $\hat{f}(X)$ and ϵ, and the definition of the variance $\mathrm{Var}_{\mathcal{S}}\left[f(X,\mathcal{S})\right] = \mathrm{E}_{\mathcal{S}}\left[f^2(X,\mathcal{S})\right] - \mathrm{E}^2_{\mathcal{S}}\left[f(X,\mathcal{S})\right]$. Finally, noting that the last three terms form the squared expectation $\mathrm{E}^2_{\mathcal{S}}\left[f(X,\mathcal{S}) - \hat{f}(X)\right]$, we obtain what is called the bias-variance decomposition:

$$\text{EPE}(f) = \sigma^2 + \mathrm{E}_X\left[\mathrm{Var}_{\mathcal{S}}\left[f(X,\mathcal{S})\right] + \mathrm{E}^2_{\mathcal{S}}\left[f(X,\mathcal{S}) - \hat{f}(X)\right]\right]. \tag{1.21}$$

If we now consider a fixed observation $x \in \mathcal{X}$, we see that the expected prediction error at x of a model learned on a randomly drawn training set $\mathcal{S}$ can be decomposed into three terms:

- The irreducible error σ^2, which comes from the fact that the noise is an independent variable over which there is no control.
- The variance of the predictor $\mathrm{Var}_{\mathcal{S}}\left[f(x,\mathcal{S})\right]$ at the observation x. This term states how sensitive the predictor f is to the set of training samples on which it is learned. As shown in Sec. 1.2.4, a 1-nearest neighbor algorithm is a good example of high-variance algorithm.
- The squared bias $\mathrm{E}^2_{\mathcal{S}}\left[f(x,\mathcal{S}) - \hat{f}(x)\right] = \left(\mathrm{E}_{\mathcal{S}}\left[f(x,\mathcal{S})\right] - \hat{f}(x)\right)^2$. This term quantifies how well, when averaged over all possible training sets, the model can approximate the true function $\hat{f}$, i.e. tells whether the complexity of the learner is sufficient to approximate the underlying distribution. For instance, a linear model does not have the ability to approximate a non-linear function and will thus have some bias.

Increasing the complexity of a classifier typically reduces the bias: when repeated over several training sets, the training procedure leads in average to a more accurate model. However, a higher complexity usually also increases the sensitivity of the classifier to the particular training data it has been trained on, and thus its variance, leading to overfitting.

1.2.6 Practical Evaluation of a Learning Method

The quantitative estimation of the future performance of a trained model is crucial as soon as the model must be deployed in practice. As discussed above, the accuracy on the training data does not, in general, provide any indication about the quality of the learned model. In fact, having perfect predictions on the training set only shows that the model has found during training a way to separate the training examples given the available features, without any guarantee on the quality of the prediction on new data[3]. Given a labeled set $\mathcal{S}$ available for a given application, the generalization abilities of a learning algorithm are often empirically estimated by splitting $\mathcal{S}$ in two parts, where one part is used for training and the other one is used to measure the accuracy of the trained model. To obtain even more accurate estimates, a K-fold cross-validation can be performed. The set $\mathcal{S}$ is then split into K parts $S_1, \dots, S_K$ of comparable sizes. The k^{th} round of the cross-validation procedure uses the set $\cup_{i \neq k} \mathcal{S}_i$ for training and tests the learned model on $\mathcal{S}_k$. After performing such a round for every $k \in \{1, \dots, K\}$, every sample of $\mathcal{S}$ has received exactly one prediction so that the accuracy of the learning method can be evaluated on the entire dataset $\mathcal{S}$.

In the aforementioned cross-validation setting, hyperparameters must be defined beforehand. Choosing hyperparameters to optimize the cross-validation results is, strictly speaking, another form of overfitting which can lead to overly optimistic estimates. To include hyperparameter optimization in the learning stage, the dataset $\mathcal{S}$ can be split into three parts respectively called training, validation and test set. Models are learned on the training set and hyperparameters are chosen to optimize the performance on the validation set. Finally, the prediction on the still unseen test data is reported. To use the full data, a nested cross-validation can be performed as follows. The set $\mathcal{S}$ is divided into K parts and, as described above, the k^{th} round leaves the set $\mathcal{S}_k$ out to compute the test accuracy. With the remaining $K - 1$ subsets, we perform another cross-validation and choose the hyperparameters maximizing the resulting accuracy. Retraining a model with these hyperparameters, the final accuracy can be reported on the left out set $\mathcal{S}_k$. In practice, nested cross-validations are rarely employed in the computer vision and medical image analysis communities, probably because of their large computational cost. To allow a fair and overfitting-free comparison between algorithms on a given public database, a separate test set is often kept hidden and the performance evaluation on this set can only be run a

[3]Consider the one-dimensional feature space where the only feature map ϕ_1 outputs for every observation a uniformly drawn random number in $[0, 1]$ independent of the observation and of its label. The features $\phi_1(x_1), \dots, \phi_1(x_N)$ of the N training instances $x_1, \dots, x_N$ are almost surely all different, so that a 1-nearest neighbor learner will predict perfectly the label of each training instance. However, the accuracy of this learned model on new data is by definition not better than random since the available feature map does not carry any information about the label of an instance.

limited number of times (see for example Russakovsky et al. [2015]).

1.2.7 Ensemble Learning

As we discussed in Sec. 1.2.4 and Sec. 1.2.5, the complexity of the learning model must be carefully chosen to realize a tradeoff between bias and variance. If it is scalable for a given application, the hyperparameters encoding the complexity of the classifier (such as k in a k-nearest neighbors model) can be optimized on a separate validation set as explained in Sec. 1.2.6. In this section, we expose other strategies to adjust the complexity of a model based on the concept of ensemble learning. Instead of performing predictions with a single learned function f, the prediction model is built by aggregating several predictors. Depending on the chosen aggregation strategy, an ensemble learner can either have a lower variance or a lower bias than its individual building blocks.

Reducing Variance: Bagging

A model displays a high variance if it is sensitive to the particular set of training samples that it is trained on. In other words, if $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$ are two training sets both made of N i.i.d realizations of the joint random variable (X, Y), the predictions $f(x, \mathcal{S}^{(1)})$ and $f(x, \mathcal{S}^{(2)})$ at an observation $x \in \mathcal{X}$ can be different, with the notations of Sec. 1.2.5. The idea of bootstrap aggregating (abbreviated bagging) consists in averaging predictors trained on different training sets of the same size [Breiman, 1996], leading to an ensemble of lower variance than the individual learners. Formally, the aggregated predictor f_A is defined as

$$f_A(x) = \mathrm{E}_{\mathcal{S}}\left[f(x, \mathcal{S})\right], \tag{1.22}$$

i.e. the expected model prediction for an observation $x \in \mathcal{X}$ when the training set $\mathcal{S}$ is seen itself as a random variable. As was done for the bias-variance decomposition, the theoretical benefit of considering f_A can be seen on the example case of a regression with square loss. Considering the expected prediction error EPE(f) $=$ $\mathrm{E}_{\mathcal{S}}\,\mathrm{E}_{(X,\epsilon)}\left[(Y - f(X, \mathcal{S}))^2\right]$, we observe [Breiman, 1996] that

$$\begin{aligned}
\text{EPE}(f) &= \mathrm{E}_{(X,\epsilon)}\left[Y^2\right] - 2\,\mathrm{E}_{(X,\epsilon)}[Y \underbrace{\mathrm{E}_{\mathcal{S}}\left[f(X, \mathcal{S})\right]}_{=f_A(x)}] + \mathrm{E}_X\left[\mathrm{E}_{\mathcal{S}}\left[f^2(X, \mathcal{S})\right]\right] \\
&= \text{EPE}(f_A) + \mathrm{E}_X\left[\mathrm{E}_{\mathcal{S}}[f^2(X, \mathcal{S})] - \mathrm{E}^2_{\mathcal{S}}\left[f(X, \mathcal{S})\right]\right] \\
&= \text{EPE}(f_A) + \mathrm{E}_X\underbrace{[\mathrm{Var}_{\mathcal{S}}\left[f(X, \mathcal{S})\right]]}_{\geq 0} \\
&\geq \text{EPE}(f_A).
\end{aligned} \tag{1.23}$$

Therefore, the expected prediction error is always lower and the gained performance is directly related to the variance of the prediction when the training set varies, i.e. the instability of the learner. The construction of f_A assumes the feasibility of computing an expectation over all the possible training sets. However, only one fixed set of annotated data $\mathcal{S}$ is usually available in practice. The bagged predictor built on a training set $\mathcal{S}$ is then defined by sampling new training sets with replacement within $\mathcal{S}$, training a model on each of these subsets and averaging their predictions at testing time. The concept of bagging will be used in the random forest framework as a way to reduce the high variance of decision tree learners (Sec. 2.2.1).

Reducing Bias: Boosting

The bagging approach we have just described combines learners which are 'too complex' for a given task to create a global predictor which is less sensitive to the data it is trained on, thereby reducing overfitting. We mention here another ensemble learning technique called boosting which, this time, combines classifiers that are not complex enough to create an ensemble of better performance. Assuming an hypothesis space $\mathcal{H}$ made of classifiers of low bias, a boosting process iteratively builds a strong classifier as a linear combination of elements in $\mathcal{H}$. The key idea of boosting is to weight the training samples by attaching at each iteration more importance to samples that are, so far, wrongly predicted by the current ensemble. To illustrate formally this idea, we expose here briefly one of the most notable boosting algorithms called AdaBoost [Freund and Schapire, 1997] in the case of binary classification where $\mathcal{Y} = \{-1, 1\}$. Each training sample $(x_i, y_i) \in \mathcal{S}$ has a weight $w_i^{(t)}$ initialized to $\frac{1}{|\mathcal{S}|}$ for the first iteration $t = 1$. At iteration t, we:

- Find the classifier $f_t \in \mathcal{H}$ minimizing the weighted error on the training set $\epsilon_t(f) = \sum_{i|f(x_i)\neq y_i} w_i$.
- Compute the weight $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t(f_t)}{\epsilon_t(f_t)}\right)$ which will eventually state the importance of the weak learner f_t in the ensemble f.
- Update the weights of the training instances to give more importance to misclassified samples: $w_i^{(t+1)} = w_i^{(t)} \exp\left(-\alpha_t y_i f_t(x_i)\right)$, i.e. $w_i^{(t+1)} = w_i^{(t)} \exp\left(\alpha_t\right)$ if x_i was misclassified by f_t and $w_i^{(t+1)} = w_i^{(t)} \exp\left(-\alpha_t\right)$ if the classification was correct. The weights $\left(w_i^{(t+1)}\right)_{1\leq i\leq N}$ are then normalized to sum to 1.

After a predefined number of iterations T, the final ensemble classifier is defined as $f = \sum_{1\leq t\leq T} \alpha_t f_t$. To summarize, the AdaBoost algorithm allows to combine linearly several classifiers which might be individually only slightly better than random into a strong classifier. As an example, AdaBoost has been notably effective to detect faces in images by combining several low-level decisions which simply compare intensities within rectangular parts of an image [Viola and Jones, 2004]. In the case of decision tree models for semantic segmentation, we will encounter again both this type of features (see Sec. 1.3.1 and Sec. 2.4.1) and the idea of combining weak features into strong models, with the difference that the combination will be hierarchical instead of linear.

1.3 Semantic Segmentation as Pixelwise Classification

After the general overview of the basic concepts of supervised learning presented in Sec. 1.2, we now go back to the original problem of semantic image segmentation and discuss how to define this objective as a learning scenario. Assuming both the dimensionality d and the number of channels n_{channels} of the images fixed, a training set is composed of images labeled by a human expert, i.e. couples $\left(I_1, \hat{L}(I_1)\right), \ldots, \left(I_N, \hat{L}(I_N)\right)$ where each I_i is a d-dimensional image and $\hat{L}(I_i)$ its corresponding true labeling. In this context, the direct application of the supervised learning framework invites to define the set of observations $\mathcal{X}$ as the set of all possible input images (of all possible sizes), and the label space $\mathcal{Y}$ would be similarly defined as the set of all labelings, i.e of images taking their pixel values among the predefined set of semantic labels. However, the output space made of image labelings has a more complex structure than in the classical cases of classification and regression and is therefore usually not compatible with standard supervised learning techniques. Predicting an entire image labeling corresponds in fact to a structured learning scenario and prompts more sophisticated learning techniques (see Sec. 1.4 and Sec. 2.3.2). In this section, we expose the simplest approach for learning-based semantic segmentation, which treats each pixel as an independent observation and uses the formalism of supervised classification to predict their individual label.

In the pixelwise classification model, we define an observation x as a location within a particular image, i.e. as a couple $x = (\mathbf{p}, I)$, where I is an image and $\mathbf{p} \in \Omega_I$ the location in the image domain. Given an acquired image I, we denote $\mathcal{X}_I = \{(\mathbf{p}, I), \mathbf{p} \in \Omega_I\}$ the set of all observations belonging to an acquired image I, and $\mathcal{X} = \cup_{I \in \mathcal{I}_{d,n_{\text{channels}}}} \mathcal{X}_I$ the set of observations given an image dimensionality d and a number of image modalities (or color channels) n_{channels}. $I \in \mathcal{I}_{d,n_{\text{channels}}}$ is the set of all d-dimensional images with n_{channels} modalities. The label space $\mathcal{Y}$ is the set of possible semantic labels to which a pixel can be associated. Splitting the image labeling into independent pixelwise decisions strongly simplifies the learning task in comparison to the structured case: for a same number of labeled images, the available amount of training data is larger (equal to the number of labeled pixels instead of the number of labeled images) and the size of the label space is reduced and independent of the image size (K instead of $K^{|\Omega_I|}$). The simplified setting of pixelwise classification is therefore more suitable for the application of standard classification algorithms and allows to perform predictions independently of the size of the images.

As illustrated in Sec. 1.2.1, the design of a standard supervised learning approach can be usually decomposed into two steps:

- The feature extraction, which allows to transform the abstract space of observations $\mathcal{X}$ into a mathematically tractable (e.g. Euclidean) feature space via the definition of a collection of real-valued functions $(\phi_{\boldsymbol{\lambda}})_{\boldsymbol{\lambda} \in \Lambda}$, which extract in a quantitative way a certain type information about an observation $x \in \mathcal{X}$.

- The definition of a learning strategy, which creates a function $f : \mathcal{X} \to \mathcal{Y}$ from a set of labeled examples whose label prediction for an observation $x \in \mathcal{X}$ is only based on the extracted feature responses $(\phi_{\boldsymbol{\lambda}}(x))_{\boldsymbol{\lambda} \in \Lambda}$.

In the two following subsections, we respectively expose the classical strategies encountered to tackle these two steps in the case of pixelwise classification, i.e. where an observation x is a pixel in an image.

1.3.1 Modeling the Visual Context around a Pixel

The objective of the feature extraction stage is to create a quantitative representation of an observation $x \in \mathcal{X}$ based on which (and only on which) the decisions about the label of x will be performed. To do so, we must define a collection $(\phi_{\boldsymbol{\lambda}})_{\boldsymbol{\lambda} \in \Lambda}$ of functions called feature maps, where each $\phi_{\boldsymbol{\lambda}} : \mathcal{X} \to \mathbb{R}$ extracts a certain kind of visual cue about x. In a pixelwise classification setting, each observation $x = (\mathbf{p}, I)$ is defined as a location $\mathbf{p}$ in a particular image I so that the role of the functions $(\phi_{\boldsymbol{\lambda}})_{\boldsymbol{\lambda} \in \Lambda}$ is to quantify the visual information at the location $\mathbf{p}$ in the image I. The choice of features for quantifying an observation has a critical impact on the learning and prediction tasks, both in terms of accuracy and computational time. Prior knowledge about the application at hand can be effectively incorporated directly in the features: for instance, if the task is known to be invariant to a certain kind of image deformation, only considering features possessing this invariance guarantees the invariance of the predictions. In general, including prior knowledge into the features reduces the necessary amount of training data for the learning task since it intuitively reduces the complexity of the task. Engineering features can thus be particularly useful in situations where training data are difficult to obtain.

We expose now several simple or classical examples of feature choices, i.e. some possible types of feature maps. These examples can in practice be concatenated to form the final collection of features. We consider for simplicity the case of a 2D image where elements of the image domain are of the form $\mathbf{p} = (u, v)$. These examples will also give us the opportunity to become familiar with some concepts encountered in the field of image segmentation and on which our first methodological contribution in this thesis will be based (Sec. 2.4).

Local Features

To characterize an observation $x = (\mathbf{p}, I)$, the most immediate solution is perhaps to focus on the image value observed at the location $\mathbf{p}$ and in its immediate vicinity. We can for instance:

- Extract the image value for a given color channel i.e. $\phi_c^{\text{val}}(x) = I(\mathbf{p}, c)$
- Consider the differences with neighboring pixels such as the horizontal difference $\phi_c^{\text{diff\_hor}}(x) = \frac{\partial I}{\partial u}(\mathbf{p}, c) = I(u+1, v, c) - I(u, v, c)$.
- More generally, compute features based on the gradient $\vec{\nabla} I(\mathbf{p}, c)$ of I at the location $\mathbf{p}$ and for the color channel c. We can for instance define feature maps extracting the norm of the gradient, i.e. $\phi_c^{\text{grad\_norm}}(x) = \left\| \vec{\nabla} I(\mathbf{p}, c) \right\|$, or its orientation $\phi_c^{\text{grad\_or}}(x) = \operatorname{atan2}(\frac{\partial I}{\partial v}(\mathbf{p}, c), \frac{\partial I}{\partial u}(\mathbf{p}, c))$. A high value of the norm of the gradient can be for example a good indicator of the presence of an edge at the considered location.

In practice, unless the application is strongly related to color information, local features are alone not sufficient. The image values at the location of interest and at its neighboring locations are not informative enough to perform an accurate decision: a human who would only see the pixel of interest or a 3×3 patch centered on it could not make any decision about its semantic content for most applications. In general, more visual context is thus required, meaning that we have to enlarge the size of the patch centered on **p**.

Filter-Based Features

Considering a patch of size $\bar{\delta} \times \bar{\delta}$ centered on $\mathbf{p}$ (where, therefore, we consider an odd value for $\bar{\delta} = 2\delta + 1$), filter-based features propose to compute a weighted sum of the intensities within this patch. A simple example is the computation of the mean within a color channel, defined for an observation $x = (\mathbf{p}, I)$ as

$$\phi^{\text{mean}}_{c,\delta}(x) = \sum_{\substack{-\delta \le u \le \delta \\ -\delta \le v \le \delta}} \frac{1}{(2\delta+1)^2} I\left(\mathbf{p} - (u,v), c\right). \tag{1.24}$$

Spatially varying weights can also be used, such as Gaussian weights of covariance matrix Σ:

$$\phi^{\text{gaussian}}_{c,\delta,\Sigma}(x) = \sum_{\substack{-\delta \le u \le \delta \\ -\delta \le v \le \delta}} G_\Sigma(u,v) I\left(\mathbf{p} - (u,v), c\right) \tag{1.25}$$

where, writing (u, v) as a column vector $\mathbf{h}$, we have $G_\Sigma(u,v) \propto \exp(-\mathbf{h}^T \Sigma \mathbf{h})$ and

$$\sum_{\substack{-\delta \le u \le \delta \\ -\delta \le v \le \delta}} G_\Sigma(u,v) = 1. \tag{1.26}$$

The choice of Gaussian weights has a smoothing effect and can therefore give a version of ϕ^{val} which is more robust to noise or allows more accurate computations of derivatives. More generally, we can define a filter-based feature by providing a collection of weights $\mathbf{W} = (w_{u,v})_{-\delta \le u,v \le \delta}$ and compute

$$\phi^{\text{filter}}_{c,\delta,\mathbf{W}}(x) = \sum_{\substack{-\delta \le u \le \delta \\ -\delta \le v \le \delta}} w_{u,v} I\left(\mathbf{p} - (u,v), c\right). \tag{1.27}$$

An appropriate choice of weights can output a characteristic response if a certain kind of structure is contained in the patch. This idea is for instance at the origin of a popular method for enhancement of vessels in medical images [Frangi et al., 1998]. A large number of different filter-based feature maps can also be extracted to form a filter bank, counting on the fact that their combination is sufficiently informative. Gabor filters [Fogel and Sagi, 1989] are an example of such a strategy. Filter-based features are also a basic building block of some deep learning approaches such as convolutional neural networks.

In addition, filter-based features can be computed efficiently in the context of pixel-wise segmentation. The value of a function $\phi^{\text{filter}}_{c,\delta,\mathbf{W}}$ can be computed quickly at all locations of the image I by performing a convolution product along the c^{th} color channel, i.e. $\phi^{\text{filter}}_{c,\delta,\mathbf{W}}(x) = \left[I(.,c) \star W\right](\mathbf{p})$, for which fast computation techniques exist based for example on the fast Fourier transform.

Haar Features

Away from application-oriented features and going rather in the same direction as a large filter bank, Haar features encode low-level information in a generic and computationally efficient way. They were originally used in combination with the AdaBoost classifier (Sec. 1.2.7), first in the context of face detection [Viola and Jones, 2004] and later extended to object recognition and segmentation [Shotton et al., 2006]. While their exact implementation technically appears in numerous variants, the key concept of Haar features is to represent a pixel $\mathbf{p}$ in an image by its content in two boxes located at offset locations within a (possibly large) patch of size $\bar{\delta} \times \bar{\delta}$. More precisely, for a d-dimensional image, the feature maps $\left(\phi^{\text{Haar}}_{\boldsymbol{\lambda}}\right)_{\boldsymbol{\lambda}\in\Lambda}$ are parametrized by a vector

$$\boldsymbol{\lambda} = (\vec{v_1}, \vec{v_2}, \mathbf{s}_1, \mathbf{s}_2, c_1, c_2, \omega)\,, \tag{1.28}$$

where $\vec{v_1}, \vec{v_2} \in \{-\delta, \ldots, \delta\}^d$, $\mathbf{s}_1, \mathbf{s}_2 \in \{1, \ldots, \delta\}^d$, $c_1, c_2 \in \{1, \ldots, n_{\text{channels}}\}$ and $\omega : \mathbb{R}^2 \to \mathbb{R}$ performs an operation between two numbers (Fig. 1.3a). To compute the feature response $\phi_{\boldsymbol{\lambda}}(x)$ at an observation $x = (\mathbf{p}, I)$ for a given choice of $\boldsymbol{\lambda}$, we:

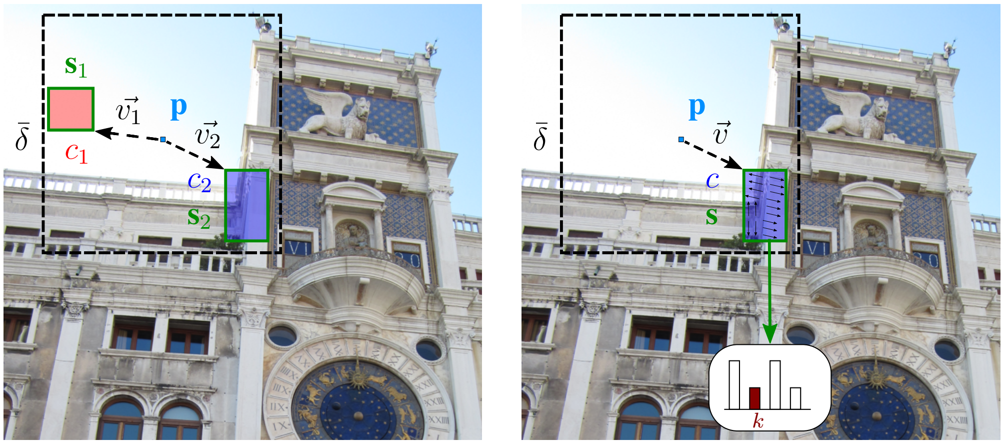
1. Consider the two boxes $\mathcal{B}(\mathbf{p} + \vec{v_i}, \mathbf{s}_i), 1 \leq i \leq 2$ respectively centered on $\mathbf{p} + \vec{v_i}$ and of size $\mathbf{s}_i$.

2. Compute in each box the average $\bar{B}_i(x, \boldsymbol{\lambda})$ of the image intensities over the specified color channel c_i, i.e. for $i \in \{1, 2\}$ $\bar{B}_i(x, \boldsymbol{\lambda}) = \langle I(\mathbf{p}, c_i)\rangle_{\mathcal{B}(p+\vec{v_i}, \mathbf{s}_i)}$.

3. Combine these two averages with the operation ω, so that the final feature value is $\phi^{\text{Haar}}_{\boldsymbol{\lambda}}(x) = \omega\left(\bar{B}_1(x, \boldsymbol{\lambda}), \bar{B}_2(x, \boldsymbol{\lambda})\right)$.

Possible operations ω can for example be the difference $\omega(a, b) = a - b$, the sum $\omega(a, b) = a + b$ or the absolute value of the difference $\omega(a, b) = |a - b|$. This set of operations corresponds, in fact, to one of the first uses of Haar features in the random forest framework [Shotton et al., 2008], but can be adapted according to the available prior knowledge of the application at hand. In Sec. 2.4, we will for example only allow binary versions of the difference to enforce the invariance to changes of contrast in certain types of medical applications (magnetic resonance and ultrasound imaging).

On one hand, each individual Haar feature encodes a weak type of information, which is alone unlikely to be sufficient to perform accurate predictions. On the other hand, the set Λ of possible parameters $\boldsymbol{\lambda}$ as defined in Eq. 1.28, and thus the amount of available feature maps $\left(\phi^{\text{Haar}}_{\boldsymbol{\lambda}}\right)_{\boldsymbol{\lambda}\in\Lambda}$, is very large, so that lots of different low-level cues are present in this feature representation. This is a key characteristic of Haar features: although individually weak, they are sufficiently generic and in large number to be suitable for many applications if used together with an appropriate learner able to combine them effectively. Therefore, they are mostly used in boosting and tree ensembles. At first sight, the use of Haar features faces a computational problem. For an observation $x \in \mathcal{X}$, computing all feature responses $\left(\phi^{\text{Haar}}_{\boldsymbol{\lambda}}(x)\right)_{\boldsymbol{\lambda}\in\Lambda}$ is not feasible due to their large number. Therefore, we could either decide to limit ourselves to a small subset of these features and ignore the other ones, which limits the richness of the description, or we could recompute on-the-fly



(a) Haar Features (b) Histograms of Gradients

Figure 1.3: **Contextual features.** The visual appearance around a pixel $\mathbf{p}$ is quantitatively described by its surroundings within a patch of size $\bar{\delta} \times \bar{\delta}$. (a) Each Haar feature consists of the combination of the average of the intensities within two offset boxes. (b) In the case of histograms of gradients, each feature corresponds to one of the bins of a histogram of the gradient directions observed within a box.

each feature value $\phi_{\boldsymbol{\lambda}}(x)$ whenever it is needed during training or prediction. However, the latter choice seemingly requires repeated computations of the mean intensity over boxes, whose cost grows linearly with the number of pixels in the box and could be computationally too expensive. In fact, this situation can be alleviated by the use of integral images. For an image I, at the cost of precomputing once an integral image $\tilde{I}$, the sum of intensities over any rectangular box can be obtained in a few memory accesses only [Viola and Jones, 2004]. With an integral image, we thus have a straightforward access to $\phi^{\text{Haar}}_{\boldsymbol{\lambda}}(x)$ for every $\boldsymbol{\lambda} \in \Lambda$ and every observation $x \in \mathcal{X}_I$ belonging to the image I. We refer to Appendix A for a detailed description of integral images.

Histograms of Gradients

Our last example of popular features are the histograms of gradients (HoG) introduced in the context of human detection [Dalal and Triggs, 2005]. HoGs encode the distribution of gradient orientations around the location of interest, which intuitively describes the shape of the object a pixel belongs to or the shape of the surrounding structures. We limit our description to the 2-dimensional case and consider the parametrization

$$\boldsymbol{\lambda} = (\vec{v}, \mathbf{s}, c, k)\,, \tag{1.29}$$

where $\vec{v} \in \{-\delta, \dots, \delta\}^2$, $\mathbf{s} \in \{1, \dots, \delta\}^2$, $c \in \{1, \dots, n_{\text{channels}}\}$ and $k \in \{1, \dots, B\}$ where B is a predefined number of bins. Within the box $\mathcal{B}(\mathbf{p} + \vec{v}, \mathbf{s})$, we create the histogram of gradient orientations, weighting each gradient by its magnitude, and output the value of the k^{th} histogram bin (Fig. 1.3b). For a location $\mathbf{u} \in \Omega_I$, we denote $N_\nabla(\mathbf{u}, c) \in \mathbb{R}_+$ and $O_\nabla(\mathbf{u}, c) \in [0, 2\pi[$ the magnitude and orientation of the gradient

$\vec{\nabla} I(\mathbf{u}, c)$ respectively. The response at $x = (\mathbf{p}, I)$ given a feature $\boldsymbol{\lambda}$ can be mathematically written as

$$\phi_{\boldsymbol{\lambda}}^{\mathrm{HoG}}(x) = \frac{\sum_{\mathbf{u}\in\mathcal{B}(\mathbf{p}+\vec{v},\mathbf{s})} N_{\nabla}(\mathbf{u}, c)\mathbf{1}_{[\frac{2(k-1)\pi}{B}, \frac{2k\pi}{B}[}\{O_{\nabla}(\mathbf{u}, c)\}}{\sum_{\mathbf{u}\in\mathcal{B}(\mathbf{p}+\vec{v},\mathbf{s})} N_{\nabla}(\mathbf{u}, c)}, \tag{1.30}$$

where $\mathbf{1}_{[\frac{2(k-1)\pi}{B}, \frac{2k\pi}{B}[}\{O_{\nabla}(\mathbf{u}, c)\} = 1$ if the orientation $O_{\nabla}(\mathbf{u}, c)$ belongs to the k^{th} histogram bin defined as $[\frac{2(k-1)\pi}{B}, \frac{2k\pi}{B}[$, and 0 otherwise. We can notice that HoGs are invariant to changes of illuminations, i.e. stay unchanged when a constant intensity is added to the considered color channel. From Eq. 1.30, we can see that $\phi_{\boldsymbol{\lambda}}^{\mathrm{HoG}}(x)$ is the ratio of two sums over a rectangular box. Assuming that B is fixed, we can consider, for every $k \in \{1, \ldots, B\}$, the image $I_{\nabla,k}$ of the same dimensions as I defined as

$$I_{\nabla,k}(\mathbf{u}, c) = N_{\nabla}(\mathbf{u}, c)\mathbf{1}_{[\frac{2(k-1)\pi}{B}, \frac{2k\pi}{B}[}\{O_{\nabla}(\mathbf{u}, c)\}, \tag{1.31}$$

and the gradient magnitude image N_{∇} (which is independent of B and k). With these definitions, we have

$$\phi_{\boldsymbol{\lambda}}^{\mathrm{HoG}}(x) = \frac{\sum_{\mathbf{u}\in\mathcal{B}(\mathbf{p}+\vec{v},\mathbf{s})} I_{\nabla,k}(\mathbf{u}, c)}{\sum_{\mathbf{u}\in\mathcal{B}(\mathbf{p}+\vec{v},\mathbf{s})} N_{\nabla}(\mathbf{u}, c)}. \tag{1.32}$$

Hence, the response $\phi_{\boldsymbol{\lambda}}^{\mathrm{HoG}}(x)$ corresponding to any parameter $\boldsymbol{\lambda}$ and observation $x \in \mathcal{X}_I$ is a ratio of sums computed over two boxes. Therefore, as was the case for Haar features, $\phi_{\boldsymbol{\lambda}}^{\mathrm{HoG}}(x)$ can be quickly computed for any value of $\boldsymbol{\lambda}$ and x via the precomputation of $B + 1$ integral images (from the images $I_{\nabla,1}, \ldots, I_{\nabla,B}$ and N_{∇}). Although HoG features are usually computed for a fixed set of boxes regularly distributed over the location of interest **p**, we will propose in this thesis to merge them together with Haar features in a joint generic representation of color and gradient information (Sec. 2.4.1).

1.3.2 Classification Methods for Pixelwise Semantic Segmentation

Once the choice of a feature representation has been made, the modeling of semantic image segmentation as a pixelwise classification task allows the use of any standard supervised classification technique. However, in practice, the semantic segmentation scenario can involve huge amounts of training data. A single labeled 2D image of standard size (e.g 300×300 pixels) already contains around 10^5 training samples, with even more extreme situations for 3D medical data. Therefore, scalability at training and prediction time can quickly become crucial due to the high number of instances to be processed independently. Boosting-based and tree-based approaches (often used with Haar-like features) have had a lot of popularity for both 2D and 3D cases, due to the efficiency of this combination and, in the case of decision trees, to the natural treatment of multi-class problems. We propose a detailed review on forest-based semantic segmentation approaches in Sec. 2.3.

In the situations where a lot of labeled data is available, convolutional neural networks (CNNs) have recently gained a huge interest in computer vision in general, and have demonstrated state-of-the-art performance for many applications including semantic image segmentation [Long et al., 2015, Zheng et al., 2015]. Starting from a most elementary image representation, CNNs jointly optimize the feature representation and the decision rule via a series of stacked convolutional layers, which can be handled in a tractable manner with GPU computing. The use of CNNs has also been proposed for the semantic segmentation of 3D volumes in the medical field [Ronneberger et al., 2015, Roth et al., 2015, Havaei et al., 2016].

In this thesis, we will focus on the random forest model and will review in the next chapter the existing forest-based approaches for semantic image segmentation in details, both in the computer vision and medical communities.

1.4 Structured Learning

By modeling the semantic segmentation objective as a pixelwise classification problem in Sec. 1.3, we reduced the size of the output space and the complexity of the learning task but ignored the fact that both inputs and outputs are fundamentally images, i.e. objects containing a certain amount of structure. The field of structured learning [Nowozin and Lampert, 2011] focuses on the development of methods treating both inputs and outputs directly as images and modeling dependencies between pixels. For example, an image can be seen as a graph where neighboring pixels are connected and dependencies between neighboring pixels are modeled, so that unrealistic label configurations can be avoided in the predicted labeling. Conditional random fields [Lafferty et al., 2001], abbreviated CRFs, are a class of graphical models which has been widely used for semantic image segmentation. Using the pixelwise output of a classifier as *unary* probability, additional *binary* terms are introduced to model the contextual relationship between a pixel and its neighbors, so that likely transitions between neighboring pixels based on the observed visual content can be inferred [Rabinovich et al., 2007, Shotton et al., 2009]. To overcome the fact that standard CRF-based approaches are often restricted to short-range configurations, multiscale or hierarchical CRFs have been designed [He et al., 2004, Gonfaus et al., 2010]. Away from fixed topologies, the auto-context approach [Tu and Bai, 2010] generates a series of classifiers, where the output probability of the $(k-1)^{\text{th}}$ classifier is used as additional features for training the k^{th} one, thus gaining contextual knowledge progressively, in an automatic and flexible way. For an auto-context framework to be effective, it is essential that classifiers do not overfit, as feeding a perfect posterior probability (on the training set) to the next classifier leads to an early convergence where all next classifiers simply reproduce this input. Structured approaches based on the concept of decision forests have also been developed and will be reviewed in more details in Sec. 2.3.2.

1.5 Conclusion

In this chapter, we have presented an overview of the main concepts and challenges of the field of supervised learning, which aims at automatically creating a predictive model from a set of labeled observations. We also exposed how the semantic image segmentation objective can be modeled within the supervised learning framework, either by performing independent pixelwise predictions or by directly exploiting the structure of images. After this introductory chapter, we propose to treat more specifically the case of decision forests in Chapter 2. By describing in details this class of models and the related approaches, we will illustrate further most of the notions mentioned in this first chapter.

Chapter 2

Decision Forests

In this chapter, we expose in details the model of decision forests, which is one of the most popular supervised learning techniques for classification and regression. We start by describing it as a general machine learning technique and discuss its possible variants, before focusing more specifically on the semantic image segmentation task, both in computer vision and medical applications. After having reviewed the diverse challenges occurring in this context and the already existing approaches, we introduce the first methodological contribution of this thesis: an easy-to-implement alternative to the standard forest training technique which, at no computational cost nor tuning of additional parameters, improves the segmentation accuracy on a series of datasets.

2.1 Decision Trees

The decision forest framework builds on the concept of decision tree which we expose in this section. Considering a supervised learning setup (Sec. 1.2), we assume given a label space $\mathcal{Y}$ and an observation space $\mathcal{X}$ which is only numerically accessible via a collection of feature maps $(\phi_{\boldsymbol{\lambda}})_{\boldsymbol{\lambda}\in\Lambda}$. A decision tree follows a discriminative strategy, i.e. models for each $y \in \mathcal{Y}$ the probability[1] $P(Y = y|X = x)$ that an observed sample x has the label y.

2.1.1 Description of the Tree Structure

A decision tree is a rooted hierarchical structure made of nodes such that each node has exactly 0 or 2 child nodes. Nodes having 2 child nodes are said internal and contain a splitting rule $\psi : \mathcal{X} \to \{0, 1\}$ which routes an observation $x \in \mathcal{X}$ towards the left ($\psi(x) = 0$) or right ($\psi(x) = 1$) child node. By recursive application of these splitting rules starting from the root node, an observation is directed towards a terminal node or leaf L, which does not have any child nodes and contains a prediction model $P(Y|X \in L)$ instead of a splitting function. As a consequence, the leaves partition the input space $\mathcal{X}$ and provide a piecewise approximation of the probability $P(Y|X = x)$. A simple example

[1]Or a density function in the regression case.

