Multiscale Centerline Extraction Based on Regression and Projection onto the Set of Elongated Structures

THÈSE N^{O} 7241 (2016)

PRÉSENTÉE LE 2 DÉCEMBRE 2016

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE VISION PAR ORDINATEUR
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Amos SIRONI

acceptée sur proposition du jury:

Dr M. Rajman, président du jury
Prof. P. Fua, Prof. V. Lepetit, directeurs de thèse
Prof. S. Mallat, rapporteur
Dr G. Malandain, rapporteur
Prof. M. Unser, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2016

Abstract

Automatically extracting linear structures from images is a fundamental low-level vision problem with numerous applications in different domains. Centerline detection and radial estimation are the first crucial steps in most Computer Vision pipelines aiming to reconstruct linear structures.

Existing techniques rely either on hand-crafted filters, designed to respond to ideal profiles of the linear structure, or on classification-based approaches, which automatically learn to detect centerline points from data.

Hand-crafted methods are the most accurate when the content of the image fulfills the ideal model they rely on. However, they lose accuracy in the presence of noise or when the linear structures are irregular and deviate from the ideal case.

Machine learning techniques can alleviate this problem. However, they are mainly based on a classification framework. In this thesis, we show that classification is not the best formalism to solve the centerline detection problem. In fact, since the appearance of a centerline point is very similar to the points immediately next to it, the output of a classifier trained to detect centerlines presents low localization accuracy and double responses on the body of the linear structure.

To solve this problem, we propose a regression-based formulation for centerline detection. We rely on the distance transform of the centerlines to automatically learn a function whose local maxima correspond to centerline points. The output of our method can be used to directly estimate the location of the centerline, by a simple Non-Maximum Suppression operation, or it can be used as input to a tracing pipeline to reconstruct the graph of the linear structure. In both cases, our method gives more accurate results than state-of-the-art techniques on challenging 2D and 3D datasets.

Our method relies on features extracted by means of convolutional filters. In order to process large amount of data efficiently, we introduce a general filter bank approximation scheme. In particular, we show that a generic filter bank can be approximated by a linear combination of a smaller set of separable filters. Thanks to this method, we can greatly reduce the computation time of the convolutions, without loss of accuracy. Our approach is general, and we demonstrate its effectiveness by applying it to different Computer Vision problems, such as linear structure detection and image classification with Convolutional Neural Networks.

We further improve our regression-based method for centerline detection by taking advantage of contextual image information. We adopt a multiscale iterative regression approach to efficiently include a large image context in our algorithm. Compared to previous approaches, we use context both in the spatial domain and in the radial one. In this way, our method is also able to return an accurate estimation of the radii of the linear structures. The idea of using regression can also be beneficial for solving other related Computer Vision problems. For example, we show an improvement compared to previous works when applying it to boundary and membrane detection.

Finally, we focus on the particular geometric properties of the linear structures. We observe that most methods for detecting them treat each pixel independently and do not model the strong relation that exists between neighboring pixels. As a consequence, their output is geometrically inconsistent. In this thesis, we address this problem by considering the projection of the score map returned by our regressor onto the set of all geometrically admissible ground truth images. We propose an efficient patch-wise approximation scheme to compute the projection. Moreover, we provide conditions under which the projection is exact. We demonstrate the advantage of our method by applying it to four different problems.

Keywords: Centerline Detection, Linear Structure Reconstruction, Multiscale Detection, Radius Estimation, Distance Transform, Regression, Iterative Regression, Boundary Detection, Separable Convolution, Tensor Decomposition, Membrane Detection, Nearest Neighbor Projection, Junction Detection.

Résumé

Extraire automatiquement des structures linéaires dans des images est un problème fondamental pour de nombreux systèmes de vision, avec des applications dans différents domaines. La détection des lignes médianes et l'estimation du rayon en sont les premières étapes, cruciales pour le succès du reste de l'extraction.

Les techniques existantes reposent soit sur des filtres prédéfinis, conçus pour répondre à un profil idéal de la structure linéaire, ou sur des approches de classification qui apprennent automatiquement à détecter les points de la ligne médiane à partir d'exemples annotés manuellement.