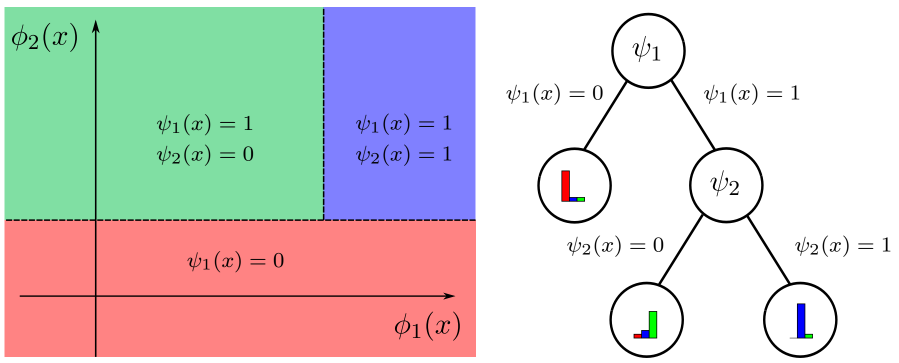


Figure 2.1: **Example of a decision tree.** We consider here a feature space where $\Lambda = \{1, 2\}$, i.e. where observations $x \in \mathcal{X}$ are modeled with exactly two feature maps ϕ_1 and ϕ_2, and a classification scenario with 3 labels `Red`, `Blue` and `Green`. A decision tree is a collection of nodes which are hierarchically ordered, starting from a root node at the top. Each internal node contains a splitting function, which are in our example axis-aligned. At the root node, a split with respect to the dimension ϕ_2 is performed. Another internal node divides the corresponding upper subspace with respect to the value of ϕ_1. At each leaf, a probability distribution over labels is stored which models the predicted label for observations belonging to this part of the feature space.

of decision tree in a two-dimensional space is shown in Fig. 2.1. Since our supervised learning formulation assumes that the knowledge about an observation $x \in \mathcal{X}$ is only accessible through some feature maps, the splitting functions stored in internal nodes must be based on the feature responses $(\phi_{\boldsymbol{\lambda}}(x))_{\boldsymbol{\lambda}\in\Lambda}$. The most usual splitting functions are said axis-aligned and defined by one feature $\boldsymbol{\lambda} \in \Lambda$ and a threshold $\theta \in \mathbb{R}$ such that an observation x is sent left if $\phi_{\boldsymbol{\lambda}}(x) < \theta$ and is sent right otherwise. Formally, axis-aligned functions can be written as

$$\psi(x) = H(\phi_{\boldsymbol{\lambda}}(x) - \theta), \tag{2.1}$$

where H is the Heaviside function which outputs 1 if its input is nonnegative and 0 otherwise. Although axis-aligned splitting functions remain the most used in the literature, more complex ones can be encountered. For example, oblique functions generalize axis-aligned ones by combining several features:

$$\psi(x) = H(w_0 + \sum_{k=1}^{d} w_k \phi_{\boldsymbol{\lambda}_k}(x)). \tag{2.2}$$

The popularity of axis-aligned splits resides in their simplicity which, at prediction time, requires the computation of only one feature at each internal node. As we will discuss further in Sec. 2.1.4, axis-aligned functions are also easier to optimize at training time. Unless otherwise mentioned, splitting functions are assumed axis-aligned in this thesis.

From their structure alone, some advantages of decision tree models are already apparent:

- To predict the label of an observation, only the features located on its path through the tree must be computed. This means in particular that the set of features can be very large or even infinite without affecting the computational time, as all the features that are not used will simply be ignored. Due to this aspect, Haar features (Sec. 1.3.1) can be used in decision trees for semantic segmentation.

- Since features are only looked individually at each node in the axis-aligned case, they do not need to be comparable with each other. While many learning algorithms perform analytic combinations of features (such as a linear combination in logistic regression (Eq. 1.17) or the definition of a distance (Eq. 1.19) for a k-nearest neighbor model) which may require some pre-processing or normalization beforehand, this does not apply to decision trees which can for instance handle a combination of real-numbered features and categorical ones.

- Decision trees are easily interpretable. The series of decisions leading to the final label prediction can be easily read on the path of the observation through the tree, where each decision involves only one feature at a time.

In a supervised learning scenario, a decision tree (i.e. its node arrangement, the splitting functions stored in the internal nodes and the predictive models stored in the leaf nodes) is automatically inferred from a set of labeled observations $\mathcal{S} = \{(x_i, y_i), 1 \leq i \leq N\}$. We now describe the standard training procedure which generates a decision tree from such a training set.

2.1.2 Overview of the Training Procedure

In the literature, several variants can be considered as a standard training of a decision tree [Breiman et al., 1984, Quinlan, 1986, 1993, Criminisi, 2011]. In fact, the main idea behind these different strategies is similar and they differ only through some technical details. We give here a global overview and discuss more in details in the next subsections the different possible variants for some building blocks of the training, as well as the choices we retained in our own implementation which was used for the contributions of this thesis.

Given a training set $\mathcal{S} = \{(x_i, y_i), 1 \leq i \leq N\}$ where each $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ is a pair made of an observation and a label, the goal of the training procedure is to learn a decision tree able to approximate the probability $P(Y|X = x)$ related to the mechanism generating these observed samples. The training of a decision tree is conducted recursively and leads to the progressive creation of nodes, starting with a single node which will be the root of the learned tree. Using as input the training set $\mathcal{S}$, we first decide on the splitting function which will be stored in this root node. This splitting function is obtained via an optimization procedure looking for a split of high quality, where the quality of a candidate splitting function ψ is intuitively defined as its ability to group together the training instances of $\mathcal{S}$ which possess similar labels. More precisely, we look for a split ψ splitting

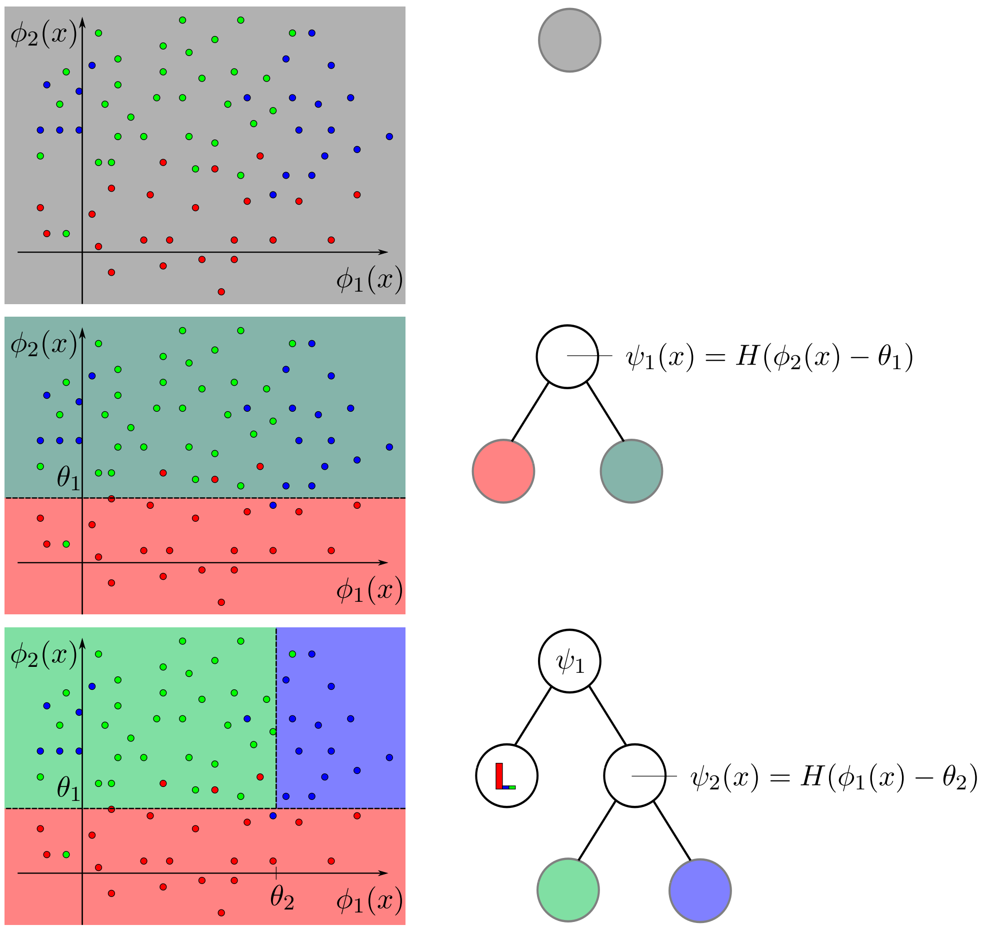

$\phi_2(x)$
$\phi_1(x)$
θ_1
θ_2
$\psi_1(x) = H(\phi_2(x) - \theta_1)$
ψ_1
$\psi_2(x) = H(\phi_1(x) - \theta_2)$

Figure 2.2: **Training a decision tree**. We consider the setting of Fig. 2.1 and show here a possible scenario leading to the decision tree shown in Fig. 2.1. Given a set of labeled observations, a tree is trained recursively by finding at each stage the most informative splits, i.e. the splits which intuitively separate the labels best. At each node, stopping criteria are tested to decide whether one keeps splitting or creates a leaf instead. In the latter case, the leaf prediction model is inferred from the labels of the training samples contained in the corresponding part of the feature space.

the data $\mathcal{S}$ into two subsets as pure[2] as possible. More precisely, we consider the two sets

$$\mathcal{S}_{\psi,\text{left}} = \{(x, y) \in \mathcal{S} | \psi(x) = 0\} \tag{2.3}$$

and

$$\mathcal{S}_{\psi,\text{right}} = \{(x, y) \in \mathcal{S} | \psi(x) = 1\} \tag{2.4}$$

which correspond to the way ψ distributes samples to the left and right child nodes at prediction time. Once an impurity measure G has been defined (Sec. 2.1.3), the quality

[2]An impurity score $G(\mathcal{S})$ of an arbitrary set of labeled samples $\mathcal{S}$ is intuitively equal to 0 if all instances in $\mathcal{S}$ share the same label and maximal if the labels in $\mathcal{S}$ are uniformly distributed over their set of possible values $\mathcal{Y}$. We give explicit examples of impurity measures for both classification and regression in Sec. 2.1.3.

of the split ψ is then defined as a purity gain

$$\text{Gain}(\psi, \mathcal{S}) = G(\mathcal{S}) - \frac{|\mathcal{S}_{\psi,\text{left}}|}{|\mathcal{S}|} G(\mathcal{S}_{\psi,\text{left}}) - \frac{|\mathcal{S}_{\psi,\text{right}}|}{|\mathcal{S}|} G(\mathcal{S}_{\psi,\text{right}}), \tag{2.5}$$

where $|\mathcal{S}|$ denotes the number of elements in the set of samples $\mathcal{S}$. This definition being given, the splitting function $\hat{\psi}$ that is eventually retained and stored at the root node is the one which maximizes the purity gain among a predefined set Ψ of candidate splitting functions, i.e.

$$\hat{\psi} = \underset{\psi \in \Psi}{\text{argmax}}\, \text{Gain}(\psi, \mathcal{S}). \tag{2.6}$$

The actual feasibility of this optimization depends on the type of splitting functions that has been chosen and will be discussed in Sec. 2.1.4. Once the splitting function at the root has been selected, two child nodes are created and the corresponding subsets $\mathcal{S}_{\hat{\psi},\text{left}}$ and $\mathcal{S}_{\hat{\psi},\text{right}}$ of the training set are respectively sent to the left and right child nodes. The training procedure is then recursively repeated at these two child nodes, where the left child node (respectively the right child node) uses as input the set $\mathcal{S}_{\hat{\psi},\text{left}}$ (respectively $\mathcal{S}_{\hat{\psi},\text{right}}$) instead of $\mathcal{S}$. Before (or in some cases after) each node optimization, some stopping criteria (Sec. 2.1.5) are tested and, if one of them is satisfied, no child nodes are created and the current node is transformed into a leaf node. The predictive function stored at the leaf is then empirically estimated from the set of training samples that has been sent to this node (Sec. 2.1.6). We illustrate the training procedure on Fig. 2.2.

Following this overview of the training of a decision tree, a few important design choices remain to be defined:

1. How to define an impurity measure G?
2. How is the optimization procedure defined by Eq. 2.6 conducted?
3. What kind of criteria should be used to stop the recursive growth of the tree?
4. How are the leaf predictive models computed from their respective set of arriving training instances?

The answer to the questions 1 and 4 directly depends on the nature of the task and prompts a different strategy for a classification problem ($\mathcal{Y} = \{c_1, \ldots, c_K\}$) and a regression problem ($\mathcal{Y} = \mathbb{R}^m$). In the segmentation case, we will be mostly interested in classification. The question 2 is perhaps the most difficult to answer in general, as the allowed type of splitting functions directly impacts the type and computational cost of the resulting optimization. The four next subsections respectively expose common strategies to address each one of these problems.

2.1.3 Impurity Measure

The impurity measure $G(\mathcal{S})$ of a set of labeled samples $\mathcal{S}$ is expected to be 0 if all samples in $\mathcal{S}$ have the same label and to be maximum in case of label uniformity over $\mathcal{S}$. We summarize here common choices for G. Starting with the classification case, where the set $\mathcal{Y} = \{c_1, \ldots, c_K\}$ consists of K discrete labels without particular ordering, two measures are particularly common: the Gini index [Breiman et al., 1984] and the Shannon entropy [Quinlan, 1986, 1993]. If we define, for $c \in \mathcal{Y}$, the set $\mathcal{S}_c = \{(x, y) \in \mathcal{S} | y = c\}$ of samples in $\mathcal{S}$ being labeled as c and the corresponding proportion $p_c = \frac{|\mathcal{S}_c|}{|\mathcal{S}|}$ of observations of label c, the Gini index is defined as

$$G(\mathcal{S}) = \sum_{c \in \mathcal{Y}} p_c (1 - p_c), \tag{2.7}$$

and the Shannon entropy as

$$G(\mathcal{S}) = - \sum_{c \in \mathcal{Y}} p_c \ln p_c. \tag{2.8}$$

Although they are not equivalent, there is no clear consensus on whether one of these two measures is better than the other [Raileanu and Stoffel, 2004]. We will use the Gini index (hence the notation G). In the case of a regression task where $\mathcal{Y} = \mathbb{R}^m$, an indicator of the label dispersion is the covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{m \times m}$, which can be empirically estimated as

$$\mathbf{\Sigma} = \frac{1}{|\mathcal{S}| - 1} \sum_{(x,y) \in \mathcal{S}} (y - \bar{y}) (y - \bar{y})^T , \tag{2.9}$$

where $\bar{y}$ is the empirical mean vector of the labels over $\mathcal{S}$. Using the covariance matrix, a popular impurity measure [Breiman et al., 1984] is its trace

$$G(\mathcal{S}) = \operatorname{tr}(\mathbf{\Sigma}), \tag{2.10}$$

which amounts to summing the m individual variances over each output dimension, or the differential entropy (under a Gaussian assumption) given by

$$G(\mathcal{S}) = \frac{m}{2} (1 + \ln (2\pi)) + \frac{1}{2} \ln |\mathbf{\Sigma}| , \tag{2.11}$$

where $|\mathbf{\Sigma}|$ is the determinant of $\mathbf{\Sigma}$. In practice, in an optimization context, the first term of Eq. 2.11 is constant and can be ignored. For all these impurity measures, the corresponding purity gain (Eq. 2.5) is non-negative. Note that the entropy estimates presented here are commonly used but, in fact, statistically biased. It has been recently shown that replacing them by better estimates increases the final accuracy of a decision tree for both discrete and differential entropies [Nowozin, 2012].

2.1.4 Splitting Function Optimization

Once an impurity measure has been chosen, the objective function Gain to be optimized in Eq. 2.6 is well defined. However, the choice of the optimization algorithm and of the set of functions Ψ over which the optimization is run is not straightforward. In most cases, an exhaustive optimization is too costly computationally and, in fact, rarely desired to guarantee good generalization abilities of the trained model (see Sec. 2.1.8 and Sec. 2.2.1). In this thesis, we will mainly use axis-aligned splitting functions and therefore describe in detail the corresponding standard optimization approach. We then review existing works related to the use and optimization of more sophisticated splitting functions.

Axis-Aligned Case: Greedy Optimization

As described in Sec. 2.1.1, an axis-aligned splitting function is defined by a feature $\boldsymbol{\lambda} \in \Lambda$ and a threshold $\theta \in \mathbb{R}$ and consists in thresholding one dimension of the feature space. In the axis-aligned case, the usual optimization procedure is conducted in a greedy fashion as follows. We first define a set of n_{tries} candidate features $\Lambda^{\text{candidates}} = \{\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^{n_{\text{tries}}}\}$. If the set of feature maps Λ is small enough, we can simply take $\Lambda^{\text{candidates}} = \Lambda$, but most of the time $\Lambda^{\text{candidates}}$ is in fact a randomly drawn subset of Λ, both out of computational necessity if Λ is too large but also to counter overfitting (Sec. 2.2.1). For each candidate feature $\boldsymbol{\lambda} \in \Lambda^{\text{candidates}}$, a predefined number $n_{\text{thresholds}}$ of thresholds are greedily tried. This set of thresholds $\Theta_{\boldsymbol{\lambda}} = \{\theta^1_{\boldsymbol{\lambda}}, \ldots, \theta^{n_{\text{thresholds}}}_{\boldsymbol{\lambda}}\}$ is taken between the extremal observed values over the input set of samples $\mathcal{S}$ sent to the node, i.e.

$$\theta^{\min}_{\boldsymbol{\lambda}} = \min_{(x,y)\in\mathcal{S}} \phi_{\boldsymbol{\lambda}}(x) \tag{2.12}$$

and

$$\theta^{\max}_{\boldsymbol{\lambda}} = \max_{(x,y)\in\mathcal{S}} \phi_{\boldsymbol{\lambda}}(x). \tag{2.13}$$

We will more precisely pick these candidate thresholds on a regular grid, i.e

$$\theta^{i}_{\boldsymbol{\lambda}} = \theta^{\min}_{\boldsymbol{\lambda}} + \frac{i\left(\theta^{\max}_{\boldsymbol{\lambda}} - \theta^{\min}_{\boldsymbol{\lambda}}\right)}{n_{\text{thresholds}} + 1}. \tag{2.14}$$

Other possibilities include taking these thresholds as randomly sampled uniformly between $\theta^{\min}_{\boldsymbol{\lambda}}$ and $\theta^{\max}_{\boldsymbol{\lambda}}$ [Geurts et al., 2006] or following quantiles of the feature response. The strategy above assumes that the feature map $\phi_{\boldsymbol{\lambda}}$ take continuous real-values. If it is known that $\phi_{\boldsymbol{\lambda}}$ takes a finite set of values, the threshold optimization is usually straightforward, although it can also be randomized out of computational necessity. We will especially encounter the case where $\phi_{\boldsymbol{\lambda}}$ is binary (outputing only 0 or 1) in Sec. 2.4.1, in which case a single threshold (for instance $\theta_{\boldsymbol{\lambda}} = 0.5$) needs to be tried.

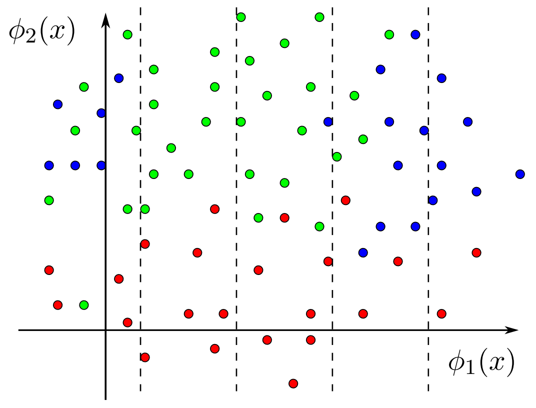
Under these considerations, we pick among these finite combinations the couple feature / threshold maximizing the gain (Eq. 2.6), i.e. we retain the axis-aligned splitting function $\hat{\psi}$ based on the feature $\hat{\boldsymbol{\lambda}}$ and the threshold $\hat{\theta}$, where

$$\hat{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}\in\Lambda^{\text{candidates}}}{\operatorname{argmax}} \max_{\theta\in\Theta_{\boldsymbol{\lambda}}} \text{Gain}(\boldsymbol{\lambda}, \theta, \mathcal{S}) \tag{2.15}$$

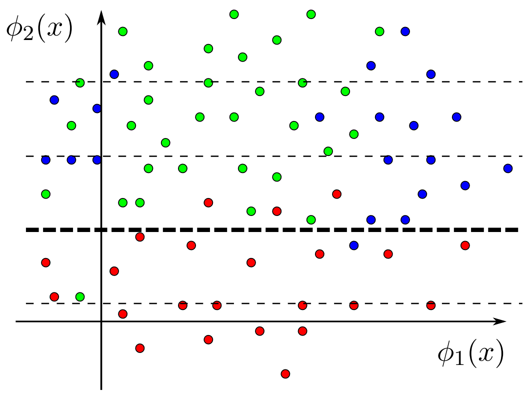
and

$$\hat{\theta} = \underset{\theta\in\Theta_{\hat{\boldsymbol{\lambda}}}}{\operatorname{argmax}} \text{Gain}(\hat{\boldsymbol{\lambda}}, \theta, \mathcal{S}). \tag{2.16}$$

Figure 2.3 illustrates this optimization. The gain $\text{Gain}(\boldsymbol{\lambda}, \theta, \mathcal{S})$ above is naturally defined as the gain $\text{Gain}(\psi, \mathcal{S})$ where ψ is the axis-aligned splitting function based on the feature $\boldsymbol{\lambda}$ and the threshold θ. This optimization is conducted over a finite set and is thus easy to implement. In the case of classification, we can use a computational simplification which tries efficiently all thresholds for a given feature by storing class histograms [Criminisi and Shotton, 2013].



(a) Optimization along $\boldsymbol{\lambda} = 1$



(b) Optimization along $\boldsymbol{\lambda} = 2$

Figure 2.3: **Greedy optimization in the case of axis-aligned splits.** Keeping the setting of Fig. 2.1 and Fig. 2.2, we illustrate the various thresholds tried during the training of the root node, in the case $n_{\text{thresholds}} = 4$ and $\Lambda^{\text{candidates}} = \Lambda$. We retain the most informative split (in (b), in bold) and repeat recursively (see Fig. 2.2)

More Complex Splitting Functions

Although axis-aligned splits are the most frequent choice, more general splitting functions can be used such as hyperplanes [Heath et al., 1993, Menze et al., 2011, Schneider et al., 2015], ellipsoids [Criminisi, 2011, Heinrich and Blendowski, 2016], support vector machines [Bennett and Blue, 1998], boosting classifiers [Tu, 2005] or multi-layered perceptrons [Rota Bulò and Kontschieder, 2014]. These approaches demonstrate in general higher accuracy or more compact trees. However, they are rarely used in practice in comparison to their axis-aligned counterparts, mainly because of the computational difficulties arising during the corresponding node optimization. Handling jointly several features and parameters results indeed in a considerable increase in complexity at training time, so that a greedy optimization is no longer sufficient. This difficulty is often a bottleneck in practical applications. As a partial solution to this problem, a differentiable version of the information gain was introduced [Montillo et al., 2013] and obtained by replacing the usual binary splitting function by a sigmoid. Doing so, if a subset of features is fixed, the gradient with respect to the function parameters (e.g. the hyperplane coefficients in the oblique case) can be computed and a gradient ascent procedure can be performed to find a satisfactory splitting function based on this subset of features.

The simplicity of the greedy node optimization can also be seen as one of the strengths of the decision tree framework, as it is precisely what allows this model to handle large amounts of training data and to perform fast predictions at test time. Introducing a complex decision function at each node loses these two benefits and goes rather towards a hierarchical combination of strong classifiers. Moreover, although it could be argued that axis-aligned splits may be too limited to fit effectively the underlying distribution in some cases, we will see in Sec. 2.2 that an ensemble of axis-aligned decision trees, i.e. a decision forest, offers a greater flexibility.

2.1.5 Stopping Criteria

The training procedure is a recursive process where child nodes are progressively created, and which is eventually stopped resulting in the creation of a leaf. The following stopping criteria can be used:

- A node is automatically turned into a leaf if it reaches a certain predefined depth D. The depth of a node in a tree is the number of edges necessary to reach the root node, which has thus itself a depth 0. Fixing a predefined tree depth can be seen as a computational safeguard as it puts a clear upper bound on the number of nodes which will be created regardless on the amount of training data. It may however not be entirely suitable for applications which require very unbalanced trees.

- A leaf must receive a minimum number of samples $n_{\text{samples/leaf}}$. If, during node split optimization, no split can be found which sends at least $n_{\text{samples/leaf}}$ to both left and right child nodes, no splitting is performed and the node in question is turned into a leaf. Note that if less than $2n_{\text{samples/leaf}}$ samples are sent to a node, it is not even necessary to run the optimization.

- The impurity $G(\mathcal{S})$ of the received set of samples $\mathcal{S}$ is below a certain threshold. In our implementation, we only stop the process if a node reaches complete purity ($G(\mathcal{S}) = 0$), in which case there is no reason to split further.

2.1.6 Leaf Models

At the creation of a leaf L, a local predictive model is computed from the set of incoming training samples $\mathcal{S}^L$. The goal is to obtain, for each possible label $y \in \mathcal{Y}$, the probability $P(Y = y|X \in L)$ that an observation x reaching this leaf has the label y. The nature of the prediction model depends on the learning task. In the classification case where $\mathcal{Y} = \{c_1, \dots, c_K\}$, the leaf probabilistic model is computed from the empirical label distribution, i.e. we have for each $c \in \mathcal{Y}$

$$P(Y = c|X \in L) = \frac{\left|\mathcal{S}_c^L\right|}{|\mathcal{S}^L|}, \tag{2.17}$$

where $\mathcal{S}_c^L$ is the subset of samples of label c in $\mathcal{S}^L$. In the regression case, the continuous aspect of the output space prompts the definition of a density function instead of a discrete probability. This is in general an ill-posed problem. Therefore, assumptions must be made about the model used to estimate this density. A normal distribution can for example be fitted to the arriving data $\mathcal{S}^L$, i.e.

$$f(Y = y|X \in L) = \mathcal{N}(\bar{y}, \mathbf{\Sigma}) \tag{2.18}$$

following the notations introduced in Sec. 2.1.3. In our context of semantic segmentation, we will only consider classification tasks for which the well-posed empirical histogram estimation given by Eq. 2.17 is the most natural choice.

2.1.7 Handling Class Imbalance

In a classification context, the available training data often presents an imbalance in terms of the represented labels. This problem of class imbalance is particularly present in the context of semantic image segmentation: taking the example of labeled road scenes, it is expected to see many more `Sky` or `Road` pixels than `Bicycle` or `Sign symbol`. If the tree training algorithm described above is applied on such a case without modification, this class imbalance may result in over-predicting frequent classes and under-predicting rare classes. To understand this phenomenon intuitively, consider a simple fictive tree consisting only of one root node L_{root} (which is, in fact, a root leaf!). Following Eq. 2.17, the class histogram stored at this leaf exactly represents the class distribution in the training set and therefore leads to a systematic prediction of the most frequent class. Is this what we desire for the prediction model stored at this leaf?

The answer to this question is not straightforward and is related to the performance measure that one seeks to optimize. If the labeled data is fairly representative of future data to come, it is, from an accuracy point of view (i.e. the proportion of observations correctly classified), beneficial to predict the most frequent class. For example, if 70% of observations have `Sky` as true label, an agnostic classifier (as is a tree made of only one leaf) predicting systematically `Sky` without even looking at the image content reaches an accuracy of 70%. However, the proportion of pixels correctly classified is not always an ideal measure for segmentation task since rare instances are often as important, if not more, than frequent ones. For instance, we would prefer a road scene understanding algorithm incorporated in a self-driving car to exchange a few misclassified `Sky` pixels against the correct identification of a pedestrian or a bicycle. This is the reason which motivates the introduction of alternative measures of the quality of a semantic labeling, such as the average classwise Recall or Jaccard index (Sec. 1.1.2), which treat all *classes* as equally important instead of all *instances* (pixels, in our case).

Therefore, we could expect our one-leaf-tree to 'abstain' without further information, i.e. to output an equal probability for all classes. This is achieved by reweighting observations to simulate the duplication of the training data so that all classes are equally sampled. We denote $|\mathcal{S}^c_{\text{train}}|$ the number of observations of label c in the training data $\mathcal{S}_{\text{train}}$ sent to the tree and we define the balancing weight of the class c as

$$\nu_c = \frac{|\mathcal{S}_{\text{train}}|}{K\,|\mathcal{S}^c_{\text{train}}|} \tag{2.19}$$

where K is the number of possible labels. We assume here that at least one training sample of each label is available (otherwise, any arbitrary finite value can be used for the weight of such an absent class without consequences). Applying the weight ν_c to each individual training instance of label c provides a new (real-valued) counting operator $|.|_{\text{bal}}$ taking into account the class imbalance and simulating the duplication of training data to achieve equal label proportions. For any subset $\mathcal{S} \subseteq \mathcal{S}_{\text{train}}$, we have

$$|\mathcal{S}^c|_{\text{bal}} = \nu_c\,|\mathcal{S}^c| \tag{2.20}$$

and

$$|\mathcal{S}|_{\text{bal}} = \sum_{c\in\mathcal{Y}} |\mathcal{S}^c|_{\text{bal}} = \sum_{c\in\mathcal{Y}} \nu_c\,|\mathcal{S}^c|\,. \tag{2.21}$$

To compensate class imbalance in the tree model, one now substitutes the operator $|.|_{\text{bal}}$ to the classical counting operator $|.|$ in the training procedure, i.e. in:

- The class proportion p_c involved in the computation of the Gini index or the entropy (Eq. 2.7 and Eq. 2.8) and, similarly, in the computation of the leaf models (Eq. 2.17).
- The ratios weighting $G(\mathcal{S}_{\psi,\text{left}})$ and $G(\mathcal{S}_{\psi,\text{right}})$ in the purity gain (Eq. 2.5).
- If applicable, when counting instances to see if a stopping criterion based on a minimum number of samples per leaf is reached.

To conclude, we can verify some properties of this new counting operator. First, we can note that the number of training samples sent to the tree is kept unchanged:

$$|\mathcal{S}_{\text{train}}|_{\text{bal}} = \sum_{c\in\mathcal{Y}} \nu_c \, |\mathcal{S}^c_{\text{train}}| = \sum_{c\in\mathcal{Y}} \frac{|\mathcal{S}_{\text{train}}|}{K\,|\mathcal{S}^c_{\text{train}}|} \, |\mathcal{S}^c_{\text{train}}| = \sum_{c\in\mathcal{Y}} \frac{|\mathcal{S}_{\text{train}}|}{K} = |\mathcal{S}_{\text{train}}| \, . \tag{2.22}$$

This means, in particular, that a stopping criterion based on a minimum number of samples per leaf still corresponds to the same proportion of the training data sent to the tree. Finally, using this property, we can also compute the weighted predictive model of our fictive tree made of one leaf:

$$P(Y=c) = \frac{|\mathcal{S}^c_{\text{train}}|_{\text{bal}}}{|\mathcal{S}_{\text{train}}|_{\text{bal}}} = \frac{\nu_c \, |\mathcal{S}^c_{\text{train}}|}{|\mathcal{S}_{\text{train}}|} = \frac{1}{K}, \tag{2.23}$$

which is the expected abstaining prediction of this classifier that now handles classes in an egalitarian way.

2.1.8 Overfitting Behavior

If allowed to be grown deep enough and equipped with a sufficiently effective node optimization algorithm, one can find for any separable set of samples a tree for which each leaf contains exactly one of these samples. In such a case, similarly to our discussion on the 1-nearest neighbor classifier (Sec. 1.2.4), the final levels of the tree are most likely to be fitting the specific training data instead of the underlying distribution, hence leading to an overfitting situation (Fig. 2.4). Using the vocabulary of Sec. 1.2.5, decision trees are typically a model with low bias since they are flexible and can fit complex distributions, but with high variance as the learned tree structure can be different if a new training data is resampled from the same distribution. To reduce the complexity of the model and thereby the overfitting behavior, one could force early stopping with appropriate stopping criteria, for instance by limiting the tree depth. Additionally, pruning methods were developed, i.e. the operation of replacing lower parts of the tree by leaves to achieve a compromise between accuracy and complexity of the model [Breiman et al., 1984, Mingers, 1989, Quinlan, 1993, Mansour, 1997]. Unlike early stopping methods, the full tree has to be grown first, which is computationally costlier but does not require to know about the required model complexity before training. However, pruning methods usually require a

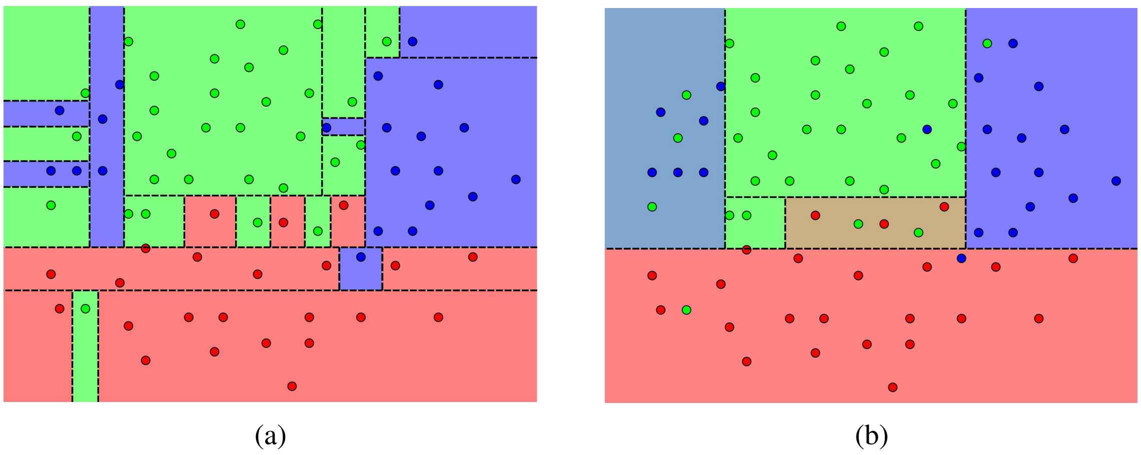


Figure 2.4: **Decision trees and overfitting.** (a) If allowed to grow infinitely, decision trees can fit exactly all the (separable) instances of the training set, which leads to unnatural and undesired decision boundaries. (b) A solution to this problem is to prune the tree, i.e. to keep only a subtree of the grown tree which generalizes better on a validation set. However, in addition to the need of these additional labeled data, the result may still lack of smoothness. Random forests offer an alternative without these shortcomings.

separate validation set to find the best pruning level, which forces to reduce the size of the training data and can thus be a limitation, especially if training data are difficult to obtain.

A more modern approach to counter overfitting without the need for a validation set is to follow the bootstrap aggregating (bagging) strategy presented in Sec. 1.2.7, i.e. to combine several randomized trees performing slightly different decisions. This is the framework of random forests.

2.2 Random Forests

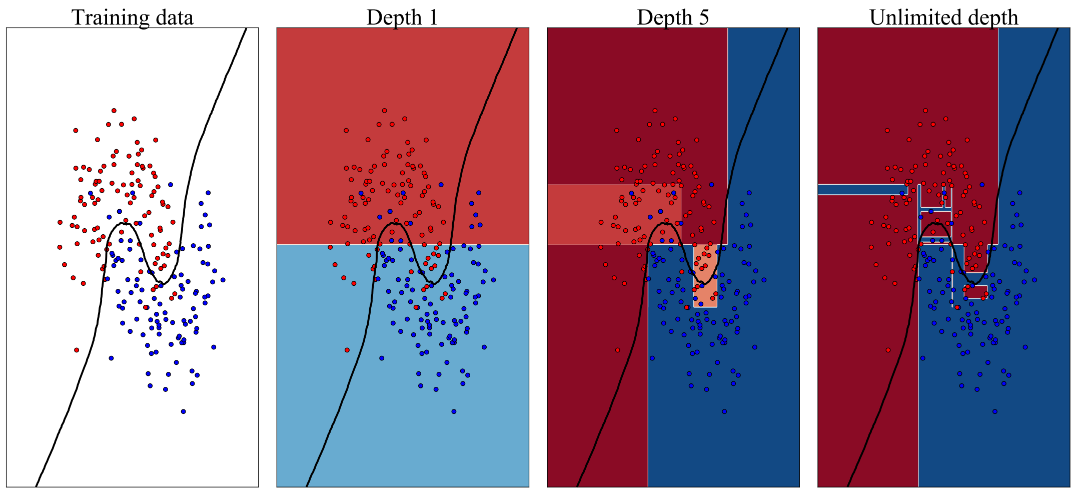
A random forest reduces the natural overfitting behavior of a decision tree by creating several decision trees, all sightly different from another, and merging their predictions. If n_{trees} trees are created, the final forest estimate is usually obtained by averaging each individual tree decision, i.e. for every $y \in \mathcal{Y}$,

$$P(Y = y|X = x) = \frac{1}{n_{\text{trees}}} \sum_{t=1}^{n_{\text{trees}}} P_t(Y = y|X = x) \tag{2.24}$$

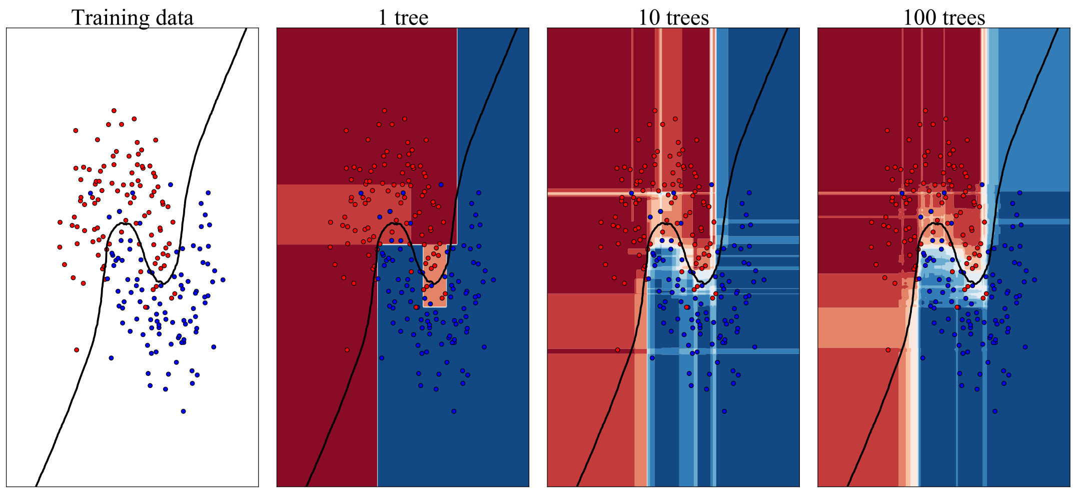
where $P_t(Y = y|X = x)$ is the prediction of the t^{th} tree. The creation of different yet related trees is achieved by randomizing the training of each tree. We illustrate in Fig. 2.5 the behavior of random forests on the toy example introduced in Fig. 1.2.

2.2.1 Injecting Randomization in Training

Several random forest ensembles have been introduced historically by using different randomization techniques. They can be classified into two main categories: either sending



(a) One tree of varying depth



(b) Varying number of trees of depth 5

Figure 2.5: **Illustration of decision forests on a toy example.** (a) We consider one tree of varying depth. The overfitting behavior with unlimited depth can be observed, and we can also notice a general lack of smoothness in the decision rule. (b) We consider an ensemble of several randomized trees of depth 5 with a combination of bagging and randomized node optimization. The ensemble prediction is smoother when the number of trees increases.

different training data to each tree in a bagging fashion, or weakening the optimization of node splitting functions (Eq. 2.6) during training so that two trees receiving the same data would eventually have different structures.

Bagging

As described in Sec. 1.2.7, bagging predictors allows to create an ensemble of classifiers of lower variance than the individual classifiers composing the ensemble. In the context of random forests, this procedure is conducted by sending to each tree a set of labeled instances which has been resampled from the initial training data. In the original approach, each tree receives a training set of the same size as the initial one by sampling training instances with replacement [Breiman, 1996]. In the context of large-scale data like semantic segmentation where millions of training samples are available, a subsampling approach is more appropriate due to computational constraints, i.e. each tree receives a randomly drawn subset smaller than the training data. By doing so, the computational cost of the training procedure is reduced while leaving each training sample still accessible in theory.

Another advantage of bagging is the possible use of out-of-bag estimates. Since each tree is trained only on a subset of the training data, the rest of the training data which has not been seen by a tree can act as a validation set for this particular tree. Eventually, we can define for each point of the training set an out-of-bag prediction, which is obtained by averaging the predictions of the trees that have not been trained on this sample. The generalization ability of the forest can thus be assessed as if a validation set was available, but without the necessity of leaving out a part of the training data. However, the independence between training samples is crucial for these estimates not to be too optimistic. In the context of pixels within an image, this assumption is rather invalid and, perhaps for this reason, out-of-bag estimates are not frequently encountered in the case of semantic segmentation, although they have been employed in other computer vision applications [Leistner et al., 2009, Saffari et al., 2009].

Randomized Split Optimization

Another way to build decorrelated trees consists in randomizing the split node optimization (Eq. 2.6) during training. Several strategies have been proposed: the random subspace method [Ho, 1998] allows each tree to access only a subset of the feature maps $(\phi_{\boldsymbol{\lambda}})_{\boldsymbol{\lambda} \in \Lambda}$. Another approach proposes to search for the n best splitting functions instead of the best one (with n arbitrarily chosen), and picks at random within these n functions the one that will be retained [Dietterich, 2000]. However, the most popular choice consists in performing the split node optimization by randomly sampling uniformly a subset of features at each node, as we have already mentioned in Sec. 2.1.4. This technique was first introduced out of practical reason due to a too complex set of splitting functions [Amit and Geman, 1997] and later employed as a way to reduce overfitting [Breiman, 2001]. Interestingly, similarly to bagging, such a randomization is necessary for practical reasons in a large set of features such as representations based on Haar features (Sec. 1.3.1). Finally, extremely randomized trees [Geurts et al., 2006] randomize in addition the threshold selection in the case of axis-aligned splits.

2.2.2 Advantages and Limitations of Random Forests

We conclude our description of random forests with a summary of their strengths and weaknesses as a learning technique. The advantages of random forests include:

- Their computational efficiency. Following a divide-and-conquer strategy, random forests scale well to large amounts of training data and can be easily parallelized over trees or even implemented on a GPU [Sharp, 2008] for both training and prediction.
- Their compatibility with large feature representations and relative robustness to irrelevant features, since a non informative feature would not be retained at a node during training. Moreover, only the features stored in the nodes must be computed at prediction time which is an additional computational advantage.
- The fact that, in the classification case, multi-class problems are naturally handled via the definition of the impurity measure and the histogram-based models stored in the leaves.
- The compatibility of axis-aligned decision trees with heterogeneous features, as already discussed in Sec. 2.1.1.
- The availability of out-of-bag predictions which provide an estimate of the generalization abilities of the model without requiring an independent validation set.

In particular, the multi-class aspect and the computational efficiency for both training and prediction make random forests a very appropriate method for semantic image segmentation, as will be discussed in details in Sec. 2.3. Regarding the limitations of the model, we can mention that:

- Although progresses have been recently made [Biau and Scornet, 2016], the effectiveness of random forests is not very well understood from a theoretical point of view.
- The predictions of a random forest are not as interpretable as for a single decision tree due to the ensemble nature of the model.
- They do not possess the full ability of learning efficiently feature representations together with the decision rule, as is for instance possible with convolutional neural networks. The fact that random forests are compatible with large feature representations mitigates this effect but they still require an explicitly defined feature space over which the greedy node optimization is able to find informative features in a reasonable amount of time during training.

Building on the standard framework, research works introduced variants of the training and prediction steps as well as other types of tree ensembles to overcome the aforementioned limitations. We propose an overview of such approaches in the next section.

2.2.3 Variants and Extensions

Several works propose to remove the independence between trees and to exploit the complementary nature of the trees more effectively than by simply averaging their predictions. Alternating Decision Forests [Schulter et al., 2013] design the forest training as a global loss minimization problem and grow trees jointly stage by stage by applying a boosting-inspired reweighting of training instances every time the trees gain a depth level. This allows a joint learning of all trees without losing the possibility of parallel computing. Instead of introducing the global loss minimization in the training in this fashion, a pre-trained forest can also be refined by re-learning (and pruning) the leaf nodes in a global manner according to the loss function [Ren et al., 2015]. A faster classification without re-training can also be achieved by identifying observations that are easy to predict and only predicting them with a few trees, while a larger number of trees focus on the challenging cases [Schwing et al., 2011]. Instead of growing a tree in the standard greedy recursive fashion, a recent approach globally optimizes the splitting functions and the leaf nodes for a tree of fixed depth [Norouzi et al., 2015]. Deep neural decision forests [Kontschieder et al., 2015] propose a similar global tree optimization where feature representation, splitting nodes and leaf models are jointly optimized.

In all these approaches, the final predictor remains a decision forest. We mention in this last paragraph other kinds of tree ensembles. Instead of bagging trees to reduce the variance of the ensemble, one can grow smaller trees and combine them with gradient boosting to reduce, this time, the bias of the ensemble [Hastie et al., 2009]. Random ferns [Ozuysal et al., 2007] can be seen as a special case of random forests, where the same splitting function is used at all nodes located at a same depth of the tree, which offers great computational simplifications. Finally, Decision Jungles [Shotton et al., 2013b] extend the tree structure to more general directed acyclic graphs, where a (non-root) internal node has two child nodes like in a tree but is allowed to have more than one parent node. The structure of the graph is then jointly optimized with the splitting functions during training, which furnishes more compact models without computational loss at prediction time.

2.3 Random Forests for Semantic Image Segmentation

For the task of semantic image segmentation, a decision must be made for each pixel or voxel in an image. Since a single 2D image typically contains at least 10^5 pixels and a 3D volume contains several millions of voxels, both the learning and prediction tasks are computationally demanding and involve very large sets of observations. For this reason, and together with their natural ability to handle multi-class problems, tree structures and especially random forests appeared as a natural and popular choice for semantic image segmentation. In this section, away from the more general models introduced so far, we review specifically the existing forest approaches that are tailored to the image labeling task and the associated challenges. We first discuss usual choices of feature representations for forest-based segmentation (Sec. 2.3.1) before reviewing works incorporating structured and prediction abilities in the random forest framework (Sec. 2.3.2). Finally, we expose in more details the tradeoff between the local and global modeling of the visual context around a pixel and illustrate its implications (Sec. 2.3.3). After having described the various strategies introduced in the literature to tackle this problem, we propose our own contribution towards this objective in Sec. 2.4.

2.3.1 Choice of Feature Representation

The design of a supervised learning method often starts by considering how training and testing instances will be quantitatively modeled, i.e. with the definition of feature maps in our formulation. We have presented in Sec. 1.3.1 a few examples of possible feature representations if observations consist of pixels within an image, as is the case in the segmentation scenario. The most traditional approach extracts only a relatively small number of features and is still used in recent forest-based works. For instance, in computer vision, Kontschieder et al. [2014] use HOG-like features as well as local first and second order derivatives, and Ravì et al. [2016] introduced new features based on the discrete cosine transform for generic 2D semantic segmentation tasks. For the labeling of hand parts, Zhu et al. [2015] employ color and gradient-based features per pixel. Application-specific features can also be designed, such as features based from the properties of the ultrasound physics for tissue characterization in ultrasound imaging [Conjeti et al., 2016]. In general, small spaces of possibly application-oriented features present some advantages like an easier interpretation of the prediction and behavior of the model or the use of dimensionality reduction techniques [Conjeti et al., 2016]. However, a computational tradeoff is required when the number of features increases, especially at training time: either one computes and stores all feature responses in memory before training, which is then limited by the amount of memory available, or one recomputes them individually at each node which results in slower training. This may not scale well with the large amount of training samples available for segmentation tasks. In some cases, the feature design allows mathematical shortcuts. For instance, Schneider et al. [2015] use steerable filters in a random forest for the segmentation of vessels in 3D images. By precomputing the response of a set of basis filters only, the feature response can then be computed at any arbitrary orientation, emulating a theoretically infinite set of features which the random forest framework can effectively handle.

Going in the same direction, since their first use within the random forest framework under the form of semantic textons [Shotton et al., 2008], some variants of the Haar features presented in Sec. 1.3.1 have been used in a large number of forest-based approaches for semantic segmentation. This choice is both due to their computational advantages and the fact that they are generic enough to be applied to diverse segmentation tasks. They can be combined with other pixelwise handcrafted features by storing these visual cues as artificial color channels [Schroff et al., 2008, Rota Bulò and Kontschieder, 2014, Ravì et al., 2016], offering thereby a contextual layer on top of these pixelwise descriptors.

Beyond 2D computer vision applications such as human pose estimation from depth images [Shotton et al., 2013a], Haar-like features enjoy especially a huge popularity in the medical field where they provide a tractable and accurate way to perform segmentation of 3D volumes. In fact, they are present in the majority of works involving forest-based segmentation in 3D images. A non exhaustive list of successful applications of these features include organ sementation in CT volumes [Yi et al., 2009, Montillo et al., 2011, Glocker et al., 2012, Lindner et al., 2013, Heinrich and Blendowski, 2016], segmentation of echogenicities [Pauly et al., 2012] and of the midbrain [Chatelain et al., 2013] in 3D ultrasound data, delimitation of heart anatomy in 3D [Lempitsky et al., 2009b] and even 4D [Margeta et al., 2011] data, segmentation of multiple sclerosis lesions [Geremia et al., 2011] and tumors [Zikic et al., 2012] in MR brain images. Beyond segmentation, they were also used for instance for the localization of organs [Criminisi et al., 2009, Pauly et al., 2011, Ebner et al., 2014, Gauriau et al., 2015]).

2.3.2 Structured Learning

Seeing the segmentation objective as a pixelwise classification task is a simple strategy which is compatible with traditional supervised learning techniques. However, as discussed in Sec. 1.4, the fact that both inputs and outputs are in fact images could also be used to increase the robustness of the predictions. We review here contributions which introduce structure in the forest framework.

For multi-organ segmentation within 3D CT scans, Glocker et al. [2012] regularize the training of the classification by adding a regression objective aiming at predicting the distance of a voxel to each organ of interest in addition to its discrete label. The regression task is incorporated in the training via a joint information gain for classification and regression. By tackling these several tasks simultaneously, the retained features are expected to generalize better which brings this strategy close to a multi-task learning scenario [Caruana, 1997]. In a similar idea, the 3D displacement of an active contour was recently jointly learned together with the voxel label [Gao et al., 2016].

Several approaches are inspired from the idea of auto-context [Tu and Bai, 2010] already mentioned in Sec. 1.4. In the context of forests, we can mention the work of Zikic et al. [2012] who first trained a simple (thus non-overfitting) Gaussian Mixture Models classifier on MR data for brain tumor segmentation, before feeding its output to a forest equipped with Haar-like features. In this case, this two-step process can also be seen as learning a relevant feature which is then added as additional modality. Closer from the original auto-context algorithm, cascaded random forests have been recently proposed where the output of each forest is regularized by a conditional random field before being sent to the next forest training [Richmond et al., 2015]. Other approaches include, during node optimization, a direct access to the predictions of the version of the tree grown so far: for multi-organ segmentation, Montillo et al. [2011] allow interaction between nodes during training while Kontschieder et al. [2013] presented geodesic forests to perform visually-driven regularizations between different regions of an image.

The fact that training and testing instances are collected within a set of images has also led to a few approaches putting a stronger emphasis on the notion of image in training and testing sets. For the semantic labeling of brain anatomy, atlas forests train one small forest on each training volume [Zikic et al., 2014], acting as a deterministic bagging and offering some advantages in terms of design, such as the easy addition of new training volumes. For this application where a large amount of structure is available and where training volumes are only slightly different from another, this approach proved to be at least as accurate as a standard forest and much faster than registration-based label propagation. Another work going in a similar direction performs first a clustering of training volumes with Laplacian eigenmaps before training specialized trees on local areas of this representation [Lombaert et al., 2014], i.e on some subsets of volumes sharing visual similarities. At prediction time, a test volume is first placed on the Laplacian eigenmap and trees that have been trained on similar volumes are given a higher weight in the prediction. Thereby, heterogeneous datasets such as medical volumes coming from different hospitals can be used more effectively for training.

Finally, research efforts have tried to output labels that are globally consistent within an image, e.g. to avoid predictions of non-realistic label configurations. The forest output can for instance be used as a unary probability in a conditional random field model [Kluckner et al., 2009a,b], possibly by learning additional contextual information in addition to the pixel label [Payet and Todorovic, 2013]. The parameters of a CRF have also been jointly learned with decision trees [Nowozin et al., 2011]. Yang et al. [2012] introduced so-called local label descriptors which infer, from the forest classification output, a pixelwise labeling satisfying constraints on an expected label histograms at each location. Structured prediction abilities have also been included directly in the forest framework by learning and predicting at patch level, i.e. estimating jointly the label of a pixel and of its neighbors [Kontschieder et al., 2014]. A similar idea was introduced for the semantic labeling of hand parts where shape masks are stored in leaf nodes [Zhu et al., 2015]. In the medical field, Neighborhood Approximation Forests [Konukoglu et al., 2013] similarly perform a generic clustering of medical images from the visual content according to some meta-input and were applied to a patchwise semantic segmentation task.

2.3.3 A Matter of Context

The visual information which is useful to predict the label of a pixel can be located at different scales around it. For instance, if a human is given a medical body scan and asked to segment a particular organ, the intuitive approach would probably be to first use global information to localize the organ of interest within the body and, once the organ has been found, to use the local image intensity information to delineate precisely its contour. This aspect motivates structured learning techniques, as one would like to output a segmentation that is realistic and consistent, both at local level and when considering the global arrangement of a body scan or a scene. However, we could also argue that, if equipped with a sufficiently rich feature representation which includes information about the whole surrounding image, a pixelwise segmentation approach should be able to integrate information located at various scales.

In practice, a standard forest having access to information at all visual ranges does not show the promised results. We illustrate this in Fig. 2.6 by considering a standard training with Haar features (Sec. 1.3.1) where we let the maximum scale δ vary. For small values of δ, i.e. when too little visual context is present, local decisions are quite accurate with clearly defined edges, but the unavailability of global information leads to unrealistic predictions with a clear lack of smoothness. There is nothing surprising at this stage, as even a human would often misclassify a pixel if the given contextual information around it is too narrow. As we increase the size δ of the neighborhood, we can see that the global information starts being taken into account leading to better arrangements, but at the cost of missing local structures. It is, here, slightly less expected. Increasing δ provides a larger set of features without destroying any information so that the very local features used at short scale are still perfectly accessible to the learner. However, the greedy optimization used at each node is no longer able to find relevant features: by sampling uniformly a few hundreds of Haar-like features, it becomes too unlikely to find the ones that complement the information that has been learned earlier in the tree. Therefore, in a standard forest setting, the chosen value for δ is imposed by this tradeoff and changing the training algorithm is required to effectively combine local and global information.

A first strategy consists in training different forests at each scale level δ and combining their posterior probabilities [Pauly et al., 2012, Lay et al., 2013, Ebner et al., 2014, Gauriau et al., 2015] e.g by multiplying them. Other approaches keep a small scale for the features, ensuring that the local properties of the pixel of interest are captured, and introduce additional features (e.g as artificial color channels) that are designed to incorporate long-range information. To do so, the perhaps most intuitive solution is to use the pixel coordinate as features [Lempitsky et al., 2009b], but a clearly defined coordinate system and aligned images are then required, as translation invariance is lost in the process. A more sophisticated approach for strongly structured applications such as brain imaging is to register a label prior to the test data and use this label prior as color channel [Zikic et al., 2014], or to use as features spectral coordinates instead of standard Euclidean coordinates [Lombaert et al., 2015]. If the object of interest is not necessarily expected at a particular location, these strategies cannot be employed. For the segmentation of multiple sclerosis lesions in 3D brain images, Geremia et al. [2011] introduced instead an additional feature comparing the pixel of interest with its symmetric counterpart with respect to the mid-sagittal plane. This exploits the fact that, unlike healthy parts of the brain, lesions develop in an asymmetric way.

Without introducing new features, the node optimization itself could be guided. In a large feature space, weak and strong features could be learned beforehand and used for sampling candidate features in the training of the forest [Montillo et al., 2011, Ye et al., 2013], possibly also depending on the depth of the node in question [Yaqub et al., 2014]. The limitation of a preliminary feature learning is the fact that it is global and not made dependent on the current stage of the training, although the relevance of a feature depends on the set of samples sent to a node. By learning once for all which features are weak and strong, their redundancy with what has already been learned earlier in the tree is not taken into account. Finally, Geremia et al. [2013] explicitly create a hierarchy of supervoxels and refine the representation when necessary during the forest training.

2.4 Scale-Adaptive Forest Training

In spite of their advantages, the aforementioned approaches aiming at combining local and global contextual information present a computational overhead at training or testing time. Moreover, some of them raise other practical issues such as having to choose explicitly

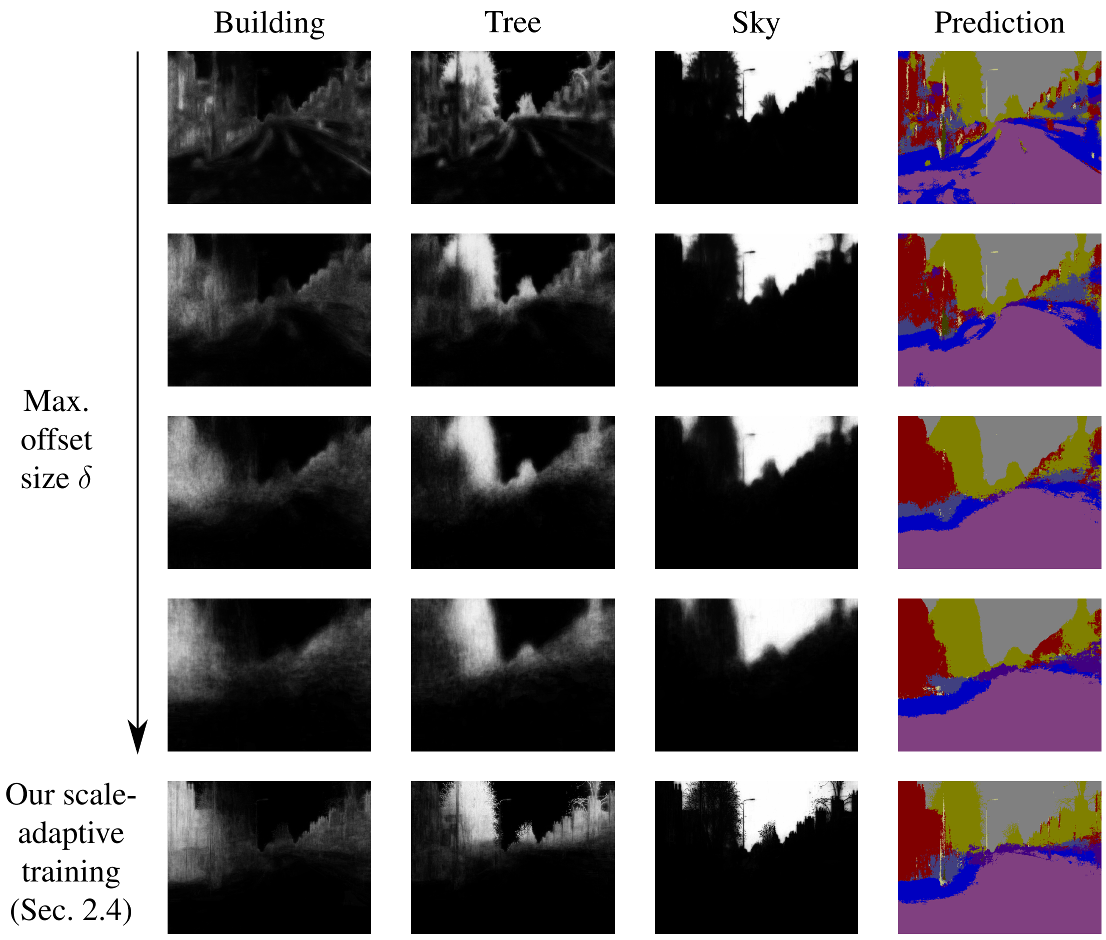

Building
Tree
Sky
Prediction
Max. offset size δ
Our scale-adaptive training (Sec. 2.4)

Figure 2.6: **How far should the visual information around a pixel be extracted?** We illustrate qualitatively on the example image shown in Fig. 1.1a the influence of the maximum scale δ at which the visual context around a pixel is extracted. For small values of δ, the posterior probabilities contain fine details (the column pole is successfully excluded from the sky) but lack global information to correctly predict visually complex classes such as buildings. As δ increases, the posteriors become smoother and more aware of the global arrangement of the scene but lack the finer details. We introduce in Sec. 2.4 an alternative training scheme which achieves a compromise between local and global contextual information without adding any computational cost.

the different scales to combine. In this section, we introduce an alternative Haar feature sampling scheme during training which:

- Does not present any computational overhead at both training and testing time in comparison to the standard training.
- Is very easy to implement.
- Does not require the tuning of any additional hyperparameter.
- Demonstrates clear segmentation improvements on 4 diverse datasets.

Our method is evaluated on both 2D computer vision images and 3D medical volumes. In Sec. 2.4.1, we expose in details our choice of feature representation in these two cases. We then introduce in Sec. 2.4.2 our scale-adaptive node optimization strategy and report a series of experimental results in Sec. 2.4.3 demonstrating its advantage in comparison to the standard strategy.

2.4.1 Visual Features

Our forest-based segmentation method can be applied to both 2D and 3D images. The only aspect for which 2D and 3D cases must be treated separately is the design of the feature representation which encodes the visual information available for classifying each pixel or voxel. Following the practice of most works using forests for segmentation, we use in the two cases a Haar feature representation. In this section, we describe our features in details and follow the notations used in Sec. 1.3.1.

For 2D images, we extend the classical Haar features by including both color and gradient information. Merging Eq. 1.28 and Eq. 1.29, we parametrize our 2D feature maps $\left(\phi^{\text{2D}}_{\boldsymbol{\lambda}}\right)_{\boldsymbol{\lambda}\in\Lambda}$ by a vector

$$\boldsymbol{\lambda} = (\underbrace{\vec{v_1}, \vec{v_2}, \mathbf{s}_1, \mathbf{s}_2}_{\text{scale-related}}, \underbrace{c_1, c_2, k_1, k_2, \tau, \omega}_{\text{categorical}}), \tag{2.25}$$

with $\vec{v_1}, \vec{v_2} \in \{-\delta, \ldots, \delta\}^2$, $\mathbf{s}_1, \mathbf{s}_2 \in \{1, \ldots, \delta\}^2$, $c_1, c_2 \in \{1, \ldots, n_{\text{channels}}\}$, $k_1, k_2 \in \{1, \ldots, B\}$, $\tau \in \{\texttt{Color}, \texttt{Gradient}\}$, and $\omega : \mathbb{R}^2 \to \mathbb{R}$. The novelty in comparison to the features described in Sec. 1.3.1 is the introduction of a parameter τ which encodes the type of information that is extracted:

- If $\tau = \texttt{Color}$, the feature response $\phi^{\text{2D}}_{\boldsymbol{\lambda}}(x)$ is the color-based Haar feature defined in Sec. 1.3.1, i.e. $\phi^{\text{2D}}_{\boldsymbol{\lambda}}(x) = \phi^{\text{Haar}}_{\boldsymbol{\lambda}'}(x)$ with $\boldsymbol{\lambda}' = (\vec{v_1}, \vec{v_2}, \mathbf{s}_1, \mathbf{s}_2, c_1, c_2, \omega)$. We remind that this feature extracts the mean intensity witin each box over their respective color channel and combines them via the operation ω.

- If $\tau = \texttt{Gradient}$, we extract a HoG feature in each box and combine them with ω, i.e. $\phi^{\text{2D}}_{\boldsymbol{\lambda}}(x) = \omega\left(\phi^{\text{HoG}}_{\boldsymbol{\lambda}'_1}(x), \phi^{\text{HoG}}_{\boldsymbol{\lambda}'_2}(x)\right)$ where $\boldsymbol{\lambda}'_i = (\vec{v_i}, \mathbf{s}_i, c_i, k_i)$ for $i \in \{1, 2\}$.

The chosen possible operations ω are the sum, the difference and the absolute value of the difference [Shotton et al., 2008]. We illustrate the 2D feature extraction process on Fig. 2.7. The distinction in Eq. 2.25 between scale-related and categorical parameters will play a role in our sampling scheme exposed in Sec. 2.4.2.

In the 3D case, we do not consider gradient-based features due to memory constraints. Indeed, since a direction in 3D is characterized by two angles, an equivalent 3D-histogram would contain B^2 bins instead of B bins. Since, moreover, we would have to store an integral *volume* for each of these B^2 bins, the required memory is too large for such an approach to be tractable. Our 3D features are thus classical Haar features in 3D, defined by

$$\boldsymbol{\lambda} = (\underbrace{\vec{v_1}, \vec{v_2}, \mathbf{s}_1, \mathbf{s}_2}_{\text{scale-related}}, \underbrace{c_1, c_2, \omega}_{\text{categorical}}), \tag{2.26}$$

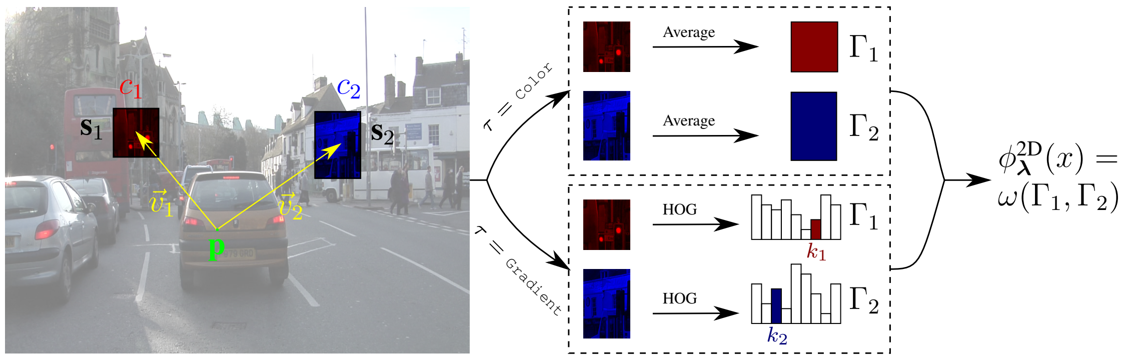


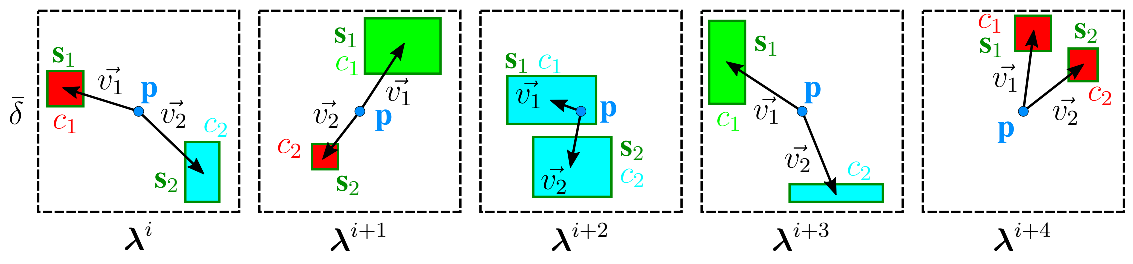
Figure 2.7: **Extended Haar features in 2D.** Our visual features are defined by a parameter vector $\boldsymbol{\lambda} = (\vec{v_1}, \vec{v_2}, \boldsymbol{s}_1, \boldsymbol{s}_2, c_1, c_2, k_1, k_2, \tau, \omega)$. At a location $\mathbf{p}$, two quantities Γ_1, Γ_2 are extracted, each Γ_i stating the visual content in the box $\mathcal{B}(\mathbf{p}+\vec{v_i}, \boldsymbol{s}_i)$ centered on $\mathbf{p}+\vec{v_i}$ and of size $\boldsymbol{s}_i = (s_{i,x}, s_{i,y})$ within the color channel c_i. The type of information encoded by Γ_i is defined by a parameter $\tau \in \{\texttt{Color}, \texttt{Gradient}\}$ which respectively corresponds to the mean intensity and to the value of the k_i^{th} bin of the histogram of oriented gradients (Sec. 1.3.1). Finally, the final feature value $\phi_{\boldsymbol{\lambda}}^{2D}(x)$ of the observation $x = (\mathbf{p}, I)$ is obtained by combining the two quantities Γ_1 and Γ_2 via a function ω. The image on the left is taken from the CamVid dataset [Brostow et al., 2008a].

with $\vec{v_1}, \vec{v_2} \in \{-\delta, \dots, \delta\}^3$, $\mathbf{s}_1, \mathbf{s}_2 \in \{1, \dots, \delta\}^3$, $c_1, c_2 \in \{1, \dots, n_{\text{channels}}\}$ and $\omega : \mathbb{R}^2 \to \mathbb{R}$. The feature response is simply the combination of the mean intensities in each box, i.e. $\phi_{\boldsymbol{\lambda}}^{3D}(x) = \phi_{\boldsymbol{\lambda}}^{\text{Haar}}(x)$. In our experiments, two 3D medical datasets from MR and ultrasound imaging respectively will be used to validate our approach. For these imaging modalities, the individual image value observed at each voxel is not necessarily consistent between images with changes in terms of illumination and contrast. Therefore, similarly to existing approaches [Pauly et al., 2011, Chatelain et al., 2013, Heinrich and Blendowski, 2016], we only allow ω to be the binarized difference between two boxes, i.e. $\omega(a, b) = 0$ if $a \leq b$ and 1 otherwise. With this operation, the visual context around a voxel is reduced to simple comparisons of the average intensity between two boxes.

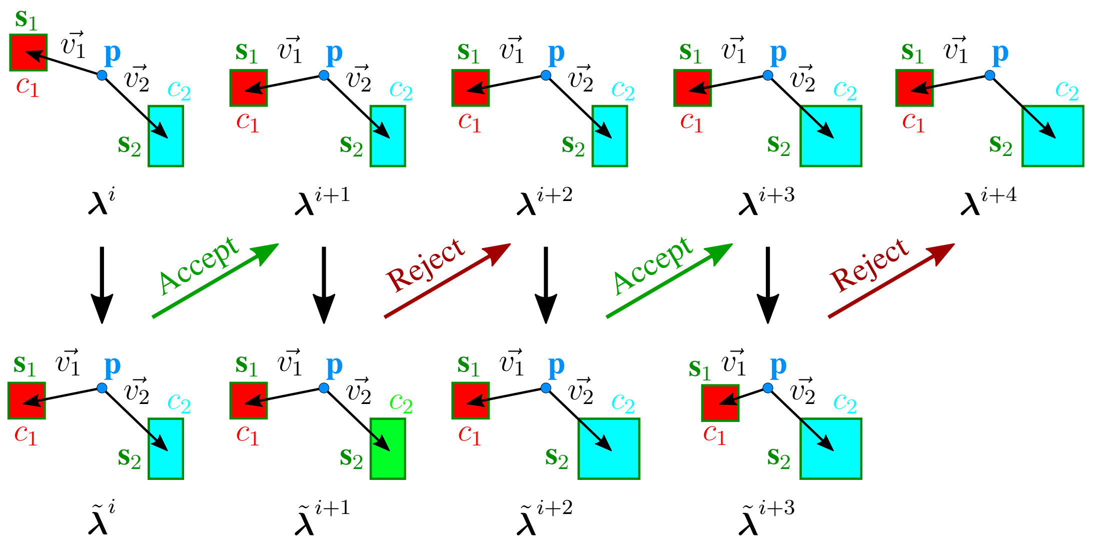
2.4.2 Scale-Adaptive Node Optimization

In the standard forest training framework, n_{tries} candidate features $\boldsymbol{\lambda}^1, \dots, \boldsymbol{\lambda}^{n_{\text{tries}}}$ are drawn uniformly and independently at each node to form the set of candidate features $\Lambda^{\text{candidates}}$ over which the node optimization is performed. Since each $\boldsymbol{\lambda}^i = (\lambda_1^i, \dots, \lambda_D^i)$ is a vector of dimension denoted D here, this is practically achieved by sampling each coordinate $\lambda_d^i, 1 \leq d \leq D$ of $\boldsymbol{\lambda}^i$ uniformly over its predefined set of possible values Λ_d. In particular, the range of scale-related parameters defined in Eq. 2.25 and Eq. 2.26 is proportional to a predefined value δ encoding the maximum scale at which the visual context can be extracted. As discussed in Sec. 2.3.3, deciding on an appropriate value of δ is usually problematic as it has a strong impact on the forest prediction. In this section, we expose an alternative sampling scheme which alleviates this difficulty at no additional cost.

Instead of sampling the $\boldsymbol{\lambda}^i$ independently, we proceed sequentially by letting each


$\bar{\delta}$
$\mathbf{s}_1$
$\vec{v}_1$
$\mathbf{p}$
$\vec{v}_2$
c_1
c_2
$\mathbf{s}_2$
$\boldsymbol{\lambda}^i$
$\boldsymbol{\lambda}^{i+1}$
$\boldsymbol{\lambda}^{i+2}$
$\boldsymbol{\lambda}^{i+3}$
$\boldsymbol{\lambda}^{i+4}$

(a) Standard feature sampling


$\mathbf{s}_1$
$\vec{v}_1$
$\mathbf{p}$
$\vec{v}_2$
c_1
c_2
$\mathbf{s}_2$
$\boldsymbol{\lambda}^i$
$\boldsymbol{\lambda}^{i+1}$
$\boldsymbol{\lambda}^{i+2}$
$\boldsymbol{\lambda}^{i+3}$
$\boldsymbol{\lambda}^{i+4}$
Accept
Reject
Accept
Reject
$\tilde{\boldsymbol{\lambda}}^i$
$\tilde{\boldsymbol{\lambda}}^{i+1}$
$\tilde{\boldsymbol{\lambda}}^{i+2}$
$\tilde{\boldsymbol{\lambda}}^{i+3}$

(b) Scale-adaptive feature sampling

Figure 2.8: **Comparison between the standard sampling and our scale-adaptive one.** We show here an example of five consecutive samples obtained with both techniques in the case of the simpler parametrization used on our medical datasets (Eq. 2.26) with only one option for ω. In the uniform case (a), every new sample is obtained by resampling the whole set of parameters. The allowed value for the offsets and box sizes is constrained by a parameter δ. In our scale-adaptive method (b), a current sample is slightly deformed to create a candidate for the next sample, which is only accepted if it does not decrease the information gain on the set of training samples sent to the current node. Instead of δ, a parameter σstates how large the deformation between two consecutive samples is allowed to be.

candidate feature depend on the previous one (Fig. 2.8). At each node, given an arriving set of training samples $\mathcal{S}$, the feature sampling is conducted as follows:

- Sample a first feature $\boldsymbol{\lambda}^1$ by setting the scale-related parameters to values corresponding to the finest scale, i.e. 0 for offset coordinates and 1 for box dimensions. The categorical parameters are set randomly.

- At each iteration (for $1 \leq i \leq n_{\text{tries}} - 1$) :

- Given the current feature $\boldsymbol{\lambda}^i$, suggest a slight modification $\tilde{\boldsymbol{\lambda}}^i$ of $\boldsymbol{\lambda}^i$ by picking at random one of the dimensions λ_d^i (with $d \in \{1, \dots, D\}$ uniformly drawn) and redraw it as follows:
    * If d is a scale-related dimension, we redraw λ_d^i uniformly within a neighborhood $[\lambda_d^i - \sigma, \lambda_d^i + \sigma] \cap \Lambda_d$.
    * If d is a categorical dimension, we redraw λ_d^i uniformly over its set of possible values Λ_d.

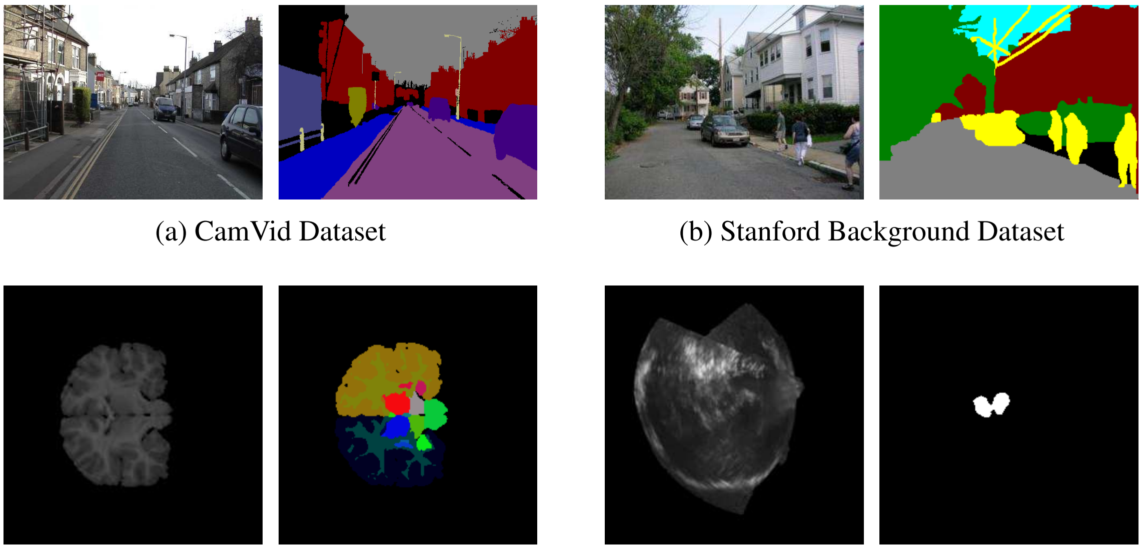
  The other components of $\boldsymbol{\lambda}^i$ are left unchanged.
- Accept this modification if it does not decrease the purity gain. Formally, define $\boldsymbol{\lambda}^{i+1} = \tilde{\boldsymbol{\lambda}}^i$ if Gain$(\tilde{\boldsymbol{\lambda}}^i, \mathcal{S}) \geq$ Gain$(\boldsymbol{\lambda}^i, \mathcal{S})$, else $\boldsymbol{\lambda}^{i+1} = \boldsymbol{\lambda}^i$. Gain$(\boldsymbol{\lambda}, \mathcal{S})$ denotes the maximum purity gain observed when trying several axis-aligned splits based on the feature $\boldsymbol{\lambda}$, i.e. Gain$(\boldsymbol{\lambda}, \mathcal{S}) = \max_{\theta \in \Theta_{\boldsymbol{\lambda}}}$ Gain$(\boldsymbol{\lambda}, \theta, \mathcal{S})$ with the notations of Eq. 2.15.

The procedure above generates n_{tries} features $\boldsymbol{\lambda}^1, \dots, \boldsymbol{\lambda}^{n_{\text{tries}}}$, exactly like the standard uniform sampling technique, and requires the same amount of purity gain evaluations so that the practical computational time is identical by construction. Intuitively, the chosen initialization of the scale-related parameters corresponds to the finest possible scale which only provides information contained at the voxel of interest. Through the creation of candidate moves $\tilde{\boldsymbol{\lambda}}^i$, changes towards larger scales are then progressively suggested, but only accepted if they convey more information than the current one, in a hill climbing fashion. Hence, the maximum scale δ can be set as high as necessary in practice, i.e. as large as the image dimensions. The issue of finding a good value for δ has now been replaced by finding a good size of neighborhood σ. As we will see in the experimental evaluation, tuning σ is much less critical and virtually any value of σ outperforms the best choice of δ (Fig. 2.10).

Our approach can also be seen as the design of a Markov chain at each node, where each feature corresponds to a state of the Markov chain and where moves are sequentially suggested from a proposal distribution and accepted if they do not decrease the purity gain. This shares some similarities with the Metropolis-Hastings algorithm [Metropolis et al., 1953] which we will encounter in Chapter 5. The difference lies in the fact that the acceptance criterion is here deterministic, and that the application of the Metropolis-Hastings algorithm usually requires more iterations than the desired number of samples n_{tries} to reduce the correlation between consecutive samples.

2.4.3 Experiments

We evaluate our approach on four datasets. We first consider two 2D datasets in computer vision: the Camvid dataset [Brostow et al., 2008a], which consists of 711 road scenes with 32 possible pixel labels, and the Stanford Background Dataset [Gould et al., 2009] which is a collection of 715 natural images parsed into 8 semantic classes. As in previous works [Brostow et al., 2008b, Kontschieder et al., 2013, 2014], we simplify the CamVid dataset by only using 600 frames (367 for training and 233 for testing) and 11 classes and



(a) CamVid Dataset (b) Stanford Background Dataset

(c) IBSR2-18 Dataset (d) Midbrain Dataset

Figure 2.9: **Overview of the datasets.** We show here one example image from each of the 4 datasets used for our experimental evaluation, together with their respective ground truth.

we downsample each image by a factor 2. The two other datasets used for our evaluation are made of 3D medical volumes: the IBSR2-18 Brain Dataset, which is a publicly available[3] set of 18 brain MR scans with up to 32 labeled regions, and a dataset which aims at segmenting the midbrain in 3D transcranial ultrasound [Ahmadi et al., 2011]. The volumes are rescaled to obtain a constant isotropic spacing over each dataset, of $1\ \mathrm{mm}$ and $0.9\ \mathrm{mm}$ respectively.

Evaluation of the Scale-Adaptive Training

To evaluate our proposed method for a scale-adaptive training, we used an identical forest setting for all 4 datasets. Each forest is composed of $n_{\text{trees}} = 50$ trees of maximal depth 20 and with at least 10 samples per leaf. At each node, $n_{\text{tries}} = 500$ features were sampled during training, and $n_{\text{thresholds}} = 10$ thresholds tried on a regular grid for each feature (Sec. 2.1.4). Each tree received 10^6 training samples uniformly drawn at random over all training pixels in a bagging fashion, and we applied for each tree the class balancing correction exposed in Sec. 2.1.7. For the CamVid dataset, we used the training and testing sets usually chosen in the literature. For the three other datasets, a 2-fold cross-validation was performed. On the two 2D datasets (CamVid and Stanford Background Dataset), the features were computed in the CIELab color space, and the histograms of gradients were created using $B = 9$ orientation bins. These settings were applied for both the standard and the scale-adaptive forest training and for values of δ and σ in $\{10, 25, 50, 100, 200\}$. We report the quality of the predictions in Fig. 2.10, using for

[3]http://www.nitrc.org/projects/ibsr

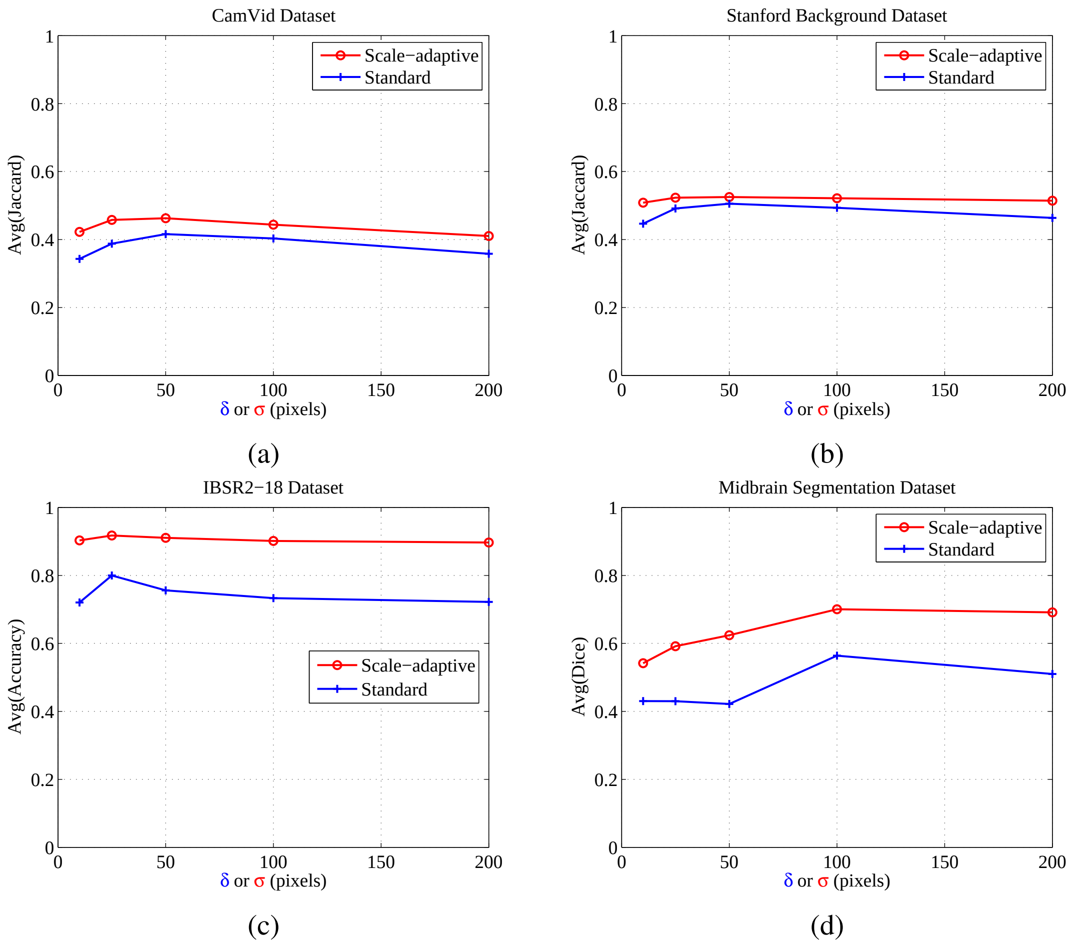


Figure 2.10: **Evaluation of our scale-adaptive training.** We compare the standard forest training based on uniform sampling with the scale-adaptive training method presented in Sec. 2.4.2. For each of these two methods, we let vary the parameter they depend on, i.e. respectively δ (size of the patch) and σ (size of the neighborhood during local search). The results demonstrate that the scale-adaptive method outperforms the standard training on a variety of datasets.

each dataset the most commonly used metric: the average Jaccard index over classes for the computer vision datasets, the average accuracy over volumes for the IBSR2-18 Brain Dataset and the average Dice score over volumes for the Midbrain Dataset. The results demonstrate a consistent improvement of the scale-adaptive approach in comparison to the standard one, almost regardless of the chosen values for σ and δ. Qualitative results on each dataset can be found in Fig. 2.11.

We also investigated on the CamVid dataset the randomization abilities of the scale-adaptive training. We performed this evaluation by removing the bagging aspect of the training and sending instead the same set of samples to each tree, more precisely by collecting samples on a regular grid of step size 8. Thereby, the variability between trees is only introduced via the feature sampling. Considering this setting, the evolution of the average classwise Jaccard index with the number of trees is shown in Fig. 2.12a. The

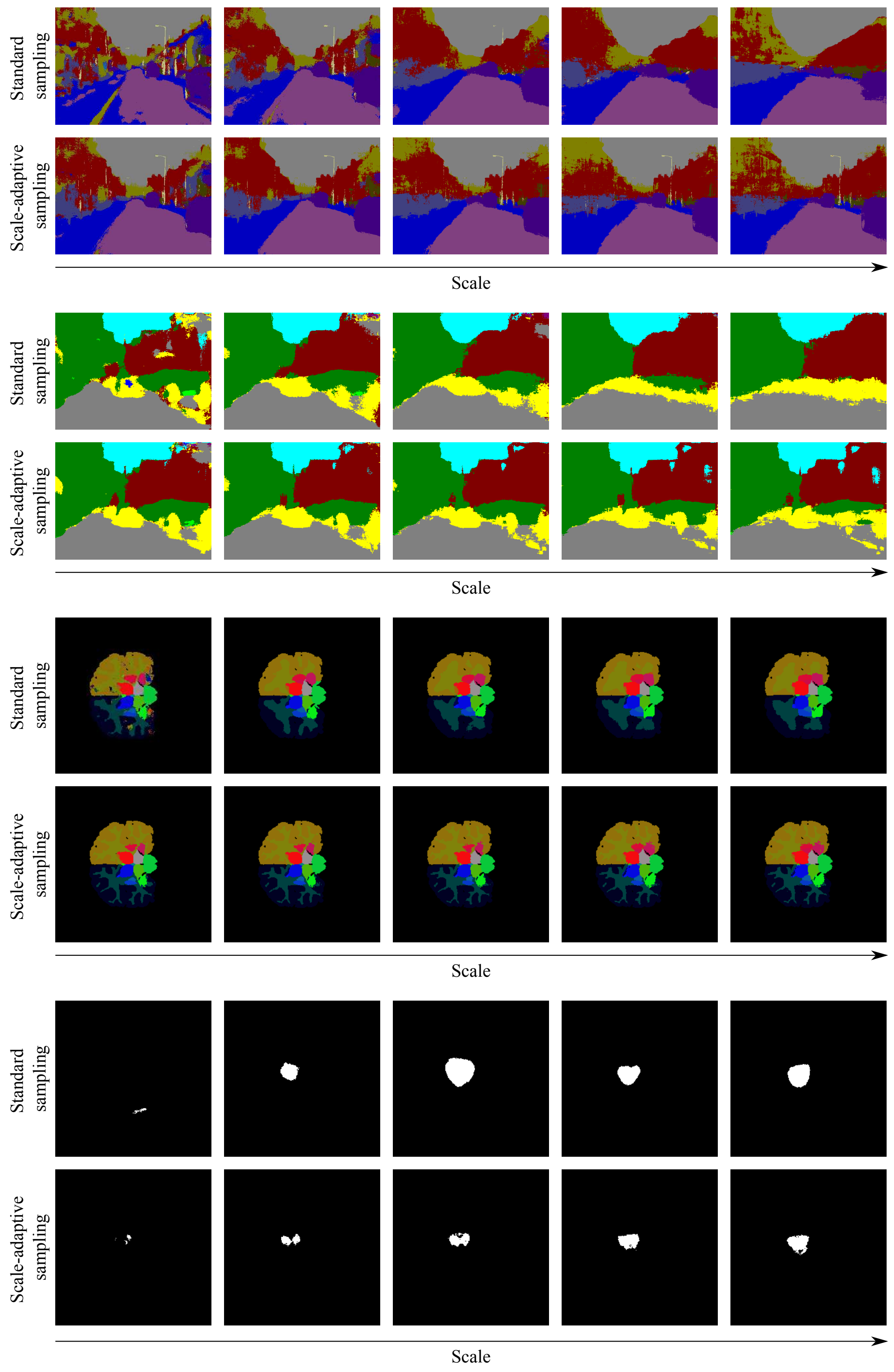
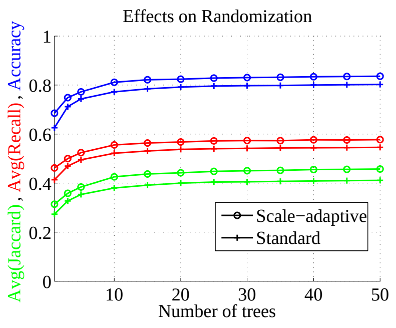


Figure 2.11: **Qualitative results.** Evolution of the predictions for standard and scale-adaptive samplings when the scale parameter increases (respectively δ and σ). The images used here are the example images shown in Fig. 2.9.



(a)

Method	Accuracy	Avg(Recall)	Avg(Jaccard)
Standard Forests			
Brostow et al. [2008b]	69.1	53.0	–
Kontschieder et al. [2013]	–	–	33.3
Kontschieder et al. [2014]	69.9	42.2	30.6
Rota Bulò and Kontschieder [2014]	68.5	50.3	32.4
Best Results (including structured approaches)			
Yang et al. [2012]	73.7	36.3	29.6
Kontschieder et al. [2013]	–	–	38.3
Kontschieder et al. [2014]	83.8	53.2	43.5
Rota Bulò and Kontschieder [2014]	82.1	56.1	43.3
Ravì et al. [2016]	76.4	72.5	–
Ours			
KS, Color only	70.4	42.6	31.7
KS, Color + HoG	79.7	47.8	38.6
Standard	80.4	54.5	41.6
Scale-adaptive	83.8	58.6	46.2

(b)

Figure 2.12: **Complementary results on the CamVid dataset.** (a) We study whether the randomization abilities are conserved with the scale-adaptive sampling. To do so, we no longer perform bagging and we leave the tree randomization mechanism to the feature sampling only. We can observe that the evolution of the evaluation measures with the number of trees is similar for standard and scale-adaptive sampling, which tends to demonstrate similar randomization abilities. (b) We position our work with respect to other existing forest-based approaches. Although our final classifier remains an axis-aligned forest without regularization, it outperforms most of the structured methods.

benefit of adding trees follows a similar pattern for both the standard and scale-adaptive trainings, so that no clear loss of randomization abilities can be observed with the latter.

Finally, we also mention that following the publication of this approach [Peter et al., 2015], an external research group reimplemented independently the scale-adaptive training as a baseline and was able to confirm on another medical dataset the observed improvement over the standard uniform training scheme [Štern et al., 2016].

Comparison with Existing Methods

The CamVid dataset has been used in computer vision to evaluate several other contributions on random forests, which gives us the opportunity to position our approach with respect to these other models. A summary of the results of forest-based methods reported in the literature is shown in Table 2.12b. We distinguished the results using a standard forest setting (often used as baseline in the corresponding papers), and the more sophisticated contributions involving either structured output [Yang et al., 2012, Kontschieder et al., 2013, 2014], stronger splitting functions in nodes [Rota Bulò and Kontschieder, 2014], or considering random forests as a part of a more complex pipeline [Ravì et al., 2016]. We can see that, in spite of being by nature a classical random forest with axis-aligned splits and without regularization of the output, our scale-adaptive forest outperforms almost all other approaches, including the structured ones. The method of Ravì et al. [2016] demonstrates a better average classwise Recall but uses forests within a larger pipeline, which makes the contribution of the forest itself difficult to assess. To estimate the impact of the feature representation on our results, we considered an additional forest with similar settings to the ones used by the best-performing baseline [Kontschieder et al., 2013], i.e. $n_{\text{trees}} = 10$, $n_{\text{samples/leaf}} = 5$ and $n_{\text{tries}} = 500$. We trained a forest in these settings but with our features, both using only color-based Haar features (`KS, Color only`) and with both color and histogram of gradients (`KS, Color + HoG`). The results demonstrate the clear benefit of introducing histograms of gradients within Haar-like features and that our feature representation provides a benefit of around 5 points in the Jaccard index. This should be thus considered when performing comparisons. However, we also mention that our scale-adaptive scheme only impacts the node optimization stage during training and is therefore compatible with structured models [Yang et al., 2012, Kontschieder et al., 2013, 2014].

Finally, taking for example $\sigma = 50$, we also obtain a higher mean Dice score (62.4 vs 33.0) than the state-of-the-art forest-based method on the midbrain dataset [Chatelain et al., 2013]. Our accuracy on the IBSR2-18 Brain Dataset (91.0) also compares favorably with the state-of-the-art [Zikic et al., 2014] who reported a mean score of 83.5. As can be seen in Fig. 2.10c, our standard forest is however slightly below the result reported by Zikic et al. [2014], which demonstrates that the scale-adaptive training scheme is directly responsible for the observed improvement. Interestingly, when training a standard forest at the largest scale $\delta = 200$, we obtain an accuracy of 72.2 which approximately corresponds to the performance of a registered label prior as reported by Zikic et al. [2014], respectively 65.8 with an affine registration and 76.8 for a non-linear registration. This confirms the idea that forests trained at large scales capture the general organization of labels. Finally, we can note that the two aforementioned approaches used a leave-one-out cross-validation instead of our 2-fold setting, which presumably gives them an advantage in terms of amounts of available training data.

2.5 Conclusion

In this chapter, we have exposed in details the learning framework of decision forests and demonstrated its applicability to semantic image segmentation problems. We especially introduced an alternative to the standard training which, when used in combination with Haar features, improves the quality of the predictions without difference in terms of computational cost. This concludes the first part of this thesis which concerns fully automatic techniques for segmentation. In the next chapters, we demonstrate how learning techniques can assist the user for the manual segmentation of images in an interactive fashion.

Chapter 3

Bringing Humans In the Loop

So far, this thesis has focused on the problem of automatic semantic segmentation of images where, at prediction time, the image must be accurately parsed into regions whose semantic content is simultaneously identified without a single human intervention. We now discuss the limitations of defining an image segmentation problem as a learning-based semantic segmentation task. We will show that fully automatic segmentation is not always necessary or even desirable in practice. Instead, algorithms can be designed in an interactive fashion to *assist* a human user in their manual analysis of images. After reviewing briefly the standard setting of interactive segmentation, we discuss two situations where its applicability is not straightforward which motivates the last two contributions of this thesis, respectively exposed in Chapter 4 and Chapter 5.

3.1 Practical Limitations of Supervised Learning

In practice, several difficulties can be encountered when a problem is modeled as a supervised learning task. In this section, we discuss some of these challenges and review methods which have been proposed to address them, with a particular focus on forest-based techniques. We group these limitations into three main thematics, respectively discussing the time-consuming aspect of annotating data, the possibility to store an entire labeled dataset at training time and the reliability of the labeled training data.

3.1.1 Time-Consuming Construction of a Training Set

A fundamental assumption of the supervised learning formulation is the availability of labeled training data corresponding to the task at hand. In the case of semantic segmentation, labelings must be provided by a human user on some (usually numerous) example images. However, the preliminary acquisition and manual delineation of training data is a time-consuming step, so that building an entire representative training set is not always feasible. The following strategies may be considered to reduce the annotation time.

Facilitating the labeling step itself Instead of requesting explicitly contours for the annotations, an interactive segmentation tool can be used to automatically generate a labeling based on weaker kinds of user inputs such as scribbles or bounding-boxes, together with the possibility of refining the output until the human user is satisfied by the segmentation. In Sec. 3.2, we will mention examples of such algorithms for interactive segmentation. Note that, if the context allows it, these algorithms may as well be directly applied on the test image removing the need for a prediction method. Another alternative is to design algorithms which directly learn from weak annotations, for instance by using a multiple instance learning formulation for which forest-based methods have been introduced [Leistner et al., 2010, Vezhnevets and Buhmann, 2010].

Labeling a few well-chosen images Within an available set of images which is to be labeled to serve as a training set, it is reasonable to assume that some visual redundancies can be found. Given a fixed time constraint on the number of images that we can afford to label, it is intuitively more informative to label cases which are complementary instead of similar ones. Active learning methods aim at reducing the annotation time by iteratively querying the label of the examples that are considered to be the most useful for a classifier trained on the already available training data [Settles, 2010]. Some active learning techniques have been specifically introduced to tackle segmentation tasks. For the segmentation of organs within CT volumes, Iglesias et al. [2011] suggest the next volume to label as the one that maximizes the disagreement between a small-range voxel-wise forest and a generative model encoding the long-range arrangement between organs. To annotate a 3D volume, another active learning method selects planes with the highest aggregated uncertainty, reducing the 3D segmentation task to a series of well-chosen 2D segmentations [Top et al., 2011]. In computer vision, the compromise between effort and informativeness can be predicted for several types of labelings (i.e. weak or strong annotations), so that both the next image to label and the type of annotations are suggested to the human user [Vijayanarasimhan and Grauman, 2011]. Finally, another method predicts whether a human should segment an image or if an automatically generated one would suffice, so that efforts are spent on worthwile examples only [Gurari et al., 2015] .

Adding unlabeled data Semi-supervised learning algorithms are trained on a mixture of labeled and unlabeled data, where the unlabeled data are usually present in greater amounts. Although they do not carry label information, unlabeled data can still be useful, for instance to identify clusters of samples to let the decision boundary of a classifier split the training data through areas of low sample density. Semi-supervised versions of decision forests have been proposed which either iteratively train a series of forests on labeled versions of the dataset while monitoring the generalization error [Leistner et al., 2009] or which incorporate unlabeled data in the node split optimization during training [Criminisi and Shotton, 2013, Liu et al., 2015]. Although the objective is no longer the exploitation of unlabeled data to reduce the labeling effort, we also mention that semi-supervised learning methods can also be used in a transductive scenario, i.e. when the test data are already known at the training stage and incorporated as unlabeled data. A recent example of the transductive setting with semi-supervised forests segments brain tumors within post-operative scans by using the pre-operative scan as training set [Meier et al., 2014].

Adding data previously labeled for a different but related task In its strict application, the supervised learning formalism imposes to create a new training set as soon as we plan to address a new application. However, some applications are related to one another, so that treating them completely independently may seem to be a loss of information. For instance, if a labeled dataset for brain tumor segmentation was built, these data should be at least partially exploitable to tackle the task of liver tumor segmentation, although the type of images will be different. Transfer learning techniques have been investigated to propagate information from a source task to a new target task, including a modification of decision forests which performs an internal node optimization achieving a compromise between the two tasks [Goussies et al., 2014]. An important special case of transfer learning is the domain adaptation scenario, where the objective remains the same but the distribution of data has changed. For instance, the acquisition setup of a machine has been modified, or the data comes from another hospital. With supervised domain adaptation algorithms, the labeled data from a new domain can be effectively enriched with the previous data from an old domain. In this direction, we proposed a supervised domain adaptation method for decision forests [Conjeti et al., 2016] which performs tissue characterization on *in vivo* intravascular ultrasound images, mainly built on *in vitro* training examples which are easier to collect. In this approach, a forest preliminary trained on the *in vitro* images is adjusted at both internal node and leaf levels based on the few *in vivo* samples available . In a similar direction, scandent trees were introduced to enrich a few multimodal examples with a large amount of data with missing modalities [Hor and Moradi, 2016]. To save labeling time, additional source examples can also be generated in a synthetic way as was done for the task of human pose estimation [Shotton et al., 2011].

3.1.2 Offline Availability of a Training Set

In their standard formulation, supervised learning algorithms assume an access to a set of training examples on which the decision rule is learned. However, this access to the entire set of available annotations is not always guaranteed. In the case of very large data or in the medical domain where the protection of patient data is strongly regulated, data may be imposed to be stored at several different physical locations. In some situations, one might also have already trained a model before collecting data from a new source and wish to update the model without access to the original data. As a possible answer to these problems, online learning solutions have been explored to learn supervised models from a stream of incoming labeled data instead of a fixed training set. In the case of decision forests, one of the most popular strategies for online training is to grow trees progressively, starting from a root node, by turning a leaf node into an internal node as soon as a split of sufficient quality can be found, both in terms of information gain and statistical representativity [Saffari et al., 2009]. Another recent work models trees as samples from Mondrian distributions [Lakshminarayanan et al., 2014] and improves over existing online approaches at the cost of losing the compatibility of forests with high-dimensional feature spaces. Another method based on Bayesian online learning was proposed to learn online a forest of shallow trees of fixed structure [Rota Bulò and Kontschieder, 2016].

3.1.3 Reliability of the Labeled Data

Another fundamental assumption of supervised learning is the fact that the labeled dataset used for training accurately describes the prediction objective. However, for some applications, the reliability of the training data itself may be questioned. The two following situations can for instance be considered.

Inaccurate Labels To perform a manual semantic annotation of a training image, the person in charge of the labeling must be able to interpret the content of the image to provide the correct answer. However, a lot of subjectivity can be introduced in these ground truth annotations, for instance due to ambiguous labels, difficult imaging conditions or the necessity of a high expertise in the field to perform correct decisions. The latter situation is for example common in histopathology, where the accurate identification of diseased areas based on their visual appearance can be a very difficult task (see Sec. 4.1). The lack of reliable annotations complicates both the training and evaluation of learning algorithms. To mitigate this effect, several experts can be asked to annotate the same data and their labels could be merged, modeling the quality of each expert as a latent variable [Warfield et al., 2004]. To take into account inter-experts disagreements directly at the learning stage during the training of a random forest, we proposed an approach which assigns a quality to each expert at each node and performs the node optimization according to the best performing expert [Chatelain et al., 2013]. Thereby, the expertise of each annotator is made dependent on the considered area of the feature space, encoding the fact that experts can have complementary skills regarding the annotation of the objects of interest. The problem of merging several labels can also be encountered when labels are provided by a large crowd of non-experts [Maier-Hein et al., 2014].

Unexpected Change of Target Distribution In Sec. 3.1.1, we mentioned domain adaptation methods which leverage labeled data from a different but related task to enrich an existing training set. Similarly, an unexpected change of domain can be a source of mistakes during the application of a supervised learning method, as testing instances no longer match the learned distribution. We will see an example in histopathology (Chapter 4) where the experimental preparation of the tissue results in changes of dye concentration which are difficult to control. If the new target domain remains close enough from the original training data, online domain adaptation techniques can be used to update the learned model directly at prediction time based on the newly observed test data. For example, a generic unsupervised method based on Gaussian process regression was introduced to adapt the decision boundary of any black-box classifier to a target image [Jain and Learned-Miller, 2011]. A more recent approach automatically adapts a trained classifier by focusing on learning new class proportions based on the newly collected labels [Royer and Lampert, 2015]. Other approaches combine a classifier trained on the source data with an online classifier continuously updated from the target data [Zhao and Hoi, 2010, Tommasi et al., 2012]. Originally designed with kernel-based classifiers, transferring this technique to forest models would be challenging as an accurate online learning technique for random forests is then required (see Sec. 3.1.2). In fact, the online forest approach designed by Saffari et al. [2009] already includes, in the context of object tracking, the idea of online domain adaptation by discarding trees when they no longer match the distribution of the arriving samples.

3.2 Interactive Segmentation

The full automation of semantic image segmentation is appealing as it provides, for some applications such as robotics or self-driving vehicles, a building block encoding the understanding of the scene on which the automation of other processes can build. In other situations without real-time constraints (such as the segmentation within medical scans to quantify anatomical volumes), an automated segmentation can also allow to save the time that would be required for a human to delineate manually the images. However, the design of learning-based segmentation solutions reaching human-level performance is far from being an easy task: the possible limitations described in the previous section must be anticipated and taken into account, including the availability of large amounts of data. For applications where these constraints cannot be easily satisfied, e.g the case often encountered in medical applications where annotated data are difficult to obtain, fully-automated segmentation algorithms may not be the most practical option. In fact, clinicians might rather be inclined to use a segmentation algorithm to gain time while keeping the possibility to check its output visually and to correct it if necessary. If the clinician is kept in the loop, possibilities of interacting with the user can be considered.

In their most basic form, interactive segmentation methods are flexible techniques which do not rely on any learned model and only request from the human user a few lightweight inputs directly on the image to segment. These inputs can take the form of a bounding box around an object of interest [Lempitsky et al., 2009a, Grady et al., 2011] or of some seeds providing the true labels of pixels within the background and objects to segment [Boykov and Jolly, 2001, Grady, 2006, Price et al., 2010]. The provided information is then propagated to the rest of the image to generate the final segmentation output. The propagation scheme is algorithm-specific, and usually involves a graph representation of the image where neighboring pixels are connected by an edge whose weight is proportional to the difference of intensity between the two pixels. As an example of interactive segmentation algorithm, a method based on random walks [Grady, 2006] assigns to each non-seed pixel the label of the seed which is reached first by a random walk starting from this pixel, where the random walk transition probabilities are proportional to the weights of the image graph. Other algorithms are based on the geodesic distance transform which considers the shortest (weighted) path to each seed [Criminisi et al., 2008, Price et al., 2010]. Importantly, an approximation of the geodesic distance transform is available, with a computational time which is linear in the number of pixels [Toivanen, 1996]. Geodesic distance transforms will be used in Chapter 5 and are illustrated in Fig. 3.1.

The advantages of interactive segmentation algorithms are their flexibility and the fact that they allow a visual check of the segmentation output and possibilities of adjusting it via the addition of user inputs in wrongly segmented areas. However, the definition of the weights between pixels plays a critical role and must be manually defined. Usually,

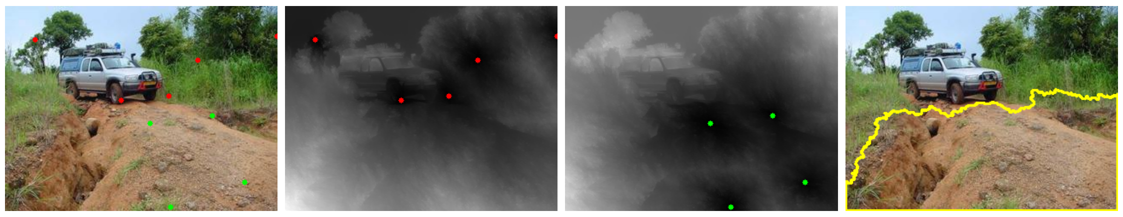


Figure 3.1: **Example of interactive segmentation using geodesic distance transforms.** (a) Input image with negative (red) and positive (green) seeds placed by the user respectively outside and inside the region of interest. (b) Geodesic distance to negative seeds. (c) Geodesic distance to positive seeds. (d) Resulting segmentation assigning to each pixel the label of the closest seed in the sense of the distances shown in (b) and (c).

weights are related to the color difference so that segmented areas are homogeneous, but this may not always be the most relevant choice (for instance in case of textured objects). Basing the segmentation on more complex features can thus be of interest, e.g from the output of a learning model such as random forests [Santner et al., 2009, 2010].

3.3 Our Contributions

As briefly reviewed in the previous section, interactive segmentation can be a relevant alternative to fully automatic segmentation, especially when the design of the latter faces practical constraints encountered in many clinical settings (Sec. 3.1). In the next chapters, we introduce two contributions building on the random forest framework and corresponding to two novel scenarios where standard interactive segmentation as described in Sec. 3.2 does not apply. More precisely:

- In Chapter 4, we consider the objective of quantification within large slides in digital pathology. In this context, a clinician is interested in finding and segmenting a small number of objects in a large image. The application of an interactive segmentation method thus faces the difficulty of *finding* the objects of interest within the large data so that foreground seeds could be placed. We propose to leverage a classification forest used for semantic image segmentation to retrieve regions of the slide likely to contain the objects of interest. The user is then allowed to report feedback on the relevance of the suggested regions, either via a direct segmentation or via lighter forms of inputs. Thereby, the forest model can be adjusted in real time to incorporate the specific properties of the current image, which may differ from the original training set due to the instability of tissue staining procedures in this particular application.

- In Chapter 5, we consider a scenario where a human user is only able to provide binary answers in the form of *Yes* or *No* and is thus no longer able to place seeds within an image. This theoretical framework opens alternative ways of interacting with the computer such as through a voice recognition system or a pedal. In this context, the interactive segmentation of an object in an image can be formalized as a Twenty Questions game between the user and the machine, where the latter must find the most appropriate questions to ask so that the object of interest can be guessed as quickly as possible. We distinguish the case where no prior knowledge about the object is available, where we then rely on traditional homogeneity assumptions, and the case where a random forest classifier has been previously trained to segment the object of interest so that the choice of questions can be guided more effectively.

Chapter 4

Merging Segmentation and Exploration: Application to Digital Pathology

In this chapter, we consider the case of very large images where the applicability of interactive segmentation would be preceded by a tedious search for the objects of interest. We introduce an interactive framework using a forest-based semantic segmentation approach to explore large digital slides according to the learned segmentation objective. In our application case, the clinical objective is the assessment of the surface covered by hematopoietic cells within mouse liver slides. Our main contributions are:

- The design of a region scoring function to convert pixelwise predictions into a score for each region of an image. Thereby, regions that are the most likely to contain objects of interest to the user can be retrieved to facilitate the exploration of the large data.

- An online domain adaptation scheme based on interactions with the user which adapts on-the-fly the classifier to the visual characteristics of the test data. Three real-time strategies are compared, including a novel approach based on online gradient descent which is compatible with lighter kinds of annotations.

- A regression-based strategy so that a whole-slide estimate of the surface covered by hematopietic cells can be predicted as a by-product of an only partial exploration of the slide.

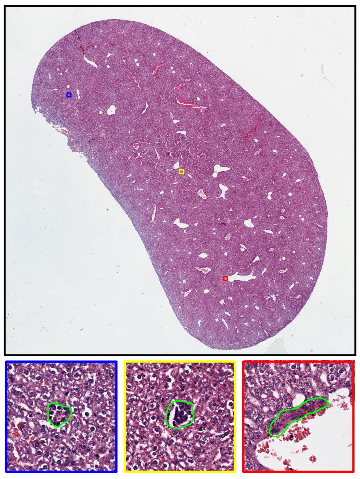
- An experimental study regarding the use of discretized user inputs for online adaptation, where we demonstrate how one-click inputs can be effectively used instead of accurate annotations.

The content of this chapter has been published as a journal article [Peter et al., 2016] and was also presented in part at a conference [Peter et al., 2014].

4.1 Clinical Motivation

Histopathology is a crucial tool in modern clinical practice. It consists in the microscopic observation of biological tissues surgically extracted from a patient, in order to collect information regarding the presence or extent of a particular disease in the sample. In particular, it is part of the standard experimental protocol for the definitive diagnosis, grading and staging of most cancer types and plays an essential role in the design of appropriate patient-specific treatments. Histopathological examinations usually aim at searching for a certain kind of anatomical structure, like biomarkers, cancer cells or necrotic areas, whose presence or proportion within the tissue has to be quantitatively estimated. Although this procedure is traditionally conducted under a standard optical microscope, digital acquisitions of entire slices can be performed at comparable resolutions and are increasingly used by pathologists in their clinical workflow as well as for educational and research purposes [Farahani et al., 2015]. Moving from optical to digital examinations has been shown to maintain similar diagnosis performances [Jukić et al., 2011, Bauer et al., 2013] and offers numerous additional advantages such as the applicability of image analysis algorithms, easier recordings and safer storage of patient data, and the possibility of displaying the scanned tissue to several examiners simultaneously [Al-Janabi et al., 2012]. However, because of their high resolution, the size of digitally acquired images is very large and commonly reaches the order of a billion of pixels [Cooper et al., 2012]. This increases the time required for manual quantification procedures: beyond the tediousness of annotating objects in images, a pathologist also spends a lot of time navigating through the large slide looking for evidence of the disease of interest. Moreover, the objects to localize may only be scarcely distributed, for instance at early stages of diseases or after a treatment has been applied. In such situations, the exploration phase even becomes the bottleneck of the process, since most of the time of the pathologist is spent scrolling through uninformative areas (Fig. 4.1a).

Some characteristics of the field of histopathology bring specific challenges for an automated analysis of the acquired images. Pathologists are typically trained several years before reaching satisfactory diagnosis abilities [Jaarsma et al., 2014], and the variability between experts remains nevertheless significant for several applications [Meyer et al., 2005, Gonul et al., 2006, Gilles et al., 2007, Eefting et al., 2009]. During a challenge on mitosis detection for breast cancer grading [Veta et al., 2015], the reported Dice overlap between two independent pathologists was reported to be 56.6, and would be presumably even less for untrained humans. Another common challenge in histopathological image analysis is the visual variability between two acquisitions. In particular, the consistency of the staining procedure is difficult to control experimentally leading to variations in terms of dye concentration (Fig. 4.1b). Therefore, an algorithm that has been trained or designed on labeled data may not generalize well to newly acquired samples. To mitigate this source of inaccuracies, color normalization can be performed as a preprocessing step and remains an active field of research [Rabinovich et al., 2003, Macenko et al., 2009, Khan et al., 2014, Onder et al., 2014, Bautista and Yagi, 2015, Vahadane et al., 2016], together with generic techniques for online domain adaptation (Sec. 3.1.3). Finally, a tissue extracted surgically and observed under a microscope is less structured than other


(a)

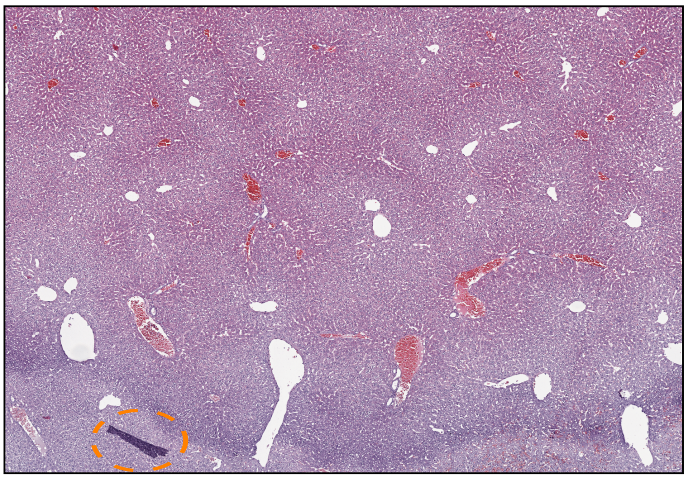

(b)

Figure 4.1: (a) **Typical histopathological slide.** Three regions (blue, yellow and red squares) containing a cluster of hematopoietic cells are highlighted. Our method aims at retrieving such regions of interest within a large slide which mostly contains irrelevant background areas. Note the absence of large-scale context to guide the visual search, which would require an exhaustive screening of the slide in the case of a manual examination. (b) **Examples of visual artifacts.** A portion of a slide is displayed here. The staining itself is inhomogeneous and presents a vertical shading. Moreover, a dark artifact is present (circled in orange). Such a visual variability between and within slides complicates the application of supervised learning techniques and prompts an adaptation at prediction time.

kinds of medical data such as body scans, while being of a much larger size. Objects of interest are expected to appear anywhere within the tissue, so that location or connectivity priors are rarely available.

In this chapter, we introduce an interactive method to assist a pathologist in exploring and quantifying large histological slides (Fig. 4.2) which we evaluate in the context of extramedullary hematopoiesis quantification within mouse liver slides. Our approach builds on a *pixelwise segmentation model* provided by a pre-trained random forest classifier as introduced in Sec. 2.4. We use its probabilistic output to perform an *interactive slide exploration* by suggesting, in a sequential manner, a series of candidate regions of interest (Sec. 4.3). This interactive navigation framework includes a component which allows the pathologist to provide, after each suggestion, some feedback about the actual relevance of the proposed region. From these user inputs, the underlying forest-based model is modified on-the-fly via a real-time online adaptation framework. This enables a progressive adjustment to the characteristics of the data at hand and compensates for possible mismatches with the original training set without specific assumptions about their nature, in contrast to the aforementioned explicit stain normalizations. We also show how a *whole-slide quantification* can be inferred after a partial exploration of the slide (Sec 4.4) and how one-click inputs can be used for faster interaction (Sec. 4.5).

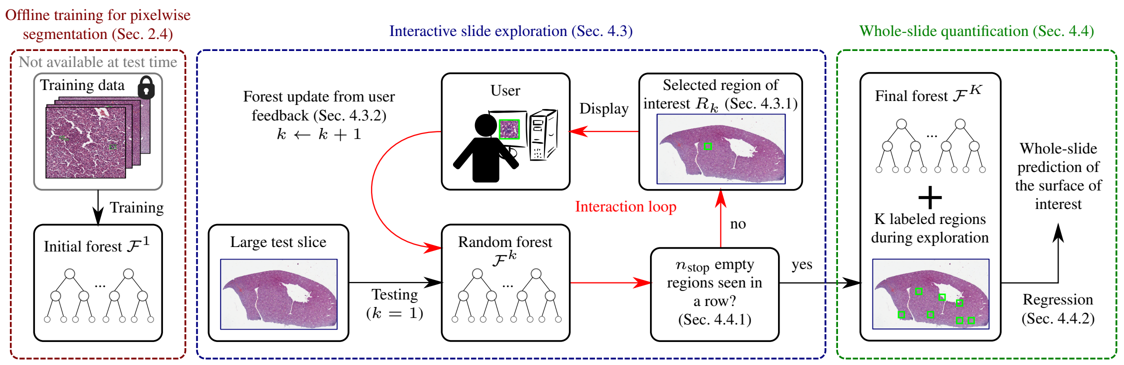

Offline training for pixelwise segmentation (Sec. 2.4)
Not available at test time
Training data
Training
Initial forest $\mathcal{F}^1$
Interactive slide exploration (Sec. 4.3)
Forest update from user feedback (Sec. 4.3.2)
$k \leftarrow k + 1$
User
Display
Selected region of interest R_k (Sec. 4.3.1)
Interaction loop
no
Large test slice
Testing ($k = 1$)
Random forest $\mathcal{F}^k$
n_{stop} empty regions seen in a row? (Sec. 4.4.1)
yes
Whole-slide quantification (Sec. 4.4)
Final forest $\mathcal{F}^K$
K labeled regions during exploration
Whole-slide prediction of the surface of interest
Regression (Sec. 4.4.2)

Figure 4.2: **Overview of our scenario.** Initially, a classification forest $\mathcal{F}^1$ is trained offline at pixel level to segment the structures of interest (in our application case, clusters of hematopietic cells). Given a large new slide to analyze, our method sequentially displays to the pathologist regions that are likely to contain these objects, thereby alleviating a tedious manual navigation through the slide. After the suggestion of a region R_k ($k \geq 1$), chosen as to maximize a forest-based scoring function $\phi(.|\mathcal{F}^k)$, the user provides a relevance feedback about its actual content. From this input, the current forest $\mathcal{F}^k$ is updated in real time leading to a new forest $\mathcal{F}^{k+1}$. By doing so, the visual characteristics of the test slide are progressively incorporated into the decision model so that upcoming region suggestions can be reconsidered. The exploration is stopped after seeing a certain number n_{stop} of negative regions suggestions in a row. The updated forest and the user labels collected during the exploration can then be combined in a regression framework to predict the total surface covered by hematopietic cells in the tissue, including the areas not observed by the pathologist.

4.2 Related Work

4.2.1 Learning-Based Image Segmentation in Histopathology

A large number of methods have been introduced for the automated analysis of histological slides and are progressively put into practice, as demonstrated by the development of general-purpose toolboxes such as Ilastik [Sommer et al., 2011], CellProfiler [Carpenter et al., 2006] and CellCognition [Held et al., 2010]. We refer to surveys [Gurcan et al., 2009, Veta et al., 2014] for a broader overview of the field and focus here more specifically on the case of segmentation of histological images. In this context, learning-based techniques have been successfully applied for different tasks, including cell segmentation within follicular lymphoma images [Kong et al., 2011] and within lung and brain tumor samples [Su et al., 2015], segmentation of cancer tissue within colon images [Xu et al., 2014], and whole-slide segmentation of necrotic areas [Homeyer et al., 2013]. The prediction of the Gleason grading, which is one of the most important quantitative measures for prostate cancer staging, has also gathered a particular interest in the field. Several statistical learning methods such as support vector machines [Nguyen et al., 2014], AdaBoost [Gorelick et al., 2013] and randomized forests [Khurd et al., 2010] were used towards an automatic prediction of this score. To overcome the difficulty of efficiently processing large whole-slide images, multi-resolution approaches were designed to find and segment regions of interest in a hierarchical manner [Sertel et al., 2009, Roullier et al., 2011, Huang et al., 2011, Doyle et al., 2012]. These approaches simulate the behavior of a pathologist, starting from the lowest resolution and progressively refining the analysis towards presumably interesting areas. In this work, our segmentation model is a pixelwise forest classifier trained with Haar-like features (Sec. 2.4), whose efficiency allows us to operate directly at the highest resolution instead.

4.2.2 Assisted Navigation within Large Digital Slides

While the aforementioned methods focus on the segmentation task, a few other approaches aim at identifying regions of interest within histological data. A method for classifying regions as relevant or irrelevant using support vector machines was introduced [Bahlmann et al., 2012], and extended to a scenario where the ground truth is generated by analyzing the actual behavior of a pathologist with viewport tracking data [Mercan et al., 2014]. These classification techniques are closer to our goal but differ methodologically in two aspects. First, the methods above model the region retrieval task as a classification problem, ignoring the differences between positive regions. In particular, one may desire to display in priority regions containing larger structures of interest. Building our region scoring scheme on an underlying segmentation model naturally provides such a ranking of regions and gives the opportunity to extrapolate the quantification estimate to unobserved areas. Secondly, our method is more flexible, as it includes the ability to update the region selection rule from user annotations collected after each suggestion, in an online domain adaptation fashion.

4.2.3 Online Domain Adaptation

Experimental constraints during the preparation of a tissue may induce inconsistencies in terms of visual aspect between acquisitions. In particular, a newly acquired sample may differ from the data used to train the initial classifier. The problem of domain adaptation consists in the correction of such a shift between the distributions of the training and testing data. Most domain adaptation strategies retrain a new classifier once samples from the target domain have been observed. In our case, new samples are collected every time a suggested region is labeled by the user, after which the current classifier is accordingly adapted. To keep this interaction loop tractable in practice, the updates must be performed in real time, which excludes a retraining of the classifier between two suggestions. Because of this constraint, our scenario is more precisely an *online domain adaptation* task (Sec. 3.1.3). In our case, the large size of histological data, and hence the large number of pixelwise predictions which must be updated at each iteration imposes strict computational constraints. In particular, modifications of the structure of the trees [Saffari et al., 2009] are compromised. By acting on the leaf probabilities only, we achieve real-time updates between two interactions (Sec. 4.3.2).

4.2.4 Active Learning

Finally, querying user labels in order to improve a classifier can be seen as a form of active learning (Sec. 3.1.1), for which a few approaches were introduced in the context of histopathology [Homeyer et al., 2011]. In general, active learning algorithms query the label of the most uncertain samples given the knowledge of the current classifier, in order to minimize the labeling effort from the user. The spirit of our approach is different: from a clinical perspective, a pathologist is only interested in seeing positive examples in a short amount of time. This asymmetry between positive and negative observations leads us to focus on finding and displaying positive regions as quickly as possible, so that they can be visually inspected and validated by the user. The annotations obtained during the process are used to assess the accuracy of the initial model and correct it if necessary. Moreover, by doing so, any erroneous region suggestion naturally provides a challenging negative example to include in the online adaptation process.

4.3 Interactive Slide Exploration

Our slide examination method is based on an initial forest-based model whose goal is to segment the objects of interest within the tissue. This initial forest, denoted $\mathcal{F}^1$, is trained offline on some labeled examples and encodes the available prior knowledge before observing the test data. After training, the original training data are no longer considered available. This assumption is driven by practical aspects: while sharing and transferring a classifier from a machine to another is straightforward, this is usually less feasible with patient data which are of larger size and subject to ethical considerations. The training mechanism generating $\mathcal{F}^1$ from labeled images is conducted with Haar features in the scale-adaptive way presented in Sec. 2.4.

Considering a new test slide, we partition it into a predefined set $\mathcal{R}$ of non-overlapping regions of fixed size $\Delta \times \Delta$. The first step of our algorithm consists in retrieving the region $R_1 \in \mathcal{R}$ of highest interest to the user. Since $\mathcal{F}^1$ provides a pixelwise estimate, this choice of region is made according to a region scoring function $\phi(R|\mathcal{F}^1)$, which predicts the expected interest of a region R given the knowledge carried by the forest model $\mathcal{F}^1$. The first region displayed to the pathologist is $R_1 = \mathrm{argmax}_{R\in\mathcal{R}}\, \phi(R|\mathcal{F}^1)$. Once R_1 has been shown, the pathologist reports the actual relevance of its content. To do so, two possibilities of user labelings are considered in this work: either a full delineation of the object of interest in R_1, which is accurate but time-consuming, or a one-click input obtained by discretization, which is faster to provide but more ambiguous (Sec. 4.5). Using the input of the pathologist on R_1, the forest $\mathcal{F}^1$ is accordingly modified, leading to a new forest $\mathcal{F}^2$. This procedure is repeated several times by showing, at each iteration, the region $R_k = \mathrm{argmax}_{R\in\mathcal{R}\setminus\{R_1,\ldots,R_{k-1}\}}\, \phi(R|\mathcal{F}^k)$. In Sec. 4.3.1, we describe in more details our choice of scoring function ϕ. Section 4.3.2 is dedicated to the techniques for online domain adaptation, where three real-time alternatives are described including a novel approach based on online gradient descent.

4.3.1 Region Scoring Function

After training on a set of labeled images, we obtain a random forest classifier $\mathcal{F}^1$ which outputs, for any observation $x = (\mathbf{p}, I)$ corresponding to a location $\mathbf{p} \in \Omega_I$ in an image I (following the notations introduced in Sec. 1.3), a probabilistic estimate $P(Y = 1|X = x, \mathcal{F}^1) \in [0, 1]$ that x is a positive instance, i.e. is part of one of the sought structures. Since the goal of our approach is to display regions of interest to a pathologist, we use these pixelwise forest predictions to build a region scoring function ϕ. We propose to define the score of a region R as

$$\phi(R|\mathcal{F}) = \sum_{x \in R} P(Y = 1|X = x, \mathcal{F}) \tag{4.1}$$

given a pixelwise classification forest $\mathcal{F}$. This scoring function can be interpreted as the mathematical expectation of a random variable counting the number of positive pixels in the region R . In particular, regions containing larger objects obtain a higher score. Since the quantification task consists in the estimation of the total surface covered by the structures of interest within the slide, this amounts to showing first regions which have a greater contribution to this quantity. Due to its simplicity, our scoring function possesses important properties in the context of forest updates (computational efficiency and convexity) which will be detailed in Sec. 4.3.2. These advantages result from the fact that $\phi(R|\mathcal{F})$ can be rewritten as a scalar product between the vector of leaf models of the forest $\mathcal{F}$ and a sparse vector characterizing the region R. The derivation of this equivalent formulation is exposed in the next paragraph.

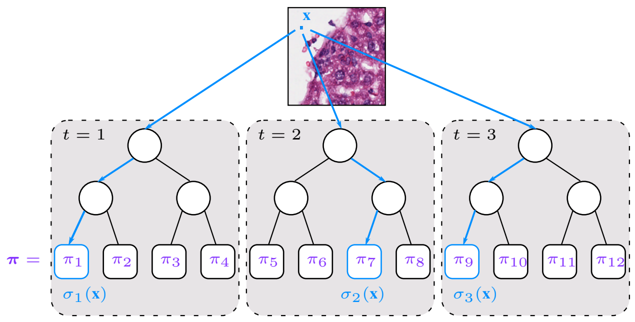
Expressing $\phi(R|\mathcal{F})$ as a scalar product Let us first introduce some notations. The set of leaf nodes belonging to the t^{th} tree is denoted $\mathcal{L}_t$. $\mathcal{L} = \cup_{1 \leq t \leq n_{\text{trees}}} \mathcal{L}_t$ is the set of all leaf nodes contained in the forest $\mathcal{F}$, and we denote tree$(L) \in \{1, \dots, n_{\text{trees}}\}$ the index of the tree to which a leaf $L \in \mathcal{L}$ belongs. We arbitrarily order the finite set $\mathcal{L}$ and consider the leaf probabilities jointly as a (finite-dimensional) vector $\boldsymbol{\pi} = (\pi_L)_{L \in \mathcal{L}}$. We denote $\Sigma = (\sigma_t)_{1 \leq t \leq n_{\text{trees}}}$ the list of routing functions $\sigma_t : \mathcal{X} \to \mathcal{L}_t$, which assign to each sample $x \in \mathcal{X}$ the leaf $\sigma_t(x)$ that it reaches when passed through the t^{th} tree (Fig. 4.3a). Intuitively, Σ encodes the structure of the forest determined by the arrangement of the nodes and the splitting functions, i.e. the way the forest partitions the space of observations $\mathcal{X}$, while the vector $\boldsymbol{\pi}$ defines the label predictions stored in the terminal nodes. Σ and $\boldsymbol{\pi}$ fully determine the forest decision rule, defined as

$$P(Y = 1|X = x, \mathcal{F}) = \frac{1}{n_{\text{trees}}} \sum_{t=1}^{n_{\text{trees}}} \pi_{\sigma_t(x)}. \tag{4.2}$$

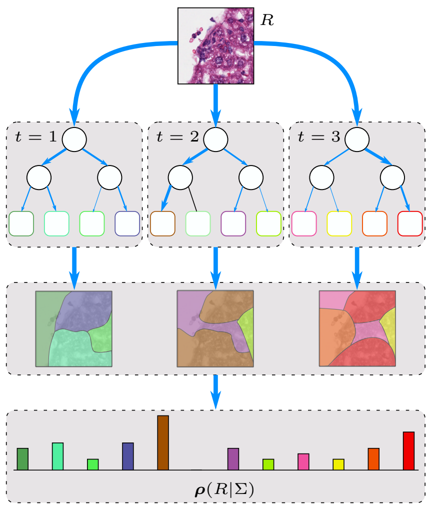
Combining Eq. 4.2 and Eq. 4.1, one obtains

$$\phi(R|\mathcal{F}) = \frac{1}{n_{\text{trees}}} \sum_{x \in R} \sum_{t=1}^{n_{\text{trees}}} \pi_{\sigma_t(x)} \tag{4.3}$$

$$= \frac{1}{n_{\text{trees}}} \sum_{x \in R} \sum_{t=1}^{n_{\text{trees}}} \sum_{L \in \mathcal{L}_t} \pi_L \mathbf{1}_{\{\sigma_t(x) = L\}}. \tag{4.4}$$


x
$t=1$
$t=2$
$t=3$
$\boldsymbol{\pi} =$
π_1 π_2 π_3 π_4 π_5 π_6 π_7 π_8 π_9 π_{10} π_{11} π_{12}
$\sigma_1(\mathbf{x})$
$\sigma_2(\mathbf{x})$
$\sigma_3(\mathbf{x})$

(a)



(b)

Figure 4.3: **Notations.** (a) Forest notations in a simplified case (3 trees of depth 2). The leaves are arbitrarily ordered and the leaf models are jointly considered as a vector $\boldsymbol{\pi}$. For each observation $x \in \mathcal{X}$, $\sigma_t(x)$ denotes the leaf reached by x when passed through the t^{th} tree. The list of functions $\Sigma = (\sigma_t)_{1\leq t\leq n_{\text{trees}}}$ encompasses information about the arrangement of the trees and the node splitting functions. (b) Characteristic vector of a region R. Applying a forest on all the pixels within a region R leads to n_{trees} different partitions of R, defined by the leaves reached by the sent observations. By counting and concatenating the leaf occurrences, one obtains a characteristic vector $\boldsymbol{\rho}(R|\Sigma)$ of the region R which only depends on the structure Σ of the forest (Eq. 4.11). Consequently, the score of any region R can be written as $\phi(R|\mathcal{F}) = \langle \boldsymbol{\rho}(R|\Sigma), \boldsymbol{\pi}\rangle$.

We defined in Sec. 4.3.1 the quantity $\text{tree}(L) \in \{1,\dots,n_{\text{trees}}\}$ as the index of the tree to which a leaf $L \in \mathcal{L}$ belongs. Following this definition, we have for all trees $t \in \{1,\dots,n_{\text{trees}}\}$ and leaves $L \in \mathcal{L}_t$ the equality $t = \text{tree}(L)$. Thus

$$\sum_{t=1}^{n_{\text{trees}}} \sum_{L\in\mathcal{L}_t} \pi_L \mathbf{1}_{\{\sigma_t(x)=L\}} = \sum_{t=1}^{n_{\text{trees}}} \sum_{L\in\mathcal{L}_t} \pi_L \mathbf{1}_{\left\{\sigma_{\text{tree}(L)}(x)=L\right\}} \tag{4.5}$$

$$= \sum_{L\in\mathcal{L}} \pi_L \mathbf{1}_{\left\{\sigma_{\text{tree}(L)}(x)=L\right\}} \tag{4.6}$$

since the double sum $\sum_{t=1}^{n_{\text{trees}}} \sum_{L\in\mathcal{L}_t}$ amounts to summing over all leaves in the forest. Finally, by incorporating Eq. 4.6 in Eq. 4.4, we obtain

$$\phi(R|\mathcal{F}) = \frac{1}{n_{\text{trees}}} \sum_{x\in R} \sum_{L\in\mathcal{L}} \pi_L \mathbf{1}_{\left\{\sigma_{\text{tree}(L)}(x)=L\right\}} \tag{4.7}$$

$$= \sum_{L\in\mathcal{L}} \frac{\pi_L}{n_{\text{trees}}} \sum_{x\in R} \mathbf{1}_{\left\{\sigma_{\text{tree}(L)}(x)=L\right\}} \tag{4.8}$$

$$= \sum_{L\in\mathcal{L}} \pi_L \rho_L(R|\Sigma) \tag{4.9}$$

$$= \langle \boldsymbol{\rho}(R|\Sigma), \boldsymbol{\pi}\rangle \tag{4.10}$$

where $\boldsymbol{\rho}(R|\Sigma) = (\rho_L(R|\Sigma))_{L\in\mathcal{L}}$ is a vector of dimension n_{leaves} characterizing the region R and defined as

$$\rho_L(R|\Sigma) = \frac{1}{n_{\text{trees}}} \underbrace{\#\left\{x \in R \mid \sigma_{\text{tree}(L)}(x) = L\right\}}_{\text{number of pixels in } R \text{ falling in the leaf } L} . \tag{4.11}$$

Hence, the scoring function of a region R appears as a scalar product between the vector of leaf models $\boldsymbol{\pi}$ and a vector $\boldsymbol{\rho}(R|\Sigma)$, which only depends on how the samples from the region R are sent to the leaves (Fig. 4.3b). Moreover, since every observation x contained in a region R falls in exactly n_{trees} leaves, each vector $\boldsymbol{\rho}(R|\Sigma)$ is sparse (or of small size) with at most $n_{\text{trees}}\,|R|$ non-zero elements.

4.3.2 Interactive Forest Adaptation

In Sec. 4.3.1, we described how to score regions of a large histological slide so that they can be ranked and displayed in decreasing order of interest to a pathologist. This ranking is based on the output of a pixelwise classification forest learned on labeled data. If the data at hand differs from the training images, for instance because of variations in terms of dye concentration or because of the presence of artifacts, this initial forest model can be prone to errors (Fig. 4.4). However, the fact that regions of interest are shown sequentially to the human expert offers the opportunity to let the user report the actual validity of the suggestions and, thereby, to recalibrate the forest model to take into account the characteristics of the slide to analyze. This scenario corresponds to an online domain adaptation problem, for which we consider three different strategies. The first two require accurate delineations of the objects of interest by the user, whereas the third approach only requires a weaker form of labeling stating the actual surface covered by such objects within a suggested region. By discretizing this quantity, faster user interactions can be performed (see Sec. 4.5).

The adaptation procedure generates, starting from a forest $\mathcal{F}^1$, a series of forests $\mathcal{F}^2, \mathcal{F}^3, \ldots$ where each forest $\mathcal{F}^{k+1}$ is created after k regions have been observed by the pathologist and the k corresponding inputs collected. At each iteration k, the region R_k is chosen as the one maximizing the scoring function $\phi(.|\mathcal{F}^k)$ over the set of remaining regions. The three alternative strategies described below are based on an assumption of fixed structure for all the forests $\mathcal{F}^k$, so that only the leaf probabilities are modified. This assumption offers the following computational advantage. For all $k \geq 1$, the structure Σ^k of the forest $\mathcal{F}^k$ is now equal to the structure Σ^1 of the initial forest $\mathcal{F}^1$. In particular, the vectors $\boldsymbol{\rho}(R|\Sigma^k), R \in \mathcal{R}$ are kept unchanged during the whole exploration process, so that they can be precomputed once for all at the first iteration and compactly stored in memory due to their sparsity. For simplicity, we omit their dependency in Σ^1 and denote these vectors $\boldsymbol{\rho}(R), R \in \mathcal{R}$. The score $\phi(R|\mathcal{F}^k) = \left\langle \boldsymbol{\rho}(R|\Sigma^k), \boldsymbol{\pi}^k \right\rangle$ of a region R described in Eq. 4.10 can be rewritten $\phi(R|\mathcal{F}^k) = \left\langle \boldsymbol{\rho}(R), \boldsymbol{\pi}^k \right\rangle$. Hence, to obtain the updated scoring functions $\phi(R|\mathcal{F}^k)$ for a new forest $\mathcal{F}^k$, one only needs to recompute the sparse scalar products $\left\langle \boldsymbol{\rho}(R), \boldsymbol{\pi}^k \right\rangle$ with the new leaf models $\boldsymbol{\pi}^k$. The efficiency of this operation yields real-time updates between two region suggestions as the histological slide

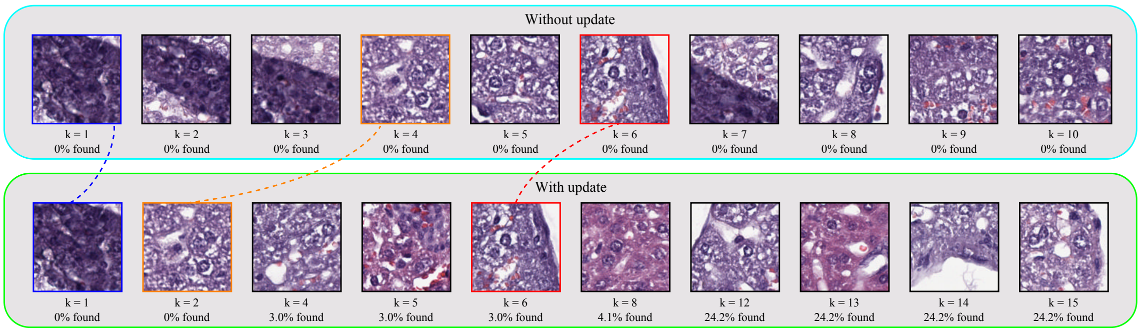


Figure 4.4: **Benefit of the interactive adaptation illustrated by the negative region suggestions.** In challenging cases (here the slide containing the portion shown in Fig. 4.1b), a direct application of the pre-trained forest model leads to difficulties in the exploration. The first 10 regions that are suggested are in fact negative (first row). These confusions are due to the variation of staining and the presence of an artifact shown in Fig. 4.1b, which confirms the necessity of adapting the initial classifier. When interactive updates are performed, each negative suggestion is signaled by the user and the underlying model is accordingly calibrated. The benefit of this adaptation scheme can be illustrated by looking at the first 10 negative suggestions (second row, using the ALM update). First, we can see that positive suggestions occur earlier in the exploration: the 3rd, 7th, 9th, 10th and 11th suggestions were positive and allowed the pathologist to localize already 24.2% of the hematopoietic patterns present in the slide. This demonstrates that adapting the classifier clearly improves the quality of the exploration. Moreover, we can see that the adaptive approach 'learns from its mistakes' through the greater diversity of its negative suggestions: regions with different shades of stain are proposed and multiple suggestions within the dark artifact are avoided. Among their 10 first negatives, only 3 suggestions are shared by the two approaches (emphasized with colors), which further illustrates their difference.

has to be passed only once through a forest (the initial $\mathcal{F}^1$) as a preliminary step before starting the exploration.

We expose now our three alternative real-time update strategies. They are all equivalent from a computational point of view, with a worst-case complexity of $O(n_{\text{trees}}\,|R|)$, and depend on one hyperparameter $\lambda > 0$ weighting the importance of the prior knowledge in comparison to the newly observed samples.

Update of Leaf Statistics (ULS) If the input provided by the pathologist is a full object delineation in the displayed region R_k, the labeled pixels of the region R_k can be seen as new training samples. Therefore, the leaf statistics can be updated [Criminisi, 2011]. We denote $N_L^{1,+}$ (resp. $N_L^{1,-}$) the number of positive (resp. negative) samples which arrived in the leaf L during the training of the initial forest $\mathcal{F}^1$, leading to the leaf models $\pi_L^1 = \frac{N_L^{1,+}}{N_L^{1,-}+N_L^{1,+}}$. We also denote $N_L^{k+1,+}$ and $N_L^{k+1,-}$ the total number of positive and negative samples collected in the regions $R_1, \ldots, R_k$. Given these quantities, the ULS strategy updates the probability of each leaf L after labeling the region R_k as

$$\pi_L^{k+1} = \frac{N_L^{k+1,+} + \lambda N_L^{1,+}}{N_L^{k+1,+} + N_L^{k+1,-} + \lambda\left(N_L^{1,+} + N_L^{1,-}\right)}. \tag{4.12}$$

Average of the Leaf Models (ALM) In the same conditions as the update of leaf statistics described above, we propose an alternative leaf update consisting in computing a separate probability π_L^{new} based on the pixels observed in $R_1, \ldots, R_k$ only (i.e. originating from the test slide) and averaging it with the initial probability π_L^1. By doing so, the choice of λ is made independent of the initial number of training samples in the leaf. This update can be written as

$$\boldsymbol{\pi}^{k+1} = \frac{1}{1+\lambda}\left(\boldsymbol{\pi}^{\text{new}} + \lambda\boldsymbol{\pi}^1\right) \tag{4.13}$$

where, for each leaf L, $\pi_L^{\text{new}} = \frac{N_L^{k+1,+}}{N_L^{k+1,+} + N_L^{k+1,-}}$ if some new samples have been observed in the leaf L (i.e. if $N_L^{k+1,+} + N_L^{k+1,-} > 0$). Else we define $\pi_L^{\text{new}} = \pi_L^1$.

Online Gradient Descent (OGD) The two previous updates require a pixelwise labeling provided by the user on each region R_k. Instead, this last update method uses a (weaker) information $\mathcal{Q}(R_k)$ which only states the amount of positive pixels located in R_k. This quantity is, in fact, what is estimated by the score $\phi(R_k|\mathcal{F}^k) = \left\langle \boldsymbol{\rho}(R_k), \boldsymbol{\pi}^k \right\rangle$ used to assess the relevance of the region R_k (Sec. 4.3.1). We propose to measure the discrepancy between the true value $\mathcal{Q}(R_k)$ revealed by the user and the prediction from the set of leaf models $\boldsymbol{\pi}$ with the squared loss $l_k(\boldsymbol{\pi}) = (\langle \boldsymbol{\rho}(R_k), \boldsymbol{\pi} \rangle - \mathcal{Q}(R_k))^2$. Hence, at iteration k, the incurred loss is $l_k(\boldsymbol{\pi}^k)$. The convexity of the loss function l_k, which directly results from the linear rewriting of our scoring function (Eq. 4.10), allows us to see the update problem as an online convex optimization scenario [Shalev-Shwartz, 2012]. We solve this problem via an online gradient descent strategy [Zinkevich, 2003], which leads to the update rule

$$\boldsymbol{\pi}^{k+1} = \Pi_{[0,1]^{|\mathcal{L}|}}\left[\boldsymbol{\pi}^k - \eta\vec{\nabla} l_k(\boldsymbol{\pi}^k)\right] \tag{4.14}$$

$$= \Pi_{[0,1]^{|\mathcal{L}|}}\left[\boldsymbol{\pi}^k - 2\eta\left(\left\langle \boldsymbol{\rho}(R_k), \boldsymbol{\pi}^k \right\rangle - \mathcal{Q}(R_k)\right)\boldsymbol{\rho}(R_k)\right], \tag{4.15}$$

where η is a learning rate. $\Pi_{[0,1]^{|\mathcal{L}|}} : \mathbb{R}^{|\mathcal{L}|} \to [0,1]^{|\mathcal{L}|}$ is the projection operator on $[0,1]^{|\mathcal{L}|}$ which projects each individual component π_l of a vector $\boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{L}|}$ onto the set $[0,1]$, ensuring that the leaf probabilities stay in $[0,1]$ after each update. By transferring generic considerations on online gradient descent to our forest-based scenario (see below), we choose a learning rate η of the form

$$\eta = \frac{1}{2\lambda\Delta^4}\sqrt{\frac{n_{\text{trees}}\,|\mathcal{L}|}{|\mathcal{R}|}}, \tag{4.16}$$

where λ is a positive hyperparameter, $|\mathcal{L}|$ (resp. n_{trees}) is the number of leaves (resp. trees) in the forest, $|\mathcal{R}|$ is the number of regions in the slide and $\Delta \times \Delta$ is the predefined region size.

The difference between ALM and ULS can be seen by considering their respective asymptotic behavior when the number of new samples increases. In Eq. 4.12, we have $\boldsymbol{\pi}^k \sim \boldsymbol{\pi}^{\text{new}}$, whereas, in Eq. 4.13, $\boldsymbol{\pi}^k$ always includes a fixed contribution from $\boldsymbol{\pi}^1$ regardless of the number of new samples which have been collected. These two variants correspond to the simplest way of updating an existing tree [Criminisi, 2011], where we introduce a parameter weighting old and new training data. More sophisticated but computationally costly strategies would involve further splitting or the replacement of old trees by new ones [Saffari et al., 2009]. In our case, since the original training data used to train $\mathcal{F}^1$ are no longer available at testing time, we keep the structure of the old trees which represent the only available prior knowledge about the quantification task. This emphasizes the scope of our adaptation procedure, which should be seen as adjusting a known supervised segmentation task (e.g. within a same clinical study) to the variations of visual appearances that may occur experimentally. However, a different task cannot be accommodated a priori and would first require the training of a new segmentation model. Finally, due to the nature of the required input, OGD can be used for a lighter kind of user interaction (see Sec. 4.5), which is not supported by existing online learning algorithms for random forests.

Choice of Learning Rate (Eq. 4.16) Finally, we conclude with the theoretical considerations leading to the form of the learning rate exposed in Eq. 4.16. We follow a classical reasoning inspired from the online learning literature [Nemirovski et al., 2009, Shalev-Shwartz, 2012] and show how it relates to our scenario by expressing bounds in terms of the parameters of our method. As described above, at each iteration $k \geq 1$, the current set of leaf models $\boldsymbol{\pi}^k$ suffers the loss $l_k(\boldsymbol{\pi}^k) = \left(\left\langle \boldsymbol{\rho}(R_k), \boldsymbol{\pi}^k \right\rangle - \mathcal{Q}(R_k)\right)^2$, after which a new vector of leaf models $\boldsymbol{\pi}^{k+1}$ is chosen according to the online gradient descent update rule (Eq. 4.14). After T region suggestions ($T \geq 1$), the cumulated regret of having used the series of models $\boldsymbol{\pi}^1, \ldots, \boldsymbol{\pi}^T$ is defined as

$$\text{Regret}_T = \sum_{k=1}^{T} \left(l_k(\boldsymbol{\pi}^k) - l_k(\boldsymbol{\pi}^*)\right), \tag{4.17}$$

where

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi} \in [0,1]^{|\mathcal{L}|}}{\text{argmin}} \sum_{k=1}^{T} l_k(\boldsymbol{\pi}) \tag{4.18}$$

corresponds to the set of leaf models which would have incurred the smallest loss over the T iterations. The reasoning consists in computing an upper bound of Regret_T depending on the learning rate η. To do so, we use the fact that the functions l_k are convex, so that, for all k, we have $l_k(\boldsymbol{\pi}^k) \leq l_k(\boldsymbol{\pi}^*) + \left\langle \boldsymbol{\pi}^k - \boldsymbol{\pi}^*, \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\rangle$ and thus

$$\text{Regret}_T \leq \sum_{k=1}^{T} \left\langle \boldsymbol{\pi}^k - \boldsymbol{\pi}^*, \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\rangle. \tag{4.19}$$

To find an upper bound of $A_k = \left\langle \boldsymbol{\pi}^k - \boldsymbol{\pi}^*, \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\rangle$, we use the update rule of Eq. 4.14 as follows. For all k, denoting $\Pi = \Pi_{[0,1]^{|\mathcal{L}|}}$ and $D_k = \left\| \boldsymbol{\pi}^k - \boldsymbol{\pi}^* \right\|$, we have

$$
\begin{aligned}
D_{k+1}^2 &= \left\| \boldsymbol{\pi}^{k+1} - \boldsymbol{\pi}^* \right\|^2 && (4.20)\\
&= \left\| \Pi \left[\boldsymbol{\pi}^k - \eta \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right] - \boldsymbol{\pi}^* \right\|^2 && (4.21)\\
&= \left\| \Pi \left[\boldsymbol{\pi}^k - \eta \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right] - \Pi \left[\boldsymbol{\pi}^* \right] \right\|^2 && (4.22)\\
&\leq \left\| \boldsymbol{\pi}^k - \eta \vec{\nabla} l_k(\boldsymbol{\pi}^k) - \boldsymbol{\pi}^* \right\|^2 && (4.23)\\
&= D_k^2 - 2\eta A_k + \eta^2 \left\| \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\|^2, && (4.24)
\end{aligned}
$$

which leads to the inequality

$$
A_k \leq \frac{1}{2\eta} \left(D_k^2 - D_{k+1}^2 + \eta^2 \left\| \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\|^2 \right). \tag{4.25}
$$

The inequality between Eq. 4.22 and Eq. 4.23 results from the fact that, in the Hilbert space $\mathbb{R}^{|\mathcal{L}|}$, performing a projection on the closed convex set $[0,1]^{|\mathcal{L}|}$ does not increase the distance between two points. Using Eq. 4.25 in Eq. 4.19, we obtain

$$
\begin{aligned}
\text{Regret}_T &\leq \frac{1}{2\eta} \sum_{k=1}^{T} \left(D_k^2 - D_{k+1}^2 + \eta^2 \left\| \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\|^2 \right) && (4.26)\\
&= \frac{1}{2\eta} \left[D_1^2 - D_{T+1}^2 + \eta^2 \sum_{k=1}^{T} \left\| \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\|^2 \right] && (4.27)\\
&\leq \frac{1}{2\eta} D_1^2 + \frac{\eta}{2} \sum_{k=1}^{T} \left\| \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\|^2. && (4.28)
\end{aligned}
$$

To obtain a final bound on the regret, we need to find an upper bound of D_1^2 and of the norm of the gradient $\left\| \vec{\nabla} l_k(\boldsymbol{\pi}^k) \right\|^2$. First, since both $\boldsymbol{\pi}^1$ and $\boldsymbol{\pi}^*$ belong to $[0,1]^{|\mathcal{L}|}$, we have

$$
D_1^2 = \left\| \boldsymbol{\pi}^1 - \boldsymbol{\pi}^* \right\|^2 \leq |\mathcal{L}|. \tag{4.29}
$$

Secondly, for all k and $\boldsymbol{\pi}$, we have

$$
\vec{\nabla} l_k(\boldsymbol{\pi}) = 2 \left(\langle \boldsymbol{\rho}(R_k), \boldsymbol{\pi} \rangle - \mathcal{Q}(R_k) \right) \boldsymbol{\rho}(R_k). \tag{4.30}
$$

The quantity $\langle \boldsymbol{\rho}(R_k), \boldsymbol{\pi} \rangle$ estimates the surface covered by positive pixels in the region R_k, while $\mathcal{Q}(R_k)$ is the actual value of this surface revealed by the user. Since, by definition, both $\left\langle \boldsymbol{\rho}(R_k), \boldsymbol{\pi}^k \right\rangle$ and $\mathcal{Q}(R_k)$ are comprised between 0 and the size Δ^2 of the region, we have $\left| \left\langle \boldsymbol{\rho}(R_k), \boldsymbol{\pi}^k \right\rangle - \mathcal{Q}(R_k) \right| \leq \Delta^2$. By definition of $\boldsymbol{\rho}$ (see Eq. 4.11), we also know that each individual component ρ_L does not exceed $\frac{\Delta^2}{n_{\text{trees}}}$ (since at most the number of pixels in the region Δ^2 can fall in a leaf L), and, moreover, that these components sum to Δ^2.

Thus

$$\|\boldsymbol{\rho}(R_k)\|^2 = \sum_{L\in\mathcal{L}} \rho_L^2(R_k) \tag{4.31}$$

$$\leq \frac{\Delta^2}{n_{\text{trees}}} \sum_{L\in\mathcal{L}} \rho_L(R_k) \tag{4.32}$$

$$= \frac{\Delta^4}{n_{\text{trees}}}, \tag{4.33}$$

hence the following upper bound on the gradient:

$$\left\|\vec{\nabla} l_k(\boldsymbol{\pi}^k)\right\|^2 \leq 4\frac{\Delta^8}{n_{\text{trees}}}. \tag{4.34}$$

Finally, including Eq. 4.29 and Eq. 4.34 in Eq. 4.28 gives the bound

$$\text{Regret}_T \leq \frac{|\mathcal{L}|}{2\eta} + \frac{2\eta T \Delta^8}{n_{\text{trees}}}. \tag{4.35}$$

We choose the value of η providing the best regret bound, i.e. minimizing the right side of Eq. 4.35. This is obtained for

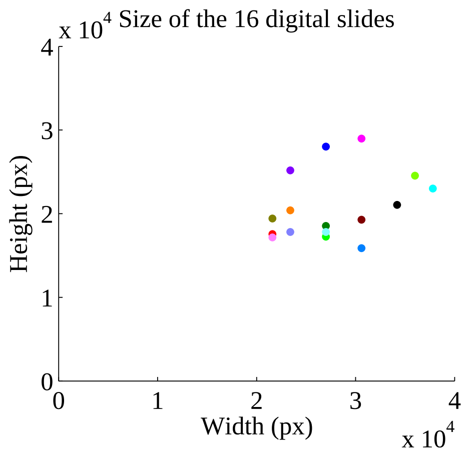
$$\eta = \frac{1}{2\Delta^4}\sqrt{\frac{n_{\text{trees}}\,|\mathcal{L}|}{T}}. \tag{4.36}$$

While the relevant number of iterations T for the practical applicability of our scenario is unknown, it should intuitively be proportional to the size of the test slide, and thus to the number of regions $|\mathcal{R}|$. This leads us to define $T = \lambda^2\,|\mathcal{R}|$ as proportional to this quantity, resulting in Eq. 4.16, and learn the hyperparameter λ on a validation set.

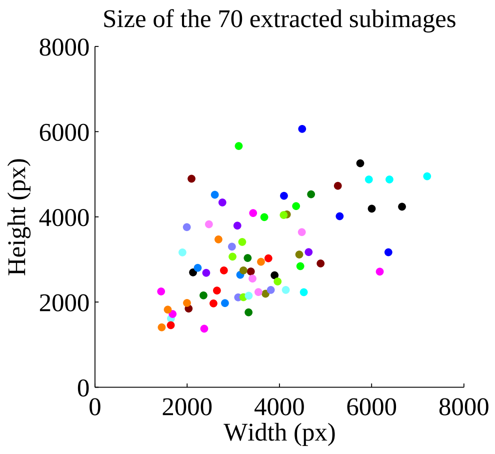
4.3.3 Experimental Evaluation

Dataset

Extramedullary hematopoiesis, i.e. the presence of hematopoietic cells outside the bone marrow, is a marker of an extensive stimulation of the immune system [Tao et al., 2008]. There is accumulating evidence that the amount of infiltrating immune cells such as cytotoxic CD8-positive T-lymphocytes into the tumor can be considered as a tumor biomarker for measuring clinical outcome [Balermpas et al., 2016]. We evaluated our approach in this clinical context on a dataset addressing the aspect of lymphocytic infiltration into mouse liver tissues, for which the amount of these cells within histological samples must be estimated. Slides from 16 mice were digitally acquired at the resolution 0.5 μm per pixel and downsized by 2 to speed up the training and testing steps. 70 large representative subimages were extracted from these slides and fully segmented, covering approximately 20% of the total tissue (Fig. 4.5). Resorting to a set of subimages follows the clinical practice and was necessary to obtain accurate labels for a sufficient number of different slides. This is particularly important in our study which focuses on issues arising from the visual variability between acquisitions. Yet, it comes at the cost of possibly introducing a few natural biases such as underrepresentation of border areas or of straightforwardly negative objects (e.g. large white parts).



(a)



(b)

Figure 4.5: **Dataset dimensions.** Our dataset consists of 16 digital slides, an example of which is provided in Fig. 4.1a. Instead of working directly on the slides, we extracted and labeled entirely a total of 70 subregions such as the one shown in Fig. 4.1b. (a) Dimensions of the slides, where each colored point corresponds to one slide. (b) Dimensions of the extracted subregions. The color code of (a) is respected so that the slide from which each subimage is extracted can be identified by its color. In total, the extracted subimages cover around 20% of the acquired tissues.

Experimental Settings

We randomly split our dataset $\mathcal{D}$ into 4 sets $\mathcal{D}_i$ of 4 slides each and we performed a 4-fold nested cross-validation. The goal of this procedure is to optimize the update-related hyperparameter λ independently of the test data to avoid overfitting. Let Λ be a set of candidate values for λ and $l^{\text{cv}}(\lambda, \mathcal{F}^1, I)$ a loss measuring the error of our method on the slide I when using the hyperparameter λ and an initial forest $\mathcal{F}^1$. The nested cross-validation consists of 4 runs, each run corresponding to a set $\mathcal{D}_{i_{\text{out}}}$ (with $i_{\text{out}} \in \{1,2,3,4\}$) left out for testing. For each run, a second cross-validation (called inner cross-validation) is performed over the 3 remaining sets $(\mathcal{D}_i)_{i \neq i_{\text{out}}}$, where 2 sets are used to train the forest and the remaining one is used as a validation set. At the end of the inner cross-validation, i.e. when 3 forests have been trained and each of the 3 sets has been used as a validation set, we define the optimal hyperparameter $\lambda_{i_{\text{out}}}$ of this run as the one minimizing the total loss over the 3 bags, i.e.

$$\lambda_{i_{\text{out}}} = \operatorname*{argmin}_{\lambda \in \Lambda} \sum_{i \neq i_{\text{out}}} \sum_{I \in \mathcal{D}_i} l^{\text{cv}}(\lambda, \mathcal{F}_{\mathcal{D} \setminus \left(\mathcal{D}_i \cup \mathcal{D}_{i_{\text{out}}}\right)}, I). \tag{4.37}$$

$\mathcal{F}_{\mathcal{D} \setminus \left(\mathcal{D}_i \cup \mathcal{D}_{i_{\text{out}}}\right)}$ denotes the forest obtained by training on the two remaining sets after excluding $\mathcal{D}_i$ and $\mathcal{D}_{i_{\text{out}}}$. Using the hyperparameter value $\lambda_{i_{\text{out}}}$, we then report independently the prediction of each of the 3 forests $\left(\mathcal{F}_{\mathcal{D} \setminus \left(\mathcal{D}_i \cup \mathcal{D}_{i_{\text{out}}}\right)}\right)_{i \neq i_{\text{out}}}$ on the left-out set $\mathcal{D}_{i_{\text{out}}}$. This procedure allows us to learn automatically the hyperparameter independently of the testing set, and outputs 3 different predictions for each test slide which gives an idea of their dependency on the original training data. This results in a total of 48 predictions. Note that, to conduct the entire nested cross-validation, only 6 forests $\left(\mathcal{F}_{\mathcal{D}_i \cup \mathcal{D}_j}\right)_{1 \leq i < j \leq 4}$ have to be trained. The user interaction was automatically simulated from the ground truth delineations. Regions were chosen of size $\Delta \times \Delta$ with $\Delta = 60\ \mu\mathrm{m}$. Every time a region is displayed, the user can easily extend the field of view around it if necessary. We simulated this behavior automatically by showing the neighboring positive region(s) in the case where an object of interest is not fully included in the displayed region.

The forests were initially trained on labeled pixels which were densely collected every $8\ \mu\mathrm{m}$ in the two directions. Parallelized on 10 threads, this training step took between 3 and 6 hours depending on the cross-validation run (with a corresponding number of training samples comprised between 5×10^5 and 10^6). Given an incoming slide, testing was performed on all pixels, which is tractable since it has to be done only once at the beginning of the process (see Sec. 4.3.2). This preliminary step took around 1 minute, after which the interaction loop could take place in real-time conditions. More precisely, the update of forest leaf models and recomputation of box scores between two iterations took between 10 and $100\ \mathrm{ms}$ without any parallelization. The visual features were computed on the Lab color space. The following forest parameters were used: $n_{\text{samples/leaf}} = 10$, $n_{\text{trees}} = 30$, $n_{\text{tries}} = 500$, $n_{\text{thresholds}} = 10$, and the bagging rate was 0.5.

Due to the efficiency of Haar-like features, our segmentation algorithm is able to work directly at the highest level of magnification, processing approximately 2.0×10^7 pixels per minute. This order of magnitude is, for instance, the same as in a recent boosting-based hierarchical segmentation approach [Doyle et al., 2012] which analyzes around 1.4×10^7 pixels in less than 3 minutes (with parallelization on 2 threads instead of 10). However, this latter work used more complex features for their application, hence justifying a hierarchical strategy.

Results

We studied the ability of our approach to retrieve regions of interest as quickly as possible within large slides. The following alternatives were compared:

- the simplest forest-based exploration approach without update from the pathologist (`No Update (Forest)`), i.e. only relying on the pre-trained forest $\mathcal{F}^1$,
- the three update strategies (`ULS`, `ALM` and `OGD`) exposed in Sec. 4.3.2,
- a baseline showing, at each iteration, a region randomly (uniformly) drawn among the remaining regions (`Random exploration`),
- an oracle whose scoring function is extracted from the ground truth, hence serving as a gold standard showing the highest achievable performance (`Oracle`).

In addition, to position the forest-based performance among other classification methods, we trained an AdaBoost classifier (Sec. 1.2.7) and used it as segmentation model instead of the forest (`No Update (AdaBoost)`). The chosen weak classifiers were decision stumps based on Haar-like features such as the splitting functions stored in tree nodes.

100 boosting iterations were conducted, and 500 stumps tried at each iteration. These design choices led to a training time similar to the forest one.

To assess quantitatively the performance of each method, we consider the curve showing the proportion of positive pixels that have been displayed after having shown a certain percentage of the slide to the pathologist (Fig. 4.6a). A good exploration method is expected to lead to a curve converging quickly towards 1. We summarize quantitatively the performance on a slide I by computing the area $A(\lambda, \mathcal{F}^0, I)$ under this curve. The nested cross-validation procedure described in Sec. 4.3.3 was accordingly performed using the loss function $l^{\text{cv}}(\lambda, \mathcal{F}^1, I) = 1 - A(\lambda, \mathcal{F}^1, I)$ and optimizing λ over a logarithmic grid. The statistical distribution of the area under curves obtained at prediction time for each method are shown in Fig. 4.6b. We performed statistical pairwise comparisons between methods by conducting paired Wilcoxon's signed-rank tests over these values. To maintain the independence between samples, we repeated each test 100 times retaining at random one of the 3 runs for each slide and considered the median p-value over these 100 runs. Denoting Method $1 \prec$ Method 2 the fact that Method 2 is significantly better than Method 1 and Method $1 \approx$ Method 2 the absence of demonstrated statistical difference between the two methods, the series of tests provided the following ranking:

$$\text{Random} \prec \text{No Update (AdaBoost)} \prec \text{No Update (Forest)} \prec \text{OGD} \prec \text{ULS} \approx \text{ALM} \prec \text{Oracle}.$$

All p-values showing statistical difference were lower than 10^{-3}, and the p-value obtained when comparing ULS and ALM was 0.5. This ranking confirms what was intuitively expected. The three methods proposing a model update from the user inputs improve over a non-interactive exploration, and the two methods using accurate pixelwise labelings outperform the online gradient descent technique which is based on a weaker but lighter type of information. We also see, from the performance of a random exploration, that using a pre-trained forest drastically helps finding relevant objects more quickly. Note that, in theory, one might have expected a straight '$y = x$' line for the random exploration. In fact, when a suggested region belongs to a larger object, the user extends the field of view to see the object in its totality. Hence, a positive suggestion may be immediately followed by other positive ones due to the user intervention. This bias explains why, in spite of a random exploration, one obtains a slightly 'better than random' curve.

Finally, the impact on each method of the update-related parameter λ was assessed experimentally (Fig. 4.6c). As expected, when $\lambda \to \infty$, the three methods converge towards the method without update. The choice $\lambda = 0$ leads to a nearly maximal performance for the two update strategies based on pixelwise labelings (ULS and ALM). Since, moreover, these two methods are equivalent for $\lambda = 0$, they behave very similarly after optimization on a validation set, as observed on Fig. 4.6a and Fig. 4.6b. Note that choosing $\lambda = 0$ does not mean that the initial forest $\mathcal{F}^1$ is ignored. The prior knowledge from this initial model is taken into account via the tree structures and their leaf models, which remain unchanged as long as no sample from the test data reaches the leaf in question.

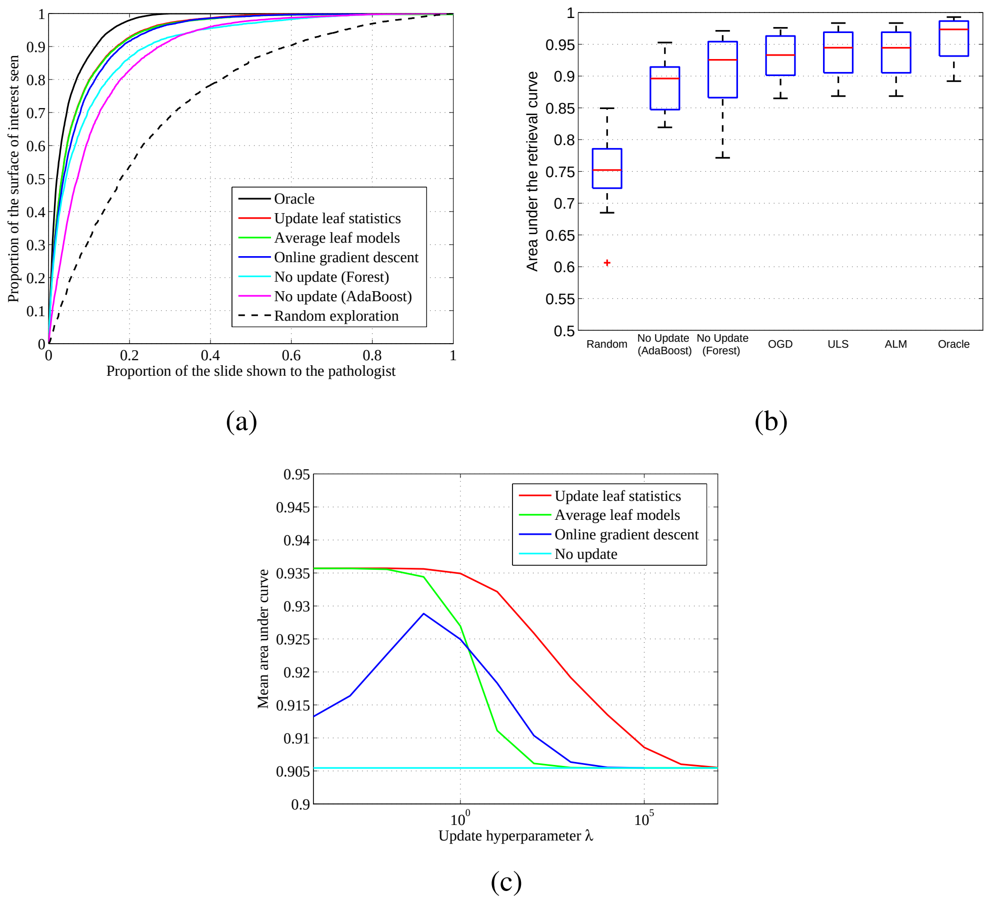


Figure 4.6: **Experimental validation of the slide exploration stage.** (a) For each method, we plot the mean retrieval curve which shows the proportion of positive pixels seen by the pathologist after having seen a certain proportion of the slide. We use the area under these curves to measure quantitatively the slide exploration abilities. (b) Statistical distribution of the area under the curve for each method. Each box plot is computed over the 48 measurements obtained during the nested cross-validation. (c) Influence of the hyperparameter λ on the performance. For each of the three update strategies, we studied the behavior of the mean area under the curve when λ varies, i.e. without optimization on a validation set.

4.4 Whole-Slide Quantification from Partial Exploration

So far, we have introduced and evaluated a method to assist a pathologist in the exploration of a large digital slide by retrieving and displaying regions that presumably contain objects of interest. We now propose a stopping criterion for the exploration process and introduce an approach which, once the exploration stage is completed, extrapolates the content observed in the displayed regions to assess the surface covered by the objects of interest in the entire slide (including the unseen areas).

4.4.1 Stopping the Exploration Stage

In practice, the exploration process is meant to be interrupted before seeing the whole slide. Since the amount of hematopoietic cell clusters is variable from a slide to another, some slides intrinsically require more time from the pathologist than others. Therefore, fixing in advance the number of iterations for the exploration would be inappropriate. Instead, we propose a stopping criterion based on the density of positive suggestions, and interrupt the exploration as soon as n_{stop} negative regions were suggested in a row, i.e. when most positive regions were presumably seen. Once the stopping criterion is reached, a whole-slide prediction can be made (Sec. 4.4.2).

The chosen value for n_{stop} depends on the amount of time that the pathologist is ready to spend for the analysis. Strictly reasoning in terms of accuracy, it is always preferable to let the pathologist see a maximum number of regions. Defining a recommended value for n_{stop} is hence subjective and results from a tradeoff between accuracy and human effort. Our experiments regarding the whole-slide quantification were conducted for several values of n_{stop}, encoding different amounts of effort that the pathologist is ready to invest.

4.4.2 Whole-Slide Quantification via Regression

Once the stopping criterion has been reached, we propose to predict an estimate $\hat{q}$ of the surface covered by hematopoietic cells within the whole slide with linear regression. Denoting K the total number of regions that have been seen during the exploration stage, a partial knowledge on $\hat{q}$ is available via the quantity $q^{\text{labeled}} = \sum_{k=1}^{K} \mathcal{Q}(R_k)$ obtained as the user annotated the regions $R_1, \ldots, R_K$ during the exploration. In addition, the updated forest model $\mathcal{F}^{K+1}$ obtained at the end of the exploration phase provides a prediction $\phi(R|\mathcal{F}^{K+1})$ of the quantity of positive pixels in each region $R \in \mathcal{R}$ of the slide. In particular, this gives a total prediction $\Phi^{\text{total}} = \sum_{R\in\mathcal{R}} \phi(R|\mathcal{F}^{K+1})$ and a prediction on the labeled regions $\Phi^{\text{labeled}} = \sum_{k=1}^{K} \phi(R_k|\mathcal{F}^{K+1})$. We formalize our regression problem by considering that the relative change between the total quantity $\hat{q}$ and the partial quantity q^{labeled} is proportional to the relative change between total prediction and the partial prediction, i.e.

$$\frac{\hat{q} - q^{\text{labeled}}}{q^{\text{labeled}}} \propto \frac{\Phi^{\text{total}} - \Phi^{\text{labeled}}}{\Phi^{\text{labeled}}}. \tag{4.38}$$

This corresponds to a prediction rule of the form

$$\hat{q} = q^{\text{labeled}} + a\frac{\Phi^{\text{total}} - \Phi^{\text{labeled}}}{\Phi^{\text{labeled}}} q^{\text{labeled}}. \tag{4.39}$$

The regression parameter $a \in \mathbb{R}$ is learned on a validation set.

4.4.3 Evaluation

To assess, at prediction time, the accuracy of a list of n_{pred} estimates $(\hat{q}_i)_{1 \leq i \leq n_{\text{pred}}}$ given the corresponding true quantities $(q_i)_{1 \leq i \leq n_{\text{pred}}}$, we use the root-mean-square deviation

$$\text{RMSD} = \sqrt{\frac{1}{n_{\text{pred}}} \sum_{i=1}^{n_{\text{pred}}} (q_i - \hat{q}_i)^2}. \tag{4.40}$$

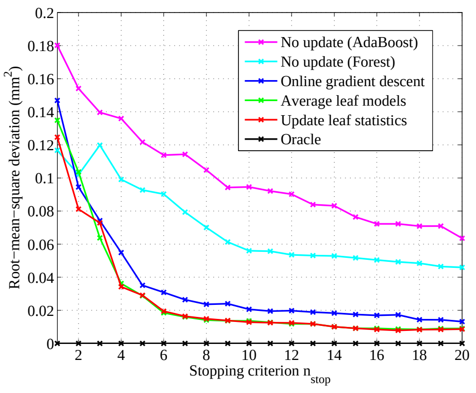
In our case, $n_{\text{pred}} = 48$. This corresponds to the 3 predictions obtained for each of the 16 slides during the cross-validation.

The experimental evaluation of the whole-slide quantification abilities was conducted as follows. We kept the cross-validation setup described in Sec. 4.3.3 and learned the hyperparameter a on a validation set, as was done for the update parameter λ, using here a squared loss $l^{\text{cv}}(a, \mathcal{F}^1, I) = (q_I - \hat{q}_I)^2$ between true and predicted whole-slide estimates. This procedure was performed independently for several values of the stopping criterion n_{stop}. We report the resulting curves in Fig. 4.7a and an example of the correspondence between estimates and true quantities in Fig. 4.7d. The ranking of methods obtained while studying the exploration abilities is preserved, due to the fact that the quality of the exploration phase is directly linked to the amount of regions which are eventually labeled. Asymptotically, if all regions containing positive samples are labeled, choosing $a = 0$ provides a perfect prediction.

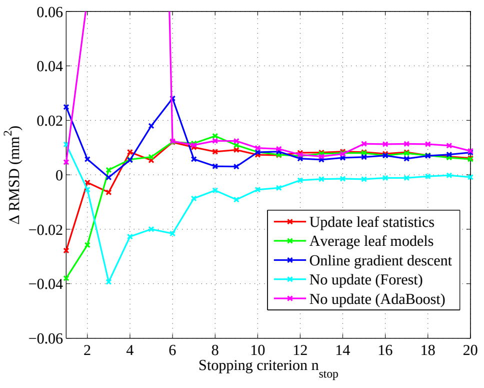
During our experiments, we observed that the sum of segmentation probabilities over the whole slide, i.e. predicting $\hat{q} = \Phi^{\text{total}}$ (with the notation of Eq. 4.39), does not form a reliable whole-slide quantification and overestimates the quantity of positive pixels due to two effects. First, the random nature of trees leads in general to small nonzero probabilities on negative pixels. When summed over all pixels, these small errors aggregate. Moreover, since we mainly retrieve positive examples during the slide exploration, the distribution of incoming samples for the update is strongly biased towards positive instances. These difficulties motivate the use of a regression. In Fig. 4.7b, we show that our regression approach (Eq. 4.39) outperforms a regression based on the user inputs alone of the form $\hat{q} = (1 + a)q^{\text{labeled}}$. This demonstrates that, in spite of its global overestimation, the forest estimate can be effectively exploited by a regression procedure.

4.5 Input Discretization for Lighter Interactions

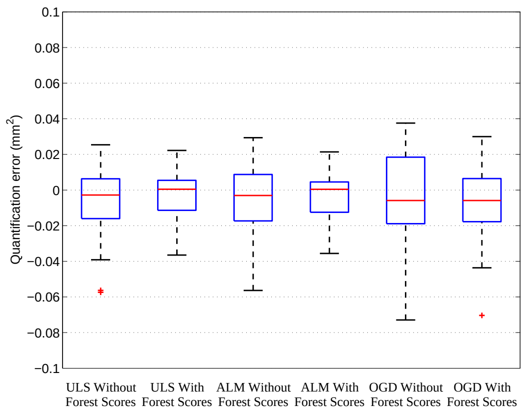
In the experiments presented in Sec. 4.3.3 and Sec. 4.4.3, the forest adaptation techniques assumed that the pathologist provides a full pixelwise labeling of the objects of interest



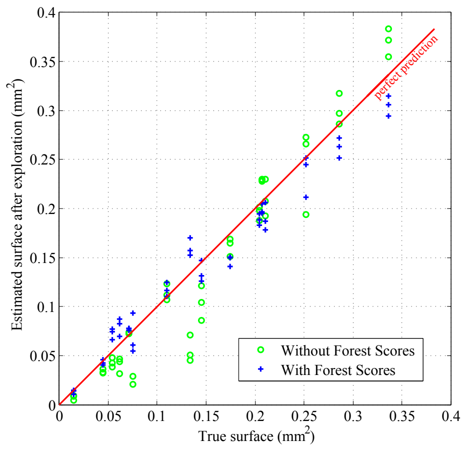
(a)



(b)



(c)



(d)

Figure 4.7: **Experimental validation of the whole-slide quantification stage.** (a) We report the root-mean-square deviation for several values of the stopping criterion n_{stop} which represents the amount of interaction provided by the pathologist. (b) Difference ΔRMSD between the root-mean-square deviation obtained when using a regression of the form $\hat{q} = (1 + a)q^{\text{labeled}}$ based on the user inputs alone, and the one obtained with the regression of Eq. 4.39 whose results are reported in Fig. 4.7a. Positive values of ΔRMSD correspond to a gain of accuracy when including the forest scores Φ^{total} and Φ^{labeled} in the regression task. As soon as n_{stop} is large enough to obtain stable results, the information carried by an updated forest improves the performance. (c) Distribution of the signed quantification error over slides for $n_{\text{stop}} = 10$, with and without including the forest scores in the regression. (d) Correlation between estimated and true hematopoietic surface within whole slides. We show an overview on the whole dataset of the whole-slide estimates after exploration. Each slide tested during the cross-validation appears 3 times corresponding to different initial forests (Sec. 4.3.3). Perfect predictions would lie on the red line. This example was obtained using the ULS adaptation for $n_{\text{stop}} = 6$, leading to RMSD $= 1.9 \times 10^{-2}$ mm^2. The median proportion of the slide which was seen during the exploration was 5.2% in this case. The predictions obtained without using the final forest scores (see Fig. 4.7b and Fig. 4.7c) are also reported.

in the displayed regions. In this section, we demonstrate how our online gradient descent scheme can instead be used with one-click inputs without decreasing its performance, thereby allowing faster user interaction.

4.5.1 One-Click User Inputs

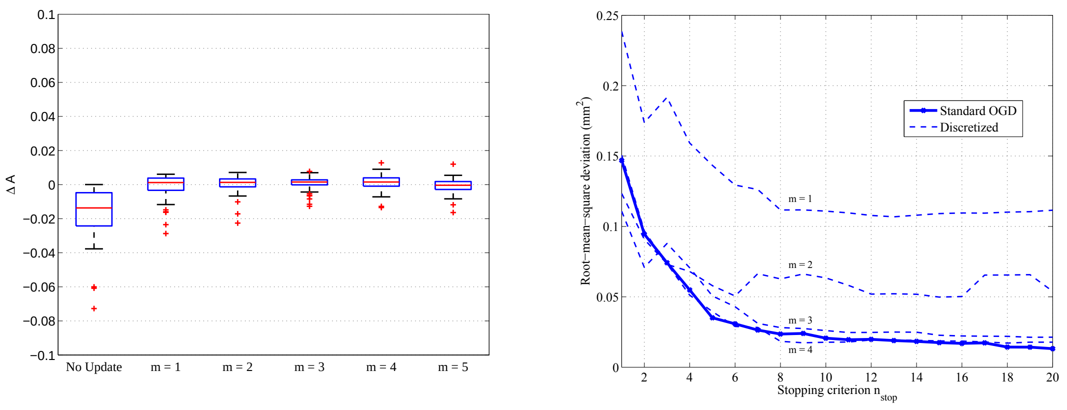
Unlike the two other techniques based on individual labels for each pixel in a region, the OGD forest update employs as user input for a region R_k the amount of positive pixels $\mathcal{Q}(R_k)$ contained in R_k. This is a different kind of input, which can be inferred from a delineation or communicated directly instead. Here, we propose to discretize the input values into bins and ask the user to select the bin to which the proportion of positive pixels belongs. This interaction is performed with only one click, or possibly without a mouse (e.g. via voice recognition).

Formally, the user annotations are discretized as follows. Instead of providing the exact quantity $\mathcal{Q}(R_k)$, the user simply indicates an interval within which the proportion $\tilde{\mathcal{Q}}(R_k) = \frac{\mathcal{Q}(R_k)}{|R_k|}$ lies. The list of available ranges is predefined as $\{0\}$, $]0; \frac{1}{m}]$, ..., $]\frac{m-2}{m}; \frac{m-1}{m}]$, $]\frac{m-1}{m}; 1[$, $\{1\}$, where m is a positive integer which encodes the fineness of the quantization. Accordingly, by taking the middle-value of each bin, the approximate proportions $\tilde{\mathcal{Q}}(R_k)$ provided by the user take their values in the finite set $D_m = \left\{0, \frac{1}{2m}, \frac{3}{2m}, \ldots, \frac{2m-1}{2m}, 1\right\}$. This gives in total $m + 2$ input possibilities for the discrete input $\tilde{\mathcal{Q}}(R_k)$, including the 2 trivial ones corresponding to an empty ($\tilde{\mathcal{Q}}(R_k) = 0$) or a full ($\tilde{\mathcal{Q}}(R_k) = 1$) region . By doing so, only one click per region is required from the pathologist, resulting in lighter interactions.

Given a discrete region label $\tilde{\mathcal{Q}}(R_k) \in D_m$ provided by the user, we have to compute the actual quantity $\mathcal{Q}(R_k)$ eventually used in the adaptation process (Eq. 4.15) and recorded for an eventual whole-slide quantification (Eq. 4.39). The simplest idea consists in directly taking $\mathcal{Q}(R_k) = |R_k| \, \tilde{\mathcal{Q}}(R_k)$, but has the drawback of losing information due to the discretization. To attenuate this aspect, we propose to perform updates only if the forest estimate $\phi(R_k|\mathcal{F}^k)$ (whose objective is to predict the quantity $\mathcal{Q}(R_k)$) deviates too strongly from the user label. More precisely, we define $\mathcal{Q}(R_k) = |R_k| \, \tilde{\mathcal{Q}}(R_k)$ if $\left|\frac{\phi(R_k|\mathcal{F}^k)}{|R_k|} - \tilde{\mathcal{Q}}(R_k)\right| \geq \frac{1}{2m}$, and $\mathcal{Q}(R_k) = \phi(R_k|\mathcal{F}^k)$ otherwise. In other words, we fully trust the forest estimate as long as it leads to the same bin as the one indicated by the user. This distinction is only made if the label is ambiguous, i.e. different than 0 and 1. Otherwise, it corresponds in fact to an exact labeling and is treated as such.

4.5.2 Evaluation

The experiments involving online gradient descent presented in Sec. 4.3.3 and Sec. 4.4.3 were repeated using these discretized inputs instead of the exact ones, for every level of quantization $m \in \{1, \ldots, 5\}$. In terms of retrieval performance, discretizing the inputs does not show any clear difference in comparison to the use of exact user inputs, and this from $m = 1$ on (Fig. 4.8a). For $m = 1$, the paired Wilcoxon's signed-rank test leads to a p-value of 0.65. Additionally, if we assume the differences to be normally distributed, the confidence interval for the mean difference is $[-4.8 \times 10^{-3}, 2.3 \times 10^{-3}]$.



(a) (b)

Figure 4.8: **Evaluation of the effects of input discretization.** (a) Impact of input quantization on the exploration phase. We consider the difference ΔA of area under the curve observed when using discretized user inputs instead of exact ones in the OGD adaptation. The parameter m encodes the fineness of the quantization. No statistical difference is observed from $m = 1$ on. To put the variations in perspective, we also report the difference obtained between the approach without updates and the OGD approach with exact inputs. (b) Whole-slide quantification from discrete inputs. From $m \geq 3$, the discretized approach reaches similar quantification performances as an OGD update with exact inputs.

Hence, by making available to the user 3 buttons (corresponding to the choice $m = 1$) stating respectively whether a region is empty, full of hematopoietic cells or partially covered, the exploration phase is of equivalent quality as the one provided by the online gradient descent method with accurate user labelings.

Since the task of whole-slide quantification from the exploration phase (Sec. 4.4) relies strongly on the user inputs $\tilde{Q}(R_k)$ (see Eq. 4.39), obtaining satisfactory results for this task with discretized inputs requires a more accurate quantization. This minimum level was experimentally found to be $m = 3$ (Fig. 4.8b), which remains nevertheless tractable in practice.

4.6 Conclusion

We introduced an interactive framework able to help a pathologist to navigate efficiently through large digital slides. Our approach is based on a pixelwise random forest classifier pre-trained to segment objects of interest within the tissue and whose predictions are used to score, rank and display regions according to their expected interest. By allowing the user to provide labels on each suggested region, the leaf nodes of the forest model are adjusted in real time during the exploration procedure so that visual characteristics of the data at hand can be gradually incorporated into the region selection process. For this purpose, in addition to two standard leaf update techniques, we introduced a novel adaptation scheme based on online gradient descent which supports one-click inputs from the pathologist instead of more tedious accurate object delineations. Experimental validation was conducted on the task of extramedullary hematopoiesis quantification within mouse liver slides. Beyond its slide exploration abilities, we demonstrated how our method can successfully exploit both the forest segmentation output and the labels collected during the exploration stage to provide accurate estimates of the surface covered by hematopietic cells in the whole slide.

Chapter 5

Image Segmentation as a Twenty Questions Game

We concluded the previous chapter with a discussion about the fact that a user might prefer or be forced to use weaker kinds of inputs requiring only one click instead of more accurate labelings. In this chapter, we investigate further this idea by considering the novel scenario of an interactive binary segmentation task where the user interaction with the machine is restricted to binary inputs (*Yes* and *No*) instead of the usual scribbles or bounding boxes. Potential applications of this scenario include the hands-free segmentation of medical images during surgery, where (i) the process of zooming and navigating through slices can be overwhelming and time-consuming, (ii) the hands of the clinicians are already busy with the operation itself, and (iii) physical interactions with objects must be avoided to keep a sterile environment around the patient. The aforementioned scenario under binary inputs can, in fact, be seen as a Twenty Questions game between the human user and the computer. Before the game starts, the user (also called oracle) thinks about an object within the given image which we will call the answer, and the computer (also called the questioner) is allowed to ask a series of binary questions to guess what the answer is (Fig. 5.1). Beyond the novelty of the scenario, our contributions can be summarized as follows:

- In Sec. 5.2, we introduce a strategy for the computer to provide a guess as accurate as possible in a limited number of questions. At each iteration, our approach builds on Markov Chain Monte Carlo (MCMC) sampling to approximate a probability distribution over the set of possible segmentations, before asking the question halving this set according to a divide-and-conquer approach.
- In Sec. 5.3, we show how a semantic prior on the nature of the object obtained from the output of a random forest classifier can be added in the MCMC framework.
- We evaluate both strategies on the Stanford Background Dataset and compare them with several baselines, demonstrating the effectiveness of the MCMC sampling and the advantage of introducing a prior probability on the object.

The content of this chapter consists of a revisited version of our published conference paper [Rupprecht et al., 2015] and includes an unpublished extension of this work (Sec. 5.3).

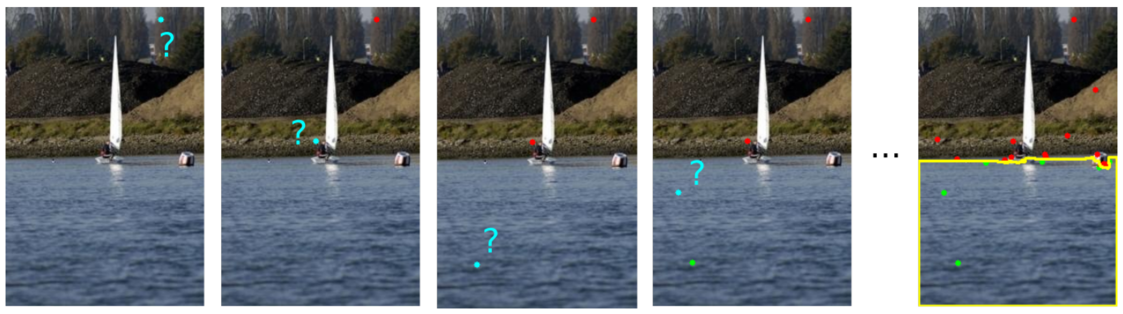


Figure 5.1: **Image segmentation in Twenty Questions.** Given an image, an object to segment (here, the sea) is secretly chosen by the human user. At every step, the computer asks whether a certain pixel is located inside the desired region. Each answer from the user provides either a positive (green) or a negative (red) seed. After a predefined number of questions, the computer returns its guess about the answer based on the collected seeds.

5.1 Related Work

Before exposing our contributions, we briefly review in this section the connections of our work with existing methods. The Twenty Questions setting has already been mentioned in the computer vision community through the work of Branson et al. [2010] and its extension by Wah et al. [2011] for interactive fine-grained image classification. These works consider the case where neither the human user nor the machine knows the label of the image of interest, but (i) the machine knows which questions are important to ask to find out the answer and (ii) the human is able to answer these questions which are based on the visual aspect of the scene. Hence, combining the expertise of the computer with the visual abilities of the human allows to find collaboratively the hidden image label. Beyond the differences in terms of task (image classification *vs* segmentation), this setting is fundamentally different from ours, where the object to be segmented is perfectly known and defined by the human user and has to be guessed by the computer.

Interactive segmentation techniques usually rely on seeds or bounding boxes that are manually placed by a human user in or around the object of interest (Sec. 3.2). Closer to our work, a few approaches [Batra et al., 2010, Fathi et al., 2011, Straehle et al., 2012, Mahapatra et al., 2013] keep the human user in the loop and suggest the most informative areas to label next, in an active learning fashion. An important aspect of our scenario is the intrinsic ambiguity of the image parsing task, as one cannot anticipate the semantic level of the segmentation picked by the oracle. In this direction, Tu and Zhu [2002] introduced a data-driven MCMC framework based on active contours able to generate several parsings of a same image. Recent alternatives identify a set of candidate relevant objects in a scene [Carreira and Sminchisescu, 2012, Endres and Hoiem, 2014, Gu et al., 2009, Krähenbühl and Koltun, 2014] by learning plausible object appearances or shapes. In our case, no offline training is performed so that the method can be adjusted to any kind of image or ground truth. Finally, if combined with e.g. a voice recognition system or a pedal, our method can provide a hands-free segmentation technique, for which a solution using an eye tracker for seed placement was proposed [Sadeghi et al., 2009].

5.2 Object Segmentation in Twenty Questions

In this section, we start by a formal presentation of the Twenty Questions segmentation scenario and an overview of our approach in Sec. 5.2.1 before offering a more detailed treatment of each part of our methodology in Sec. 5.2.2, Sec. 5.2.3 and Sec. 5.2.4.

5.2.1 Problem Statement

Following the general definitions on semantic segmentation introduced in Sec. 1.1.1, we consider here a binary label space $\mathcal{Y} = \{\texttt{Background}, \texttt{Foreground}\}$ where `Foreground` corresponds to the label of interest to the user. Given an image I defined over a two-dimensional domain Ω_I, the user decides on a labeling $\hat{L} : \Omega_I \rightarrow \mathcal{Y}$ that has to be found by the computer. To do so, the computer is going to ask a series of binary questions of the form $Q(\mathbf{p})$: "*Is the displayed location* **p** *inside the object you want to segment?*" where **p** is a location chosen within the image domain Ω_I. Choosing the best question to ask amounts to finding the most informative location **p**. In return, the answer to each question directly provides the true label $\hat{L}(\mathbf{p}) \in \mathcal{Y}$ at this location. After k questions have been posed, the collected answers provide two reliable sets Σ_-^k and Σ_+^k of background and foreground seeds respectively, with $\left|\Sigma_-^k\right| + \left|\Sigma_+^k\right| = k$. We also denote $\Sigma^k = \Sigma_-^k \cup \Sigma_+^k$ the set of reliable seeds collected. This knowledge is encoded through a Bayesian posterior probability $P(L = \hat{L}|\Sigma^k)$ over the set of segmentations $\mathbb{S} = \mathcal{Y}^{\Omega_I}$ stating how likely it is that the segmentation L has been initially picked by the user given the known seeds revealed by the answers already collected. We will denote this probability $P(L|\Sigma^k)$ in the rest of this chapter to make clear that this probability is seen as a function of L and as a probability distribution over $\mathbb{S}$. If the posterior $P(L|\Sigma^k)$ could be computed for every possible segmentation L in $\mathbb{S}$, an optimal divide-and-conquer strategy would halve at each turn the set of possible segmentations into two subsets of probability 0.5 each. However, the number of candidate labelings is extremely large, with theoretically $2^{|\Omega_I|-k}$ possibilities, which excludes any exhaustive computation of $P(L|\Sigma^k)$ over the entire set $\mathbb{S}$. To overcome this, we propose to approximate at each iteration k the posterior $P(L|\Sigma^k)$ by a series of samples $L_1^k, \dots, L_N^k \in \mathbb{S}$ drawn from the distribution $P(.|\Sigma^k)$ according to a Markov Chain Monte Carlo (MCMC) scheme (Sec. 5.2.2). After these N samples have been drawn, we select the most informative question based on these samples only (Sec. 5.2.3). These two sampling and question selection steps are then iterated until a predefined amount of allowed questions is exceeded, and a last sampling stage is then performed to generate the final segmentation output (Sec. 5.2.4).

5.2.2 Sampling Likely Segmentations with MCMC

In this section, we introduce our procedure for sampling representative segmentations from the posterior probability distribution $P(.|\Sigma^k)$ conditioned on the current knowledge provided by the user. Following the Metropolis-Hastings algorithm [Metropolis et al., 1953] and an original idea from Tu and Zhu [2002] introduced in the context of image parsing into multiple regions, our sampling procedure is defined as a Markov chain over the set of segmentations whose transition probabilities are defined as follows. Given a current segmentation L, a new segmentation L' candidate to be the new state of the Markov chain is suggested according to a *proposal distribution* $Q(.|\Sigma^k, L)$ defined over $\mathbb{S}$. This move is accepted with probability $\min(1, \alpha)$ with

$$\alpha = \frac{P(L'|\Sigma^k, I)Q(L|\Sigma^k, L')}{P(L|\Sigma^k, I)Q(L'|\Sigma^k, L)} \tag{5.1}$$

where $P(.|\Sigma^k, I)$ denotes the *posterior probability distribution* according to which we want to draw samples, i.e. the probability for a segmentation to have been picked by the user given the currently known seeds Σ^k and, of course, the image I. We make the dependency in I explicit in $P(L'|\Sigma^k, I)$ to emphasize that this probability depends on the image content over the whole domain[1]. Starting from an initial segmentation L_0^k, a succession of segmentations is generated and the $L_i^k, 1 \leq i \leq N$ are selected as samples at a fixed rate during this exploration process. A longer burn-in step where no samples are retained is performed before starting the sampling procedure to mitigate the dependency on the initial state L_0^k. Our design of posterior and proposal distributions build on a parametrization of candidate segmentations which we are now going to introduce.

State-Space Parametrization

To facilitate the design of efficient proposal distributions in Eq. 5.1, we need a way to perform moves in the set of segmentations $\mathbb{S}$. In their data-driven MCMC framework for image parsing, Tu and Zhu [2002] proposed to deform active contours. In our case, since the MCMC paradigm takes place between two questions in the context of a human/machine interaction, it is essential to keep the time between two questions as small as possible. For this reason, we introduce an alternative view of our segmentation space based on geodesic distance transforms (GDT) that recently demonstrated great efficiency for the suggestion of object proposals [Krähenbühl and Koltun, 2014].

Instead of reasoning in terms of segmentations L and L' in Eq. 5.1, we consider a state space $\mathbb{X} = \{1, \ldots, n_{\text{channels}}\} \times \mathcal{P}(\Omega_I)^2$ so that a state $\mathbf{x} = (c, \Sigma^+, \Sigma^-)$ is defined as an image color channel c and two sets of positive and negative seeds Σ^+ and Σ^-. Σ^+ and Σ^- contain both the already known and hence reliable seeds included in Σ_+^k and Σ_-^k (*fixed* seeds), and a fixed number n_{mobile} of other seeds created for the MCMC process exclusively (*mobile* seeds). We add artificial color channels to the image I by stacking blurred versions of the image, so that n_{channels} denotes the number of total color channels in this enriched version of the image. The correspondence between the state space $\mathbb{X}$ and the set of segmentations $\mathbb{S}$ is done via a labeling function $l_I^{\text{GDT}} : \mathbb{X} \to \mathbb{S}$ that associates to a state $\mathbf{x} = (c, \Sigma^+, \Sigma^-) \in \mathbb{X}$ the segmentation $l_I^{\text{GDT}}(\mathbf{x}) \in \mathbb{S}$ obtained by computing the geodesic distance transform (GDT) on the c^{th} image channel with the sets of positive and negative seeds Σ^+ and Σ^- respectively. The geodesic distance transform of an image is obtained by computing the shortest distance of every pixel to the set of seed pixels.

[1]In fact, most of the mathematical objects introduced here implicity depend on the input image I through the dimensions of its domain and the number of color channels. We consider these quantities fixed and omit these dependencies to simplify the notations.

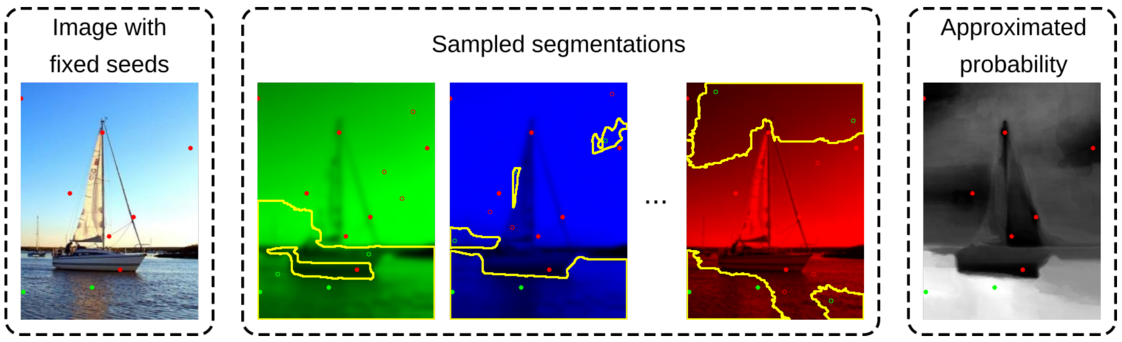


Figure 5.2: **Sampling-based approximation of the segmentation probability**. Given an image and a set of fixed seeds (full disks) provided by previous answers from the user, we sample a set of likely segmentations parametrized by a color channel and a set of mobile seeds (hollow disks). By aggregating these samples, a probability of the label of each pixel given the fixed seeds can be obtained and used to select the next question to be asked.

Usually, the distance between two neighboring pixels (i.e. the edge weights on the image graph) is defined as a mixture of the Euclidean distance and the gradient between these two points. To reduce the dependence on the seed placement within the object of interest, we almost only use the absolute intensity difference to which we simply add a small value ϵ to break ties (for instance if two seeds of different labels are placed in a uniform area). To generate the final GDT-based segmentation, each pixel receives the label of its closest seed. Note that any other seed-based interactive segmentation algorithm could be included instead at this stage of the framework. Our main motivation behind the choice of geodesic distance transforms is the fact that they can be approximated in linear time [Toivanen, 1996] and are hence very fast to compute.

The Markov chain used for our segmentation sampling is going to act at the level of the state space, i.e. on the color channel and seeds used as input for the GDT-based segmentation algorithm. Thus we propose to rewrite Eq. 5.1 as

$$\alpha = \frac{P(\mathbf{x}'|\Sigma^k, I)Q(\mathbf{x}|\Sigma^k, \mathbf{x}')}{P(\mathbf{x}|\Sigma^k, I)Q(\mathbf{x}'|\Sigma^k, \mathbf{x})}. \tag{5.2}$$

It is important to note that we do not have a one-to-one correspondence between Eq. 5.1 and Eq. 5.2. In Eq. 5.1, the probability $P(L|\Sigma^k, I)$ of a segmentation only depends on the answers collected so far, on the image I and on how L partitions the image domain Ω_I. Here, we have *a priori* a different probability for each state $\mathbf{x}$, and each state is related to one color channel only. A same partition of the domain Ω_I can thus have different probabilities depending on the considered color channel. Figure 5.2 illustrates examples of candidate segmentations sampled with our MCMC approach, as well as the aggregated probability for each pixel based on which the next question is selected.

Posterior Probability

The probability $P(\mathbf{x}|\Sigma^k, I)$ represents the probability of a state $\mathbf{x}$ to lead, after application of the GDT-based segmentation, to the segmentation picked by the user given the set Σ^k of k seeds already revealed by the answers to the k first questions. An important characteristic of the Metropolis-Hastings acceptance probability (Eq. 5.2) is the fact that the target probability distribution $P(.|\Sigma^k, I)$ only appears as the ratio between the probability $P(\mathbf{x}'|\Sigma^k, I)$ of the candidate state and the probability $P(\mathbf{x}|\Sigma^k, I)$ of the current state. Hence, multiplying P by a constant leaves the sampling process unchanged. The distribution $P(.|\Sigma^k, I)$ can thus be defined without taking into consideration a theoretically required normalization factor constraining the distribution to sum to 1. To define this probability for a state $\mathbf{x} = (c, \Sigma^+, \Sigma^-)$, we compute the segmentation $l_I^{\mathrm{GDT}}(\mathbf{x})$ and consider the discrepancy between the color content of the background and foreground areas defined by this labeling within the color channel c. More precisely, we create and normalize the corresponding background and foreground histograms $\mathbf{h}^+ = (h_1^+, \ldots, h_B^+)$ and $\mathbf{h}^- = (h_1^-, \ldots, h_B^-)$ and compute their χ^2 distance, i.e.

$$P(\mathbf{x}|\Sigma^k, I) \propto \epsilon + \chi(\mathbf{h}^+, \mathbf{h}^-), \tag{5.3}$$

where

$$\chi(\mathbf{h}^+, \mathbf{h}^-) = \sum_{b=1}^{B} \frac{(h_b^+ - h_b^-)^2}{(h_b^+ + h_b^-)}. \tag{5.4}$$

We introduce a small quantity $\epsilon > 0$ to avoid the existence of zero probabilities. If we do not take this precaution, a uniform image would for example result in a χ^2 distance (in Eq. 5.3) equal to 0 for all states, hence leaving the probability distribution undefined.

Proposal Distribution

Sampled segmentations are generated via a set of parameters $\mathbf{x} = (c, \Sigma^+, \Sigma^-) \in \mathbb{X}$ sent as input to the GDT segmentation algorithm. The main advantage of this representation is that the state space $\mathbb{X}$ gives a more natural way to move from a state to another and facilitates the design of a proposal distribution $Q(.|\Sigma^k, \mathbf{x})$. As discussed when presenting the state space parametrization, Σ^+ and Σ^- contain fixed seeds furnished by the collected answers and mobile seeds used for the sampling procedure only. From a given state $\mathbf{x}$, a state $\mathbf{x}'$ is suggested by drawing uniformly and performing one of the 2 following moves:

1. *Changing image channel*: The image channel c is redrawn uniformly.
2. *Moving a mobile seed*: The position of a mobile seed is uniformly redrawn over the image domain.

$Q(\mathbf{x}'|\Sigma^k, \mathbf{x})$ thus denotes the probability that this procedure creates a state $\mathbf{x}'$ starting from a state $\mathbf{x}$. Note that we introduced a similar Markov chain mechanism in Sec. 2.4.2 for improving node split optimization during the training of a random forest. Since the proposal distribution Q is, here, symmetric by construction, i.e. $Q(\mathbf{x}'|\Sigma^k, \mathbf{x}) = Q(\mathbf{x}|\Sigma^k, \mathbf{x}')$ for all $(\mathbf{x},\mathbf{x}') \in \mathbb{X}^2$, the acceptance probability α used in the Metropolis-Hastings algorithm (Eq. 5.2) can in fact be simplified as

$$\alpha = \frac{P(\mathbf{x}'|\Sigma^k, I)}{P(\mathbf{x}|\Sigma^k, I)}. \tag{5.5}$$

The case of a non-symmetric proposal distribution will be encountered in Sec. 5.3.

5.2.3 Question Selection

After k questions have been asked and answered ($k \geq 0$), the method described in Sec. 5.2.2 draws N segmentations $L_i^k, 1 \leq i \leq N$ that approximate the probability distribution $P(.|\Sigma^k)$, where $P(L|\Sigma^k) = P(L = \hat{L}|\Sigma^k)$ denotes the probability that the segmentation L is the answer $\hat{L}$ awaited by the oracle given the k answer seeds Σ^k. From these samples, we have to decide on the optimal question to ask to the user. Following a divide-and-conquer approach, we ask for the question halving the set of segmentations, i.e we query the label at the location $\mathbf{p}_k$ defined as

$$\mathbf{p}_k = \underset{\mathbf{p}\in\Omega_I}{\operatorname{argmin}} \left| P\left(L(\mathbf{p}) = \texttt{Foreground}|\Sigma^k\right) - \frac{1}{2} \right|. \tag{5.6}$$

Ties are broken at random. Note that, due to the binary aspect of the segmentation, `Foreground` was arbitrarily chosen in Eq. 5.6 and could be equivalently replaced by `Background`. To estimate the probability $P\left(L(\mathbf{p}) = \texttt{Foreground}|\Sigma^k\right)$ in Eq. 5.6, we can use the approximation of an expectation by sampling, i.e. the following identity valid for any label $y \in \mathcal{Y}$:

$$P\left(L(\mathbf{p}) = y|\Sigma^k\right) = \sum_{L\in\mathbb{S}} [L(\mathbf{p}) = y]\, P(L|\Sigma^k) \approx \frac{1}{N}\sum_{i=1}^{N} \left[L_i^k(\mathbf{p}) = y\right]. \tag{5.7}$$

For readability, we denoted $[L(\mathbf{p}) = y]$ the result of the indicator function $\mathbf{1}_{\{y\}}$ evaluated at $L(\mathbf{p})$, which thus outputs 1 if $L(\mathbf{p}) = y$ and 0 otherwise. The approximation in Eq. 5.7 results from the fact that the samples $L_i^k, 1 \leq i \leq N$ are drawn according to the distribution $P(.|\Sigma^k)$.

5.2.4 Creation of the Final Segmentation

At the end of the interactive process, i.e. when the predefined number of questions K is exceeded, we create the final segmentation by taking, for each pixel, the likeliest label given the answers collected, i.e. we output the labeling L^{final} defined for all $\mathbf{p} \in \Omega_I$ as

$$L^{\text{final}}(\mathbf{p}) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\, P\left(\hat{L}(\mathbf{p}) = y|\Sigma^K\right). \tag{5.8}$$

This optimization is in practice conducted by first running a last sampling procedure using the set of all answer seeds Σ_k. The probabilities $P\left(\hat{L}(\mathbf{p}) = y|\Sigma^K\right)$ are then approximated as exposed in Eq. 5.7. A qualitative overview of the entire questioning/answering process is shown in Fig. 5.3.

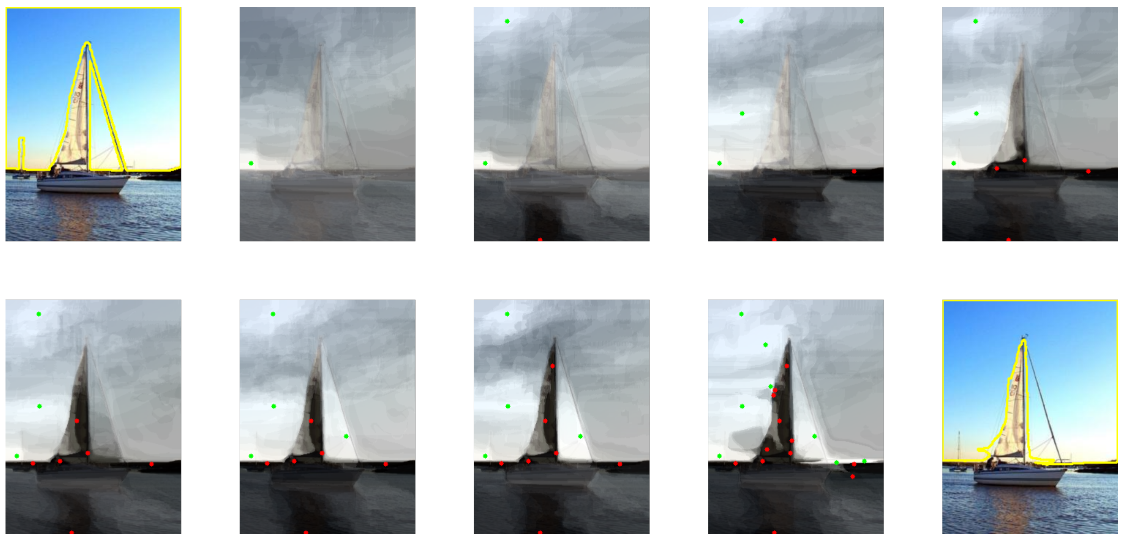


Figure 5.3: **Qualitative evolution of the segmentation belief on an example image**. Given an input image and the label `Sky` as chosen object from the user (top left), we display snapshots of the questioning/answering process and the final guess (bottom right). The pixelwise probabilities conditioned by the current fixed seeds are overlaid. At each iteration, the most uncertain location is chosen as the next question. As the number of answered questions increases, the uncertainty about the object of interest is reduced.

5.3 Introducing Semantic Prior Knowledge

In the method described in Sec. 5.2, there is no prior assumption on the nature of the object chosen by the user: we only expect the object and the background areas to have different color histograms (Eq. 5.3). In this section, we now assume that we have an external source of information (such as a trained classifier) which provides a pixelwise probability $P^{\text{ext}}(Y = y|\mathbf{p}, I)$ for every label $y \in \mathcal{Y}$ and location $\mathbf{p} \in \Omega_I$. We propose to use this additional probability to guide the choice of questions and, at the same time, we keep the flexibility of an interactive segmentation approach so that the Twenty Questions scenario can now be seen as an interactive correction of the segmentation given by P^{ext}.

Defining formally a seed as a pair $\sigma = (\sigma_{\mathbf{p}}, \sigma_y) \in \Omega_I \times \mathcal{Y}$, we use our external source of information to define an image-dependent seed prior $\pi^{\text{ext}}(\sigma|I) = P^{\text{ext}}(Y = \sigma_y|\sigma_{\mathbf{p}}, I)$ stating the compatibility of the seed σ with the prior knowledge P^{ext}. By extension, every state $\mathbf{x} = (c, \Sigma^+, \Sigma^-) \in \mathbb{X}$ can be assigned a state prior

$$\pi^{\text{ext}}(\mathbf{x}|I) = \prod_{\sigma \in \Sigma^+ \cup \Sigma^-} \pi^{\text{ext}}(\sigma|I). \tag{5.9}$$

The state prior defined by Eq. 5.9 favors states whose configuration of seeds matches the available external source of information. To include the prior in the sampling, we could use the same framework as in Sec. 5.2.2 and simply multiply the target distribution by the prior, so that Eq. 5.5 would be rewritten as

$$\alpha = \frac{P(\mathbf{x}'|\Sigma^k, I)\pi^{\text{ext}}(\mathbf{x}'|I)}{P(\mathbf{x}|\Sigma^k, I)\pi^{\text{ext}}(\mathbf{x}|I)}. \tag{5.10}$$

In other words, we sample according to the adjusted target probability $P(.|\Sigma^k, I)\pi^{\text{ext}}(.|I)$ instead of $P(.|\Sigma^k, I)$. Although technically correct, the fact that the proposed Markov chain moves (Sec. 5.2.2) are independent of the prior may result in a lot of rejected moves, and therefore would require a higher number of iterations. In fact, we can achieve an equivalent sampling more efficiently by introducing the state prior directly in the proposal distribution Q: when performing the move *Moving a mobile seed* (see Sec. 5.2.2), we now sample the new location of the seed proportionally to its seed prior so that the suggested moves are directly compatible with the prior. Since this new proposal distribution is no longer symmetric, the acceptance ratio of Eq. 5.10 becomes, following Eq. 5.2 with our new target distribution:

$$\alpha = \frac{P(\mathbf{x}'|\Sigma^k, I)\pi^{\text{ext}}(\mathbf{x}'|I)}{P(\mathbf{x}|\Sigma^k, I)\pi^{\text{ext}}(\mathbf{x}|I)} \frac{Q(\mathbf{x}|\Sigma^k, \mathbf{x}')}{Q(\mathbf{x}'|\Sigma^k, \mathbf{x})} = \frac{P(\mathbf{x}'|\Sigma^k, I)}{P(\mathbf{x}|\Sigma^k, I)}, \tag{5.11}$$

where we used the fact that, for the newly introduced prior-driven proposal distribution Q, we have

$$\frac{Q(\mathbf{x}|\Sigma^k, \mathbf{x}')}{Q(\mathbf{x}'|\Sigma^k, \mathbf{x})} = \frac{\pi^{\text{ext}}(\mathbf{x}|I)}{\pi^{\text{ext}}(\mathbf{x}'|I)}. \tag{5.12}$$

The acceptance ratio in Eq. 5.11 appears to be identical to the original one obtained in Eq. 5.5. In other words, starting from the interactive segmentation method introduced in Sec. 5.2.2, a pixelwise prior knowledge can be naturally incorporated by only modifying the proposal distribution Q, where seeds are sampled according to this prior when we suggest to change their location.

5.4 Experiments

We conducted the evaluation on the Stanford Background Dataset [Gould et al., 2009] for semantic image segmentation which we already encountered in Sec. 2.4.3. We downsized the images by a factor 2. For each image, we alternatively assumed that the object of interest was one of the 8 semantic labels provided as the ground truth and simulated the user responses accordingly. We measure the segmentation quality with the Dice score (Sec. 1.1.2). Each image was enriched by adding blurred versions as additional color channels: more precisely, we created 6 blurred versions of the image, corresponding respectively to Gaussian filtering of standard deviation $1, 2, \ldots, 6$ pixels. We ran 8 independent Markov chains in parallel. Each Markov chain had a burn-in period of 200 iterations and a sample was retained every 100 iterations to reduce the correlation between consecutive samples. The Markov chain moves between samples were run on a version of the image which was further downsized by a factor 3. The approximate geodesic transform was computed using 3 forward and backward passes. $B = 64$ bins were used in the histograms to compute the target probability (Eq. 5.4). 3 mobile seeds for each label were used, for a total of $n_{\text{mobile}} = 6$ mobile seeds. With these settings, we were able to sample a total of $N = 40$ segmentations in less than one second between two questions, which remains practical for an interactive scenario. We compared the four following strategies:

- `Random`: In this baseline, questions are asked randomly at locations which are uniformly drawn over the image domain. The final segmentation is generated by computing the geodesic distance transform from the obtained answers.
- `Uncertainty`: This baseline starts like the `Random` one. However, as soon as one positive and one negative seeds have been found, the next question is asked at the most uncertain location based on the geodesic distance transform. After k collected answers, we thus ask

$$\mathbf{p}_k = \operatorname*{argmin}_{\mathbf{p}\in\Omega_I} \left| d(\mathbf{p}, \Sigma_+^k) - d(\mathbf{p}, \Sigma_-^k) \right|, \tag{5.13}$$

  where $d(\mathbf{p}, \Sigma)$ is the geodesic distance to the closest seed in Σ.
- `MCMC without prior`: Our method described in Sec. 5.2.
- `MCMC with prior`: Our approach with the use of a label prior, as exposed in Sec. 5.3. We used as prior the posterior probability obtained as output of our scale-adaptive forest evaluated in Sec. 2.4.3 with the scale parameter $\sigma = 50$.

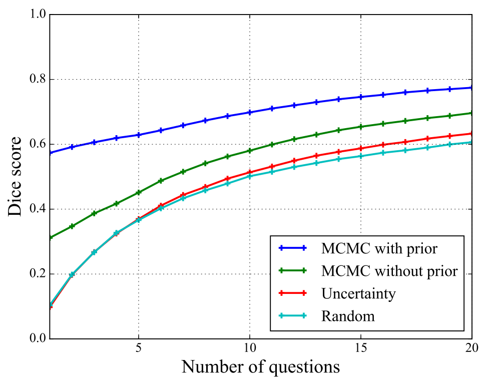
We report in Fig. 5.4a the evolution of the Dice score of the segmentation guess when the number of question increases. Quantitative results (mean and median) for each semantic label and over all segmentation runs are shown in Fig. 5.4c, where we also reported the performance of the (automatic) segmentation built on the forest probability used as semantic prior (`Prior alone`). We observe that our strategy outperforms the two baselines `Random` and `Uncertainty` and that introducing a prior improves the performance. In the latter case, the resulting Dice scores are also higher than using the prior alone thus confirming the benefit of the user interactions. A qualitative example of results with and without prior is shown in Fig. 5.5.

5.5 Possible Extensions

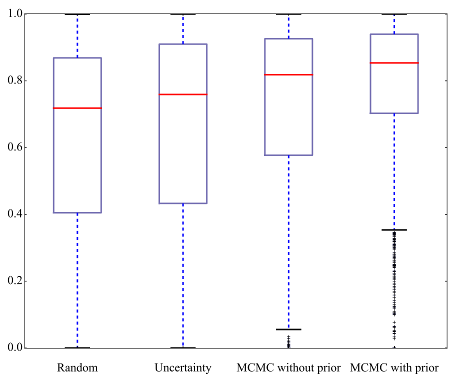
In this final chapter, we introduced the novel scenario of hands-free interactive segmentation with binary inputs and proposed a sampling-based approach following the traditional halving strategy used in the Twenty Questions game and outperforming intuitive baselines. The likelihood of a segmentation was either based on standard homogeneity assumptions or on a learned random forest model able to encode more complex appearances and, thereby, able to improve the quality of the segmentation if the nature of the object of interest is known in advance. We hope that this novel scenario, beyond its theoretical appeal, can also bring interest for alternative possibilities of human-machine interaction in practical situations. Several extensions from this work can be investigated.

Due to the nature of the questions asked by the computer, our approach has, by design, difficulties to find objects in a cluttered scene. If an oracle might possibly have chosen the answer within a large number of non-overlapping objects[2], finding a foreground seed

[2]We can simply consider the synthetic example of an image with a black background and N identical white disks spread all over the image, where N is much greater than 20.



(a) Evolution of the mean Dice score



(b) Distribution of final Dice scores

	Sky	Tree	Road	Grass	Water	Building	Mountain	Foreground	Total
`Random`	65.7	51.4	80.2	56.3	74.6	66.0	39.9	44.8	60.6
`Uncertainty`	69.8	53.2	82.9	59.7	77.0	66.1	44.8	48.5	63.3
`MCMC without prior`	77.3	60.9	85.9	65.1	80.9	74.0	54.2	54.4	69.6
`Prior alone`	82.9	54.3	80.6	55.3	41.0	63.3	4.0	53.1	63.6
`MCMC with prior`	89.2	70.3	90.0	73.6	85.4	78.7	60.6	63.4	77.5

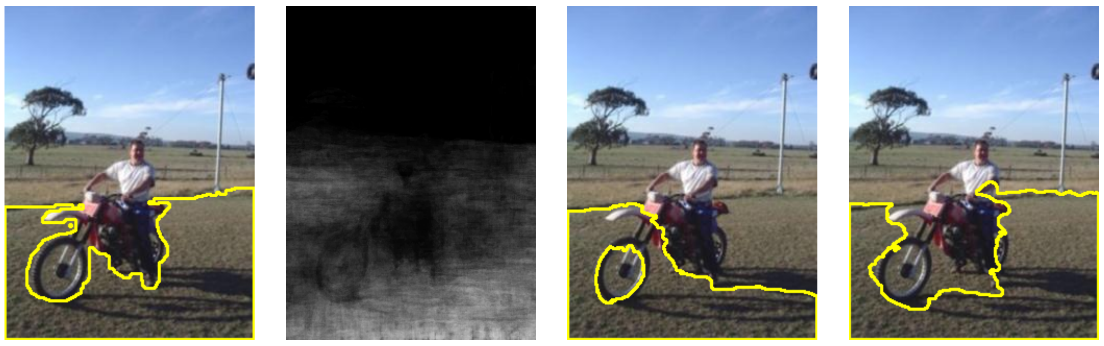
(c) Mean Dice score

Method	Sky	Tree	Road	Grass	Water	Building	Mountain	Foreground	Total
`Random`	82.0	58.6	87.4	72.6	84.7	75.1	35.7	47.9	71.8
`Uncertainty`	89.9	62.3	92.0	78.9	90.5	76.4	55.2	55.3	75.9
`MCMC without prior`	92.7	70.7	92.7	83.4	92.8	82.4	70.4	62.3	81.8
`Prior alone`	90.8	63.7	90.1	69.2	23.8	75.0	0.0	56.9	74.4
`MCMC with prior`	94.6	77.1	94.0	87.9	93.2	84.7	73.3	68.8	85.4

(d) Median Dice score

Figure 5.4: **Experimental results.** (a) Evolution of the mean Dice score depending on the number of questions that have been answered. (b) Distribution over images of the Dice scores obtained after 20 questions. (c) and (d) Mean and median Dice score after 20 questions for each semantic label and in total. The Dice scores of segmentations generated from the prior alone are also reported.

within the 20 questions becomes crucial and mostly a matter of luck. In this case, a more efficient solution would be to show a highlighted area to the user and ask "*Is the target segmentation fully inside the shown area?*", i.e. conducting a spatial search instead of the appearance-based one. If the answer is *Yes*, all the not highlighted pixels could be added to the background seeds and the search space would be greatly reduced. However, in general, a *No* provides less information since it would only state that at least one (unknown) pixel of the object is outside the shown area. Hence, with our current state space, this kind of questions suffers from the opposite problem: they are efficient to find small objects, for which obtaining a *Yes* is likely, but poor at finding large ones. Beyond binary segmentation between background and foreground, the prior knowledge about the object of interest could be queried by asking "*Is the object of semantic label c?*", and an interactive multi-class semantic segmentation could also be considered by asking a question of



(a) Input image (b) Forest prior (c) Without prior (d) With prior

Figure 5.5: **Example of results with and without semantic prior knowledge.** (a) Input image, where the object of interest corresponds to the semantic label `Road`. (b) Pixelwise probability distribution predicted by a random forest for the label `Road`. (c) Prediction after 20 questions for the MCMC method without semantic prior knowledge (Sec. 5.2). (d) Prediction after 20 questions for the MCMC method where the semantic prior knowledge about the label `Road` was introduced via the forest prediction (Sec. 5.3).

the form "*Is the location* $\mathbf{p}$ *part of an object of semantic label* c*?*". With our seed-based state space, all these types of questions have the drawback that a negative answer seems difficult to leverage to constrain efficiently the future sampling iterations. The symmetric aspect of the answers is mainly what motivated our location-based question type, but we believe that our strategy could benefit from a combination of several types of questions and could possibly be extended to other scenarios if the parametrization of the space of segmentations is accordingly adapted.

The interactive segmentation of 3D scans no longer allows the use of geodesic distance transforms in the sampling step due to the large size of the volumes. We suggested in this context an alternative parametrization of the space of segmentations using both a voxelwise prior probability and a prior on the shape of the object [Dubost et al., 2016]. Preliminary results were demonstrated for the segmentation of the prostate. This approach also includes an adaptable term taking into account the fact that the prior probability may not be reliable, and we believe that further works in this direction, together with appropriate visualization techniques to display informative 2D views of the scan, would be relevant for the application of our scenario to a clinical setting.

Conclusion

In this thesis, we proposed several contributions for image segmentation. In the first two chapters, we exposed the concept of supervised learning and its application to the semantic labeling of images, i.e the joint task of parsing an image into regions and identifying their individual content. Among the existing learning techniques, we demonstrated how the framework of decision forests can effectively segment natural scenes and medical images. In this context, we introduced a novel technique which allows a forest equipped with Haar features to be trained in a scale-adaptive fashion. This easy-to-implement variant of the standard forest training does not add any computational cost at both training and prediction stages and does not require any additional hyperparameter, and yet consistently demonstrated improvements on computer vision and medical datasets. Given its simplicity and the popularity of the combination of Haar features with decision forests in the medical field, we hope that this alternative will be tried on other applications and that it will result in a possibly small but in any case costless step towards more accurate segmentation predictions.

Although a fully automated system showing consistent human-level performance is naturally desirable in most applications, reaching this level of accuracy and reliability is a very challenging task in practice. In fact, we believe that many scenarios do not necessarily require a fully automatic system and may instead benefit from hybrid approaches combining learned models with human interactions. Keeping a human user in the loop offers, for example, clear advantages in terms of flexibility and guarantees at the same time a reassuring visual check of the outcome. We considered two situations where standard interactive segmentation techniques do not apply and where a learned forest model is used as a support for the interactions. In the field of digital pathology, the large size of the acquired slides renders the localization of the objects of interest difficult for a clinician so that a large amount of time is spent in the search for the objects themselves in addition to their delineation. We introduced a method building on an learned segmentation model which iteratively selects and displays regions of the slide likely to contain the objects of interest. Beyond the facilitated exploration of the slide, we introduced an adaptive component allowing the pathologist to report the actual revelance of the suggested regions to update the underlying forest model. Several levels of interaction were discussed, including one-click inputs which were experimentally shown to guarantee an exploration stage of the same quality as with accurate delineations. Further in the direction of minimal user interactions, we finally introduced a scenario where interactive segmentation is seen as a Twenty Questions game. In this scenario, the user only communicates with the machine via binary '*Yes*/*No*' answers, with potential applications to hands-free segmentation. We showed that, in addition to its use as a generic interactive segmentation tool, our approach can also be successfully used to correct the output of a learning-based segmentation method.

With the work presented here, we hope to have proposed a generic contribution to the fully-automated segmentation scenario with our adaptive sampling of Haar features in the training of random forest, but also to have emphasized the practical interest behind keeping a user in the loop in the application of automatic methods. Future directions of work that we believe to be promising include the natural limitations of supervised learning which prompt the development of active learning methods to label data more efficiently, of transfer learning techniques to handle slight variations in terms of domain or task, or of strategies to combine labels from several annotators (e.g. in crowdsourcing situations). Beyond these challenges which were already discussed in Sec. 3.1, a key aspect that we have not mentioned yet is the quality of the learned probabilities. Usually, or at least in the case of segmentation, learning techniques are often evaluated based on the final pixelwise label decision only. However, accurate estimates of the uncertainty of the label of each pixel can be of great interest, for instance in quantification settings like the one encountered in histopathology (Chapter 4). Improving the calibration of probabilities [Niculescu-Mizil and Caruana, 2005] and the field of metalearning [Lemke et al., 2015] are possible directions towards better estimates of the reliability of a learned model. Finally, image segmentation is often only an intermediate step to estimate another quantity which is relevant for the considered application, such as the size of an object. This final objective (and, therefore, the measure used to evaluate the accuracy of the entire pipeline) could be introduced earlier in the learning process. First insights on this aspect were provided in our discussion on class balancing (Sec. 2.1.7). For example, attempts at a direct optimization of the Jaccard index in a segmentation context were proposed [Nowozin, 2014, Ahmed et al., 2015]. If a visual check is no longer necessary, segmentation-free methods could even be considered to predict directly the final output using regression [Wang et al., 2014, Zhen et al., 2015].

Appendix A

Integral Images and Volumes

We expose here in details how the quick computation of Haar-like features can be implemented [Viola and Jones, 2004]. We here consider the case of a 2D image defined over the regular lattice $\Omega_I = \{1, \dots, w_I\} \times \{1, \dots, h_I\}$. Consider the notations defined in Fig. A.1, where u_1, u_2, v_1, v_2 are integers so that $1 \leq u_1 < u_2 \leq w_I$ and $1 \leq v_1 < v_2 \leq h_I$ and where the objective is to compute the average of intensities defined by these coordinates, i.e. the quantity

$$\frac{1}{(u_2 - u_1 + 1)\,(v_2 - v_1 + 1)} \sum_{\substack{u_1 \leq u \leq u_2 \\ v_1 \leq v \leq v_2}} I(u, v). \tag{A.1}$$

The critical part of this computation is the sum over the gray rectangle. To make this computation easily tractable for any choice of u_1, u_2, v_1 and v_2, we first build the integral image $\tilde{I}$ of I as the image of dimensions $(w_I + 1) \times (h_I + 1)$ defined as:

$$\begin{aligned} \forall u \in \{1, \dots, w_I + 1\}\ \tilde{I}(u, 1) = 0 \\ \forall v \in \{1, \dots, h_I + 1\}\ \tilde{I}(1, v) = 0 \\ \forall (u, v) \in \{2, \dots, w_I + 1\} \times \{2, \dots, h_I + 1\}\ \tilde{I}(u, v) = \sum_{\substack{1 \leq u' < u \\ 1 \leq v' < v}} I(u', v'). \end{aligned}$$

In practice, $\tilde{I}$ can be built in one pass over the image I by initializing to 0 the two known edges and using the following identity:

$$\tilde{I}(u, v) = I(u - 1, v - 1) + \tilde{I}(u - 1, v) + \tilde{I}(u, v - 1) - \tilde{I}(u - 1, v - 1). \tag{A.2}$$

Once the integral image $\tilde{I}$ is precomputed, the sum appearing in Eq. A.1 can be computed in exactly 4 accesses to the integral image via the identity

$$\sum_{\substack{u_1 \leq u \leq u_2 \\ v_1 \leq v \leq v_2}} I(u, v) = \tilde{I}(u_2 + 1, v_2 + 1) - \tilde{I}(u_2 + 1, v_1) - \tilde{I}(u_1, v_2 + 1) + \tilde{I}(u_1, v_1). \tag{A.3}$$

In the 3D case, i.e. where the region of interest becomes a cube defined by 6 coordinates u_1, u_2, v_1, v_2, z_1 and z_2, the process is similar. Two equations are not entirely straightforward to rewrite: the computation of the integral volume in one pass over

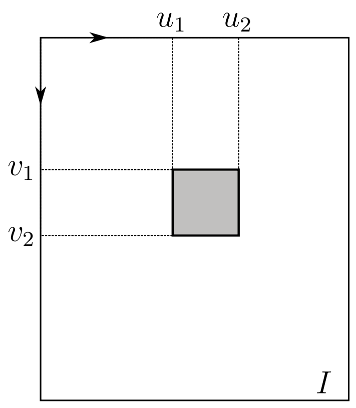


Figure A.1: By computing the integral image $\tilde{I}$ of an image I, the computation of the sum (and mean) of intensities within any rectangular region of I can be efficiently conducted.

the volume (Eq. A.2) and the computation of the sum by accessing the integral volume (Eq. A.3). The first is obtained similarly by initializing to 0 the three known edges of the volume (i.e. the planes defined by $u = 1$, $v = 1$ and $z = 1$ respectively) and using the following recursive relationship:

$$\begin{aligned}\tilde{I}(u,v,z) = {} & \tilde{I}(u-1,v-1,z-1) \\ & - \tilde{I}(u,v-1,z-1) - \tilde{I}(u-1,v,z-1) - \tilde{I}(u-1,v-1,z) \\ & + \tilde{I}(u-1,v,z) + \tilde{I}(u,v-1,z) + \tilde{I}(u,v,z-1) \\ & + I(u-1,v-1,z-1).\end{aligned} \tag{A.4}$$

Once the integral volume is precomputed, the 3D equivalent of Eq. A.3 is

$$\begin{aligned}\sum_{\substack{u_1 \le u \le u_2 \\ v_1 \le v \le v_2 \\ z_1 \le z \le z_2}} I(u,v,z) = {} & \tilde{I}(u_2+1,v_2+1,z_2+1) \\ & - \tilde{I}(u_1,v_2+1,z_2+1) - \tilde{I}(u_2+1,v_1,z_2+1) - \tilde{I}(u_2+1,v_2+1,z_1) \\ & + \tilde{I}(u_2+1,v_1,z_1) + \tilde{I}(u_1,v_2+1,z_1) + \tilde{I}(u_1,v_1,z_2+1) \\ & - \tilde{I}(u_1,v_1,z_1).\end{aligned} \tag{A.5}$$

More details on these results, including generalizations to higher dimensions, can be found in Tapia [2011].

List of Publications

Peer-Reviewed Journal Articles:

- Benjamin Gutierrez-Becker, Diana Mateus, **Loïc Peter**, Nassir Navab. Guiding Multimodal Registration with Learned Optimization Updates. *Medical Image Analysis*, 2017

- **Loïc Peter**, Diana Mateus, Pierre Chatelain, Denis Declara, Noemi Schworm, Stefan Stangl, Gabriele Multhoff, Nassir Navab. Assisting the Examination of Large Histopathological Slides with Adaptive Forests. *Medical Image Analysis*, 2017

- Marie Bieth, **Loïc Peter**, Stephan Nekolla, Matthias Eiber, Georg Langs, Markus Schwaiger, Bjoern Menze. Segmentation of Skeleton and Organs in Whole-Body CT Images via Iterative Trilateration. *IEEE Transactions on Medical Imaging*, 2017

- Sailesh Conjeti, Amin Katouzian, Abhijit Guha Roy, **Loïc Peter**, Debdoot Sheet, Stéphane Carlier, Andrew Laine, Nassir Navab. Supervised Domain Adaptation of Decision Forests: Transfer of Models Trained In Vitro for In Vivo Intravascular Ultrasound Tissue Characterization. *Medical Image Analysis*, 2016

Peer-Reviewed Conference Papers:

- Benjamin Gutierrez-Becker, **Loïc Peter**, Tassilo Klein, Christian Wachinger. A Multi-Armed Bandit to Smartly Select a Training Set from Big Medical Data. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2017

- Benjamin Gutierrez-Becker, Diana Mateus, **Loïc Peter**, Nassir Navab. Learning Optimization Updates for Multimodal Registration. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2016

- **Loïc Peter**, Olivier Pauly, Pierre Chatelain, Diana Mateus, Nassir Navab. Scale-Adaptive Forest Training via an Efficient Feature Sampling Scheme. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015

- Christian Rupprecht, **Loïc Peter**, Nassir Navab. Image Segmentation in Twenty Questions. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015

- **Loïc Peter**, Diana Mateus, Pierre Chatelain, Noemi Schworm, Stefan Stangl, Gabriele Multhoff, Nassir Navab. Leveraging Random Forests for Interactive Exploration of Large Histological Images. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2014 (Oral Presentation)

- Ralf Stauder, Asli Okur, **Loïc Peter**, Armin Schneider, Michael Kranzfelder, Hubertus Feußner, Nassir Navab. Random Forests for Phase Detection in Surgical Workflow Analysis. *International Conference on Information Processing in Computer Assisted Interventions (IPCAI)*, 2014

- Pierre Chatelain, Olivier Pauly, **Loïc Peter**, Seyed-Ahmad Ahmadi, Annika Plate, Kai Bötzel, Nassir Navab. Learning from Multiple Experts with Random Forests: Application to the Segmentation of the Midbrain in 3D Ultrasound. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2013

- **Loïc Peter**, Nicolas Brieu, Sjoert Jansen, Peter Smethurst, Willem Ouwehand, Nassir Navab. Automatic Segmentation and Tracking of Thrombus Formation within In Vitro Microscopic Video Sequences. *IEEE International Symposium on Biomedical Imaging (ISBI)*, 2012

Peer-Reviewed Workshop Papers:

- Florian Dubost, **Loïc Peter**, Christian Rupprecht, Benjamin Gutierrez-Becker, Nassir Navab. Hands-Free Segmentation of Medical Volumes via Binary Inputs. *MICCAI Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, 2016

- **Loïc Peter**, Olivier Pauly, Sjoert Jansen, Peter Smethurst, Willem Ouwehand, Nassir Navab. Automatic Event Detection within Thrombus Formation Based on Binary Integer Programming. *MICCAI Workshop on Medical Computer Vision*, 2012

Bibliography

Seyed-Ahmad Ahmadi, Maximilian Baust, Athanasios Karamalis, Annika Plate, Kai Boetzel, Tassilo Klein, and Nassir Navab. Midbrain segmentation in transcranial 3D ultrasound for parkinson diagnosis. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2011.

Faruk Ahmed, Dany Tarlow, and Dhruv Batra. Optimizing expected intersection-over-union with candidate-constrained CRFs. In *International Conference on Computer Vision*, 2015.

Shaimaa Al-Janabi, André Huisman, and Paul J Van Diest. Digital pathology: current status and future perspectives. *Histopathology*, 61(1):1–9, 2012.

Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, October 1997.

Claus Bahlmann, Amar Patel, Jeffrey Johnson, Jie Ni, Andrei Chekkoury, Parmeshwar Khurd, Ali Kamen, Leo Grady, Elizabeth Krupinski, Anna Graham, et al. Automated detection of diagnostically relevant regions in H&E stained digital pathology slides. In *SPIE Medical Imaging*, 2012.

Panagiotis Balermpas, Franz Rödel, Claus Rödel, Mechthild Krause, Annett Linge, Fabian Lohaus, Michael Baumann, Inge Tinhofer, Volker Budach, Eleni Gkika, et al. CD8+ tumour-infiltrating lymphocytes in relation to HPV status and clinical outcome in patients with head and neck cancer after postoperative chemoradiotherapy: A multi-centre study of the german cancer consortium radiation oncology group (DKTK-ROG). *International Journal of Cancer*, 138(1):171–181, 2016.

Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. iCoseg: Interactive co-segmentation with intelligent scribble guidance. In *Conference on Computer Vision and Pattern Recognition*, 2010.

Thomas W Bauer, Lynn Schoenfield, Renee J Slaw, Lisa Yerian, Zhiyuan Sun, and Walter H Henricks. Validation of whole slide imaging for primary diagnosis in surgical pathology. *Archives of Pathology & Laboratory Medicine*, 137(4):518–524, 2013.

Pinky A Bautista and Yukako Yagi. Staining correction in digital pathology by utilizing a dye amount table. *Journal of Digital Imaging*, pages 1–12, 2015.

Kristin P Bennett and Jennifer A Blue. A support vector machine approach to decision trees. In *International Joint Conference on Neural Networks*, 1998.

Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

Yuri Y Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *International Conference on Computer Vision*, 2001.

Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, 2010.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine Learning*, 2001.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. Chapman and Hall/CRC, 1984.

Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008a.

Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European Conference on Computer Vision*, 2008b.

Anne E Carpenter, Thouis R Jones, Michael R Lamprecht, Colin Clarke, In H Kang, Ola Friman, David A Guertin, Joo H Chang, Robert A Lindquist, Jason Moffat, et al. Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biology*, 7(10):R100, 2006.

João Carreira and Cristian Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

Pierre Chatelain, Olivier Pauly, Loïc Peter, Seyed-Ahmad Ahmadi, Annika Plate, Kai Bötzel, and Nassir Navab. Learning from multiple experts with random forests: Application to the segmentation of the midbrain in 3D ultrasound. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2013.

Sailesh Conjeti, Amin Katouzian, Abhijit Guha Roy, Loïc Peter, Debdoot Sheet, Stéphane Carlier, Andrew Laine, and Nassir Navab. Supervised domain adaptation of decision forests: Transfer of models trained in vitro for in vivo intravascular ultrasound tissue characterization. *Medical Image Analysis*, 32:1–17, 2016.

Lee AD Cooper, Alexis B Carter, Alton B Farris, Fusheng Wang, Jun Kong, David A Gutman, Patrick Widener, Tony C Pan, Sharath R Cholleti, Ashish Sharma, et al. Digital pathology: Data-intensive frontier in medical imaging. *Proceedings of the IEEE*, 100 (4):991–1003, 2012.

Antonio Criminisi. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 2011.

Antonio Criminisi and Jamie Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.

Antonio Criminisi, Toby Sharp, and Andrew Blake. Geos: Geodesic image segmentation. In *European Conference on Computer Vision*, 2008.

Antonio Criminisi, Jamie Shotton, and Stefano Bucciarelli. Decision forests with long-range spatial context for organ localization in CT volumes. In *MICCAI Workshop on Probabilistic Models for Medical Image Analysis*, 2009.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, 2005.

Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

Scott Doyle, Michael Feldman, John Tomaszewski, and Anant Madabhushi. A boosted bayesian multiresolution classifier for prostate cancer detection from digitized needle biopsies. *IEEE Transactions on Biomedical Engineering*, 59(5):1205–1218, 2012.

Florian Dubost, Loïc Peter, Christian Rupprecht, Benjamin Gutierrez Becker, and Nassir Navab. Hands-free segmentation of medical volumes via binary inputs. In *MICCAI Workshops on Deep Learning and Data Labeling for Medical Applications*, pages 259–268, 2016.

Thomas Ebner, Darko Štern, Rene Donner, Horst Bischof, and Martin Urschler. Towards automatic bone age estimation from mri: Localization of 3d anatomical landmarks. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2014.

Daniël Eefting, Yvonne M Schrage, Maartje JA Geirnaerdt, Saskia Le Cessie, Anthonie HM Taminiau, Judith VMG Bovée, Pancras CW Hogendoorn, et al. Assessment of interobserver variability and histologic parameters to improve reliability in classification and grading of central cartilaginous tumors. *The American journal of surgical pathology*, 33(1):50–57, 2009.

Ian Endres and Derek Hoiem. Category-independent object proposals with diverse ranking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):222–234, 2014.

Navid Farahani, Anil Parwani, and Liron Pantanowitz. Whole slide imaging in pathology: advantages, limitations, and emerging perspectives. *Pathology and Laboratory Medicine International*, 7:23–33, 2015.

Alireza Fathi, Maria Florina Balcan, Xiaofeng Ren, and James M Rehg. Combining self training and active learning for video segmentation. In *British Machine Vision Conference*, 2011.

Itzhak Fogel and Dov Sagi. Gabor filters as texture discriminator. *Biological cybernetics*, 61(2):103–113, 1989.

Alejandro F Frangi, Wiro J Niessen, Koen L Vincken, and Max A Viergever. Multiscale vessel enhancement filtering. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 1998.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, 1997.

Yaozong Gao, Yeqin Shao, Jun Lian, Andrew Z Wang, Ronald C Chen, and Dinggang Shen. Accurate segmentation of CT male pelvic organs via regression-based deformable models and multi-task random forests. *IEEE Transactions on Medical Imaging*, 35(6):1532–1543, 2016.

Romane Gauriau, Rémi Cuingnet, David Lesage, and Isabelle Bloch. Multi-organ localization with cascaded global-to-local regression and shape prior. *Medical Image Analysis*, 23(1):70–83, 2015.

Ezequiel Geremia, Olivier Clatz, Bjoern H Menze, Ender Konukoglu, Antonio Criminisi, and Nicholas Ayache. Spatial decision forests for MS lesion segmentation in multichannel magnetic resonance images. *NeuroImage*, 57(2):378–390, 2011.

Ezequiel Geremia, Bjoern H Menze, and Nicholas Ayache. Spatially adaptive random forests. In *International Symposium on Biomedical Imaging*, 2013.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

Floyd H Gilles, C Jane Tavaré, Laurence E Becker, Peter C Burger, Allan J Yates, Ian F Pollack, and Jonathan L Finlay. Pathologist interobserver variability of histologic features in childhood brain tumors: results from the CCG-945 study. *Pediatric and Developmental Pathology*, 11(2):108–117, 2007.

Ben Glocker, Olivier Pauly, Ender Konukoglu, and Antonio Criminisi. Joint classification-regression forests for spatially structured multi-object segmentation. In *European Conference on Computer Vision*, 2012.

Josep M. Gonfaus, Xavier Boix, Joost Van De Weijer, Andrew D. Bagdanov Joan Serrat, and Jordi Gonzàlez. Harmony potentials for joint classification and segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2010.

Ipek Isik Gonul, Aylar Poyraz, Cigdem Unsal, Cenk Acar, and Turgut Alkibay. Comparison of 1998 WHO/ISUP and 1973 WHO classifications for interobserver variability in grading of papillary urothelial neoplasms of the bladder. pathological evaluation of 258 cases. *Urologia internationalis*, 78(4):338–344, 2006.

Lena Gorelick, Olga Veksler, Mena Gaed, José A Gomez, Madeleine Moussa, Glenn Bauman, Aaron Fenster, and Aaron D Ward. Prostate histopathology: Learning tissue component histograms for cancer detection and classification. *IEEE Transactions on Medical Imaging*, 32(10):1804–1818, 2013.

Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *International Conference on Computer Vision*, 2009.

Norberto A Goussies, Sebastián Ubalde, and Marta Mejail. Transfer learning decision forests for gesture recognition. *Journal of Machine Learning Research*, 15(1):3667–3690, 2014.

Leo Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

Leo Grady, Marie-Pierre Jolly, and Aaron Seitz. Segmentation from a box. In *International Conference on Computer Vision*, 2011.

Chunhui Gu, Joseph J Lim, Pablo Arbelaez, and Jitendra Malik. Recognition using regions. In *Conference on Computer Vision and Pattern Recognition*, 2009.

Danna Gurari, Suyog Dutt Jain, Kristen Grauman, and Margrit Betke. Pull the plug? Predicting if computers or humans should segment images. In *International Conference on Computer Vision*, 2015.

Metin N Gurcan, Laura Boucheron, Ali Can, Anant Madabhushi, Nasir Rajpoot, and Bulent Yener. Histopathological image analysis: A review. *IEEE Reviews in Biomedical Engineering*, 2:147–171, 2009.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 2016.

Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Conference on Computer Vision and Pattern Recognition*, 2004.

David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. In *International Joint Conference on Artificial Intelligence*, 1993.

Mattias P. Heinrich and Maximilian Blendowski. Multi-organ segmentation using vantage point forests and binary context features. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2016.

Michael Held, Michael HA Schmitz, Bernd Fischer, Thomas Walter, Beate Neumann, Michael H Olma, Matthias Peter, Jan Ellenberg, and Daniel W Gerlich. CellCognition: time-resolved phenotype annotation in high-throughput live cell imaging. *Nature Methods*, 7(9):747–754, 2010.

Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

André Homeyer, Andrea Schenk, Uta Dahmen, Olaf Dirsch, Hai Huang, and Horst K Hahn. A comparison of sampling strategies for histological image analysis. *Journal of Pathology Informatics*, 2, 2011.

André Homeyer, Andrea Schenk, Janine Arlt, Uta Dahmen, Olaf Dirsch, and Horst K Hahn. Practical quantification of necrosis in histological whole-slide images. *Computerized Medical Imaging and Graphics*, 37(4):313–322, 2013.

Soheil Hor and Mehdi Moradi. Learning in data-limited multimodal scenarios: Scandent decision forests and tree-based features. *Medical Image Analysis*, 34:30–41, 2016.

Chao-Hui Huang, Antoine Veillard, Ludovic Roux, Nicolas Loménie, and Daniel Racoceanu. Time-efficient sparse analysis of histopathological whole slide images. *Computerized Medical Imaging and Graphics*, 35(7):579–591, 2011.

Juan Eugenio Iglesias, Ender Konukoglu, Albert Montillo, Zhuowen Tu, and Antonio Criminisi. Combining generative and discriminative models for semantic segmentation of CT scans via active learning. In *International Conference on Information Processing in Medical Imaging*, 2011.

Thomas Jaarsma, Halszka Jarodzka, Marius Nap, Jeroen JG Merrienboer, and Henny Boshuizen. Expertise under the microscope: processing histopathological slides. *Medical Education*, 48(3):292–300, 2014.

Vidit Jain and Erik Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *Conference on Computer Vision and Pattern Recognition*, 2011.

Drazen M Jukić, Laura M Drogowski, Jamie Martina, and Anil V Parwani. Clinical examination and validation of primary diagnosis in anatomic pathology using whole slide digital images. *Archives of Pathology & Laboratory Medicine*, 135(3):372–378, 2011.

Adnan Mujahid Khan, Nasir Rajpoot, Darren Treanor, and Derek Magee. A non-linear mapping approach to stain normalisation in digital histopathology images using image-specific colour deconvolution. *IEEE Transactions on Biomedical Engineering*, 2014.

Parmeshwar Khurd, Claus Bahlmann, Peter Maday, Ali Kamen, Summer Gibbs-Strauss, Elizabeth M Genega, and John V Frangioni. Computer-aided gleason grading of prostate cancer histopathological images using texton forests. In *International Symposium on Biomedical Imaging*, 2010.

Stefan Kluckner, Thomas Mauthner, Peter M Roth, and Horst Bischof. Semantic image classification using consistent regions and individual context. In *British Machine Vision Conference*, 2009a.

Stefan Kluckner, Thomas Mauthner, Peter M Roth, and Horst Bischof. Semantic classification in aerial imagery by integrating appearance and height information. In *Asian Conference on Computer Vision*, 2009b.

Hui Kong, Metin Gurcan, and Kamel Belkacem-Boussaid. Partitioning histopathological images: An integrated framework for supervised color-texture segmentation and cell splitting. *IEEE Transactions on Medical Imaging*, 30(9):1661–1677, 2011.

Peter Kontschieder, Pushmeet Kohli, Jamie Shotton, and Antonio Criminisi. GeoF: Geodesic forests for learning coupled predictors. In *Conference on Computer Vision and Pattern Recognition*, 2013.

Peter Kontschieder, Samuel Rota Bulo, Marcello Pelillo, and Horst Bischof. Structured labels in random forests for semantic labelling and object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(10):2104–2116, 2014.

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulò. Deep neural decision forests. In *International Conference on Computer Vision*, 2015.

Ender Konukoglu, Ben Glocker, Darko Zikic, and Antonio Criminisi. Neighbourhood approximation using randomized forests. *Medical Image Analysis*, 17(7):790–804, 2013.

Philipp Krähenbühl and Vladlen Koltun. Geodesic object proposals. In *European Conference on Computer Vision*, 2014.

John D. Lafferty, Andrew McCallum, and Fernando C N Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. Mondrian forests: Efficient online random forests. In *Advances in Neural Information Processing Systems*, 2014.

Nathan Lay, Neil Birkbeck, Jingdan Zhang, and Kevin S Zhou. Rapid multi-organ segmentation using context integration and discriminative models. In *International Conference on Information Processing in Medical Imaging*, 2013.

Christian Leistner, Amir Saffari, Jakob Santner, and Horst Bischof. Semi-supervised random forests. In *International Conference on Computer Vision*, 2009.

Christian Leistner, Amir Saffari, and Horst Bischof. MIForests: Multiple-instance learning with randomized trees. In *European Conference on Computer Vision*, 2010.

Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44(1):117–130, 2015.

Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *International Conference on Computer Vision*, 2009a.

Victor Lempitsky, Michael Verhoek, J Alison Noble, and Andrew Blake. Random forest classification for automatic delineation of myocardium in real-time 3D echocardiography. In *International Conference on Functional Imaging and Modeling of the Heart*, 2009b.

Claudia Lindner, Shankhar Thiagarajah, J Mark Wilkinson, The arcOGEN Consortium, Gillian A Wallis, and Timothy F Cootes. Fully automatic segmentation of the proximal femur using random forest regression voting. *IEEE Transactions on Medical Imaging*, 32(8):1462–1472, 2013.

Xiao Liu, Mingli Song, Dacheng Tao, Zicheng Liu, Luming Zhang, Chun Chen, and Jiajun Bu. Random forest construction with robust semisupervised node splitting. *IEEE Transactions on Image Processing*, 24(1):471–483, 2015.

Herve Lombaert, Darko Zikic, Antonio Criminisi, and Nicholas Ayache. Laplacian forests: semantic image segmentation by guided bagging. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2014.

Herve Lombaert, Antonio Criminisi, and Nicholas Ayache. Spectral forests: Learning of surface data, application to cortical parcellation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2015.

Marc Macenko, Marc Niethammer, James S Marron, David Borland, John T Woosley, Xiaojun Guan, Charles Schmitt, and Nancy E Thomas. A method for normalizing histology slides for quantitative analysis. In *International Symposium on Biomedical Imaging*, 2009.

Dwarikanath Mahapatra, Peter J Schüffler, Jeroen AW Tielbeek, Franciscus M Vos, and Joachim M Buhmann. Semi-supervised and active learning for automatic segmentation of crohn's disease. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2013.

Lena Maier-Hein, Sven Mersmann, Daniel Kondermann, Sebastian Bodenstedt, Alexandro Sanchez, Christian Stock, Hannes Gotz Kenngott, Mathias Eisenmann, and Stefanie Speidel. Can masses of non-experts train highly accurate image classifiers? In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2014.

Yishay Mansour. Pessimistic decision tree pruning based on tree size. In *International Conference on Machine Learning*, 1997.

Ján Margeta, Ezequiel Geremia, Antonio Criminisi, and Nicholas Ayache. Layered spatio-temporal forests for left ventricle segmentation from 4D cardiac MRI data. In *MICCAI Workshop on Statistical Atlases and Computational Models of the Heart*, 2011.

Raphael Meier, Stefan Bauer, Johannes Slotboom, Roland Wiest, and Mauricio Reyes. Patient-specific semi-supervised learning for postoperative brain tumor segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2014.

Bjoern Menze, Michael Kelm, Daniel Splitthoff, Ullrich Koethe, and Fred Hamprecht. On oblique random forests. In *Machine Learning and Knowledge Discovery in Databases*, 2011.

Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015.

Ezgi Mercan, Selim Aksoy, Linda G Shapiro, Donald L Weaver, Tad Brunye, and Joann G Elmore. Localization of diagnostically relevant regions of interest in whole slide images. In *International Conference on Pattern Recognition*, 2014.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

John S Meyer, Consuelo Alvarez, Clara Milikowski, Neal Olson, Irma Russo, Jose Russo, Andrew Glass, Barbara A Zehnbauer, Karen Lister, and Reza Parwaresch. Breast carcinoma malignancy grading by Bloom–Richardson system vs proliferation index: reproducibility of grade and advantages of proliferation index. *Modern Pathology*, 18(8): 1067–1078, 2005.

John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243, 1989.

Albert Montillo, Jamie Shotton, John Winn, Juan Eugenio Iglesias, Dimitri Metaxas, and Antonio Criminisi. Entangled decision forests and their application for semantic segmentation of CT images. In *International Conference on Information Processing in Medical Imaging*, 2011.

Albert Montillo, Jilin Tu, Jamie Shotton, John Winn, Juan Eugenio Iglesias, Dimitri Metaxas, and Antonio Criminisi. Entanglement and differentiable information gain maximization. In Antonio Criminisi and Jamie Shotton, editors, *Decision Forests for Computer Vision and Medical Image Analysis*, pages 273–293. Springer, 2013.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

Kien Nguyen, Anindya Sarkar, and Anil K Jain. Prostate cancer grading: use of graph cut and spatial arrangement of nuclei. *IEEE Transactions on Medical Imaging*, 33(12): 2254, 2014.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *International Conference on Machine learning*, 2005.

Mohammad Norouzi, Maxwell Collins, Matthew A Johnson, David J Fleet, and Pushmeet Kohli. Efficient non-greedy optimization of decision trees. In *Advances in Neural Information Processing Systems*, 2015.

Sebastian Nowozin. Improved information gain estimates for decision tree induction. In *International Conference on Machine Learning*, 2012.

Sebastian Nowozin. Optimal decisions from probabilistic models: the intersection-over-union case. In *Conference on Computer Vision and Pattern Recognition*, 2014.

Sebastian Nowozin and Christoph H Lampert. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 2011.

Sebastian Nowozin, Carsten Rother, Shai Bagon, Toby Sharp, Yao Bangpeng, and Pushmeet Kohli. Decision tree fields. In *International Conference on Computer Vision*, 2011.

Devrim Onder, Selen Zengin, and Sulen Sarioglu. A review on color normalization and color deconvolution methods in histopathology. *Applied Immunohistochemistry & Molecular Morphology*, 22(10):713–719, 2014.

Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit. Fast keypoint recognition in ten lines of code. In *Conference on Computer Vision and Pattern Recognition*, 2007.

Olivier Pauly, Ben Glocker, Antonio Criminisi, Diana Mateus, Axel Martinez-Möller, Stephan Nekolla, and Nassir Navab. Fast multiple organ detection and localization in whole-body MR dixon sequences. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2011.

Olivier Pauly, Seyed-Ahmad Ahmadi, Annika Plate, Kai Bötzel, and Nassir Navab. Detection of substantia nigra echogenicities in 3D transcranial ultrasound for early diagnosis of parkinson disease. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2012.

Nadia Payet and Sinisa Todorovic. Hough forest random field for object recognition and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5): 1066–1079, 2013.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Loïc Peter, Diana Mateus, Pierre Chatelain, Noemi Schworm, Stefan Stangl, Gabriele Multhoff, and Nassir Navab. Leveraging random forests for interactive exploration of large histological images. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2014.

Loïc Peter, Olivier Pauly, Pierre Chatelain, Diana Mateus, and Nassir Navab. Scale-adaptive forest training via an efficient feature sampling scheme. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

Loïc Peter, Diana Mateus, Pierre Chatelain, Denis Declara, Noemi Schworm, Stefan Stangl, Gabriele Multhoff, and Nassir Navab. Assisting the examination of large histopathological slides with adaptive forests. *Medical Image Analysis*, 2016.

Brian L Price, Bryan Morse, and Scott Cohen. Geodesic graph cut for interactive image segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2010.

J Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

J Ross Quinlan. *C4. 5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Andrew Rabinovich, Sameer Agarwal, Casey Laris, Jeffrey H Price, and Serge J Belongie. Unsupervised color decomposition of histologically stained tissue samples. In *Advances in Neural Information Processing Systems*, 2003.

Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge J Belongie. Objects in context. In *International Conference on Computer Vision*, 2007.

Laura Elena Raileanu and Kilian Stoffel. Theoretical comparison between the Gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1): 77–93, 2004.

Daniele Ravì, Miroslaw Bober, Giovanni Maria Farinella, Mirko Guarnera, and Sebastiano Battiato. Semantic segmentation of images exploiting DCT based features and random forest. *Pattern Recognition*, 52:260–273, 2016.

Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. Global refinement of random forest. In *Conference on Computer Vision and Pattern Recognition*, 2015.

David Richmond, Dagmar Kainmueller, Ben Glocker, Carsten Rother, and Gene Myers. Uncertainty-driven forest predictors for vertebra localization and segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

Samuel Rota Bulò and Peter Kontschieder. Neural decision forests for semantic image labelling. In *Conference on Computer Vision and Pattern Recognition*, 2014.

Samuel Rota Bulò and Peter Kontschieder. Online learning with bayesian classification trees. In *Conference on Computer Vision and Pattern Recognition*, 2016.

Holger R Roth, Le Lu, Amal Farag, Hoo-Chang Shin, Jiamin Liu, Evrim B Turkbey, and Ronald M Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

Vincent Roullier, Olivier Lézoray, Vinh-Thong Ta, and Abderrahim Elmoataz. Multi-resolution graph-based analysis of histopathological whole slide images: Application to mitotic cell extraction and visualization. *Computerized Medical Imaging and Graphics*, 35(7):603–615, 2011.

Amelie Royer and Christoph H Lampert. Classifier adaptation at prediction time. In *Conference on Computer Vision and Pattern Recognition*, 2015.

Christian Rupprecht, Loïc Peter, and Nassir Navab. Image segmentation in twenty questions. In *Conference on Computer Vision and Pattern Recognition*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Maryam Sadeghi, Geoffrey Tien, Ghassan Hamarneh, and M Stella Atkins. Hands-free interactive image segmentation using eyegaze. In *SPIE Medical Imaging*, 2009.

Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. In *International Conference on Computer Vision Workshops*, 2009.

Jakob Santner, Markus Unger, Thomas Pock, Christian Leistner, Amir Saffari, and Horst Bischof. Interactive texture segmentation using random forests and total variation. In *British Machine Vision Conference*, 2009.

Jakob Santner, Thomas Pock, and Horst Bischof. Interactive multi-label segmentation. In *Asian Conference on Computer Vision*, 2010.

Matthias Schneider, Sven Hirsch, Bruno Weber, Gábor Székely, and Bjoern H Menze. Joint 3-D vessel segmentation and centerline extraction using oblique hough forests with steerable filters. *Medical Image Analysis*, 19(1):220–249, 2015.

Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Object class segmentation using random forests. In *British Machine Vision Conference*, 2008.

Samuel Schulter, Paul Wohlhart, Christian Leistner, Amir Saffari, Peter M Roth, and Horst Bischof. Alternating decision forests. In *Conference on Computer Vision and Pattern Recognition*, 2013.

Alexander G Schwing, Christopher Zach, Yefeng Zheng, and Marc Pollefeys. Adaptive random forest: How many 'experts' to ask before making a decision? In *Conference on Computer Vision and Pattern Recognition*, 2011.

Olcay Sertel, Jun Kong, Hiroyuki Shimada, Umit V Catalyurek, Joel H Saltz, and Metin N Gurcan. Computer-aided prognosis of neuroblastoma on whole-slide images: Classification of stromal development. *Pattern Recognition*, 42(6):1093–1103, 2009.

Burr Settles. Active learning literature survey, 2010.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundation and Trends® in Machine Learning*, 4(2):107–194, 2012.

Toby Sharp. Implementing decision trees and forests on a GPU. In *European Conference on Computer Vision*, 2008.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, 2006.

Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2008.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 2009.

Jamie Shotton, Andrew W Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Conference on Computer Vision and Pattern Recognition*, 2011.

Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, et al. Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2821–2840, 2013a.

Jamie Shotton, Toby Sharp, Pushmeet Kohli, Sebastian Nowozin, John Winn, and Antonio Criminisi. Decision jungles: Compact and rich models for classification. In *Advances in Neural Information Processing Systems*, 2013b.

Christoph Sommer, Christoph Straehle, Ullrich Köthe, and Fred A Hamprecht. Ilastik: Interactive learning and segmentation toolkit. In *International Symposium on Biomedical Imaging*, 2011.

Ralf Stauder, Aslı Okur, Loïc Peter, Armin Schneider, Michael Kranzfelder, Hubertus Feussner, and Nassir Navab. Random forests for phase detection in surgical workflow analysis. In *International Conference on Information Processing in Computer-Assisted Interventions*, 2014.

Darko Štern, Thomas Ebner, and Martin Urschler. From local to global random regression forests: Exploring anatomical landmark localization. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2016.

Christoph Straehle, Ullrich Köthe, Graham Knott, Kevin Briggman, Winfried Denk, and Fred A Hamprecht. Seeded watershed cut uncertainty estimators for guided interactive segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2012.

Hai Su, Fuyong Xing, Xiangfei Kong, Yuanpu Xie, Shaoting Zhang, and Lin Yang. Robust cell detection and segmentation in histopathological images using sparse reconstruction and stacked denoising autoencoders. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

Kai Tao, Min Fang, Joseph Alroy, and G Gary Sahagian. Imagable 4T1 model for the study of late stage breast cancer. *BMC Cancer*, 8(1), 2008.

Ernesto Tapia. A note on the computation of high-dimensional integral images. *Pattern Recognition Letters*, 32(2):197–201, 2011.

Pekka J Toivanen. New geodosic distance transforms for gray-scale images. *Pattern Recognition Letters*, 17(5):437–450, 1996.

Tatiana Tommasi, Francesco Orabona, Mohsen Kaboli, and Barbara Caputo. Leveraging over prior knowledge for online learning of visual categories. In *British Machine Vision Conference*, 2012.

Andrew Top, Ghassan Hamarneh, and Rafeef Abugharbieh. Active learning for interactive 3D image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2011.

Zhuowen Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *International Conference on Computer Vision*, 2005.

Zhuowen Tu and Xiang Bai. Auto-context and its application to high-level vision tasks and 3D brain image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1744–1757, October 2010.

Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5): 657–673, 2002.

Abhishek Vahadane, Tingying Peng, Amit Sethi, Shadi Albarqouni, Lichao Wang, Maximilian Baust, Katja Steiger, Anna Melissa Schlitter, Irene Esposito, and Nassir Navab. Structure-preserving color normalization and sparse stain separation for histological images. *IEEE Transactions on Medical Imaging*, 35(8):1962–1971, 2016.

Mitko Veta, Josien PW Pluim, Paul J Van Diest, and Max A Viergever. Breast cancer histopathology image analysis: A review. *IEEE Transactions on Biomedical Engineering*, 61(5):1400–1411, 2014.

Mitko Veta, Paul J. van Diest, Stefan M. Willems, Haibo Wang, Anant Madabhushi, Angel Cruz-Roa, Fabio A. González, Anders Boesen Lindbo Larsen, Jacob S. Vestergaard, Anders B. Dahl, et al. Assessment of algorithms for mitosis detection in breast cancer histopathology images. *Medical Image Analysis*, pages 237–248, 2015.

Alexander Vezhnevets and Joachim M Buhmann. Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. In *Conference on Computer Vision and Pattern Recognition*, 2010.

Sudheendra Vijayanarasimhan and Kristen Grauman. Cost-sensitive active visual category learning. *International Journal of Computer Vision*, 91(1):24–44, 2011.

Paul Viola and Michael J Jones. Robust real-time face detection. *International Journal of Computer Vision*, 2004.

Catherine Wah, Steve Branson, Pietro Perona, and Serge Belongie. Multiclass recognition and part localization with humans in the loop. In *International Conference on Computer Vision*, 2011.

Zhijie Wang, Mohamed Ben Salah, Bin Gu, Ali Islam, Aashish Goela, and Shuo Li. Direct estimation of cardiac biventricular volumes with an adapted bayesian formulation. *IEEE Transactions on Biomedical Engineering*, 61(4):1251–1260, 2014.

Simon K Warfield, Kelly H Zou, and William M Wells. Simultaneous truth and performance level estimation (staple): an algorithm for the validation of image segmentation. *IEEE Transactions on Medical Imaging*, 23(7):903–921, 2004.

Yan Xu, Jun-Yan Zhu, I Eric, Chao Chang, Maode Lai, and Zhuowen Tu. Weakly supervised histopathology cancer image segmentation and classification. *Medical Image Analysis*, 18(3):591–604, 2014.

Yiqing Yang, Zhouyuan Li, Li Zhang, Christopher Murphy, Jim Hoeve, and Hongrui Jiang. Local label descriptor for example based semantic image labeling. In *European Conference on Computer Vision*, 2012.

Mohammad Yaqub, M Kassim Javaid, Cyrus Cooper, and J Alison Noble. Investigation of the role of feature selection and weighted voting in random forests for 3-D volumetric segmentation. *IEEE Transactions on Medical Imaging*, 33(2):258–271, 2014.

Yunming Ye, Qingyao Wu, Joshua Zhexue Huang, Michael K Ng, and Xutao Li. Stratified sampling for feature subspace selection in random forests for high dimensional data. *Pattern Recognition*, 2013.

Zhao Yi, Antonio Criminisi, Jamie Shotton, and Andrew Blake. Discriminative, semantic segmentation of brain tissue in MR images. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2009.

Peilin Zhao and Steven C Hoi. OTL: A framework of online transfer learning. In *International Conference on Machine Learning*, 2010.

Xiantong Zhen, Ali Islam, Mousumi Bhaduri, Ian Chan, and Shuo Li. Direct and simultaneous four-chamber volume estimation by multi-output regression. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H S Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision*, 2015.

Xiaolong Zhu, Xuhui Jia, and Kwan-Yee K Wong. Structured forests for pixel-level hand detection and hand part labelling. *Computer Vision and Image Understanding*, 141: 95–107, 2015.

Darko Zikic, Ben Glocker, Ender Konukoglu, Antonio Criminisi, Çağatay Demiralp, Jamie Shotton, Owen M Thomas, Tilak Das, Raj Jena, and Stephen J Price. Decision forests for tissue-specific segmentation of high-grade gliomas in multi-channel MR. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2012.

Darko Zikic, Ben Glocker, and Antonio Criminisi. Encoding atlases by randomized classification forests for efficient multi-atlas label propagation. *Medical Image Analysis*, 18(8):1262–1273, 2014.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, 2003.