Les méthodes aux critères prédéfinis sont les plus précises quand le contenu de l'image correspond au modèle idéal sur lequel elles reposent. Toutefois, elles perdent leur précision en présence de bruit ou lorsque les structures linéaires sont irrégulières et s'éloignent du cas idéal.

Les techniques d'apprentissage automatique peuvent atténuer ce problème. Cependant, elles sont principalement fondées sur des méthodes de classification. Dans cette thèse, nous montrons que la classification n'est pas la meilleure manière d'affronter le problème de détection des lignes médianes. En effet, comme l'apparence d'un point sur la ligne médiane est très similaire à celle des points immédiatement voisins, le classificateur, exercé à détecter les lignes médianes, ne parvient pas à les localiser précisément et donne souvent des réponses multiples pour une même ligne.

Pour résoudre ce problème, nous proposons une méthode fondée sur la régression pour détecter la ligne médiane. En utilisant la transformée de distances des lignes médianes, notre méthode apprend une fonction dont les maxima locaux correspondent aux points de la ligne médiane. La réponse de notre algorithme peut être utilisée pour estimer directement la position de la ligne médiane, par une simple opération de suppression des non-maxima, ou encore comme point d'entrée dans une pipeline de délinéation pour reconstituer le schéma de la structure linéaire. Dans les deux cas, notre méthode donne de meilleurs résultats que les techniques de l'état de l'art sur des bases de données complexes en 2D et en 3D.

Notre méthode repose sur des descripteurs extraits avec des filtres convolutionnels. Afin de traiter un grand nombre de données efficacement, nous introduisons un procédé général d'approximation d'un ensemble de filtres. En particulier, nous montrons que, quel que soit l'ensemble de filtres, une combinaison linéaire d'un ensemble plus petit de filtres séparables en donne une très bonne approximation. Grâce à cette méthode, on peut réduire considérablement le temps de calcul des convolutions, sans perte de précision. Notre approche est générale et nous montrons son efficacité en l'appliquant à différents problèmes de vision par ordinateur, notamment la détection de structures linéaires mais aussi la classification d'images avec des réseaux de neurones profonds.

Nous améliorons encore notre méthode de détection de la ligne médiane par régression, en exploitant le contexte de l'image. Nous adoptons une approche de régression itérative à plusieurs échelles pour réussir à inclure un grand contexte d'images dans notre algorithme. A la différence des approches précédentes, nous utilisons le contexte à la fois dans le domaine spatial et radial. De cette manière, notre méthode est également capable de fournir une estimation précise des rayons des structures linéaires. Notre idée d'employer la régression peut aussi être utile pour résoudre d'autres problèmes similaires de vision par ordinateur. Par exemple, nous montrons une amélioration par rapport aux travaux précédents quand nous l'appliquons à la détection des contours et des membranes.

Pour finir, nous nous concentrons sur les propriétés géométriques particulières des structures linéaires. Nous observons que la plupart des méthodes de détection traitent chaque pixel indépendamment et ne modélisent pas la forte relation qui existe entre pixels voisins. En conséquence, leur résultat peut être incohérent d'un point de vue géométrique. Dans cette thèse, nous traitons ce problème en prenant en compte la projection de l'image obtenue avec notre régresseur sur l'ensemble de toutes les images-témoin servant à la vérification, qui sont géométriquement admissibles. Nous proposons un procédé d'approximation efficient grâce à une décomposition en mosaïque pour réaliser la projection rapidement. De plus, nous établissons aussi des conditions dans lesquelles la projection est exacte. Nous montrons l'avantage de notre méthode en l'appliquant à quatre problèmes différents.

Mots clefs : Détection de la ligne médiane, Reconstruction de structures linéaires, Détection multiscalaire, Estimation du rayon, Transformée de distances, Régression, Régression itérative, Détection de contours, Convolution séparable, Décomposition de tenseurs, Détection des membranes, Projection du voisin le plus proche, Détection des points de jonction.

Acknowledgements

More than four years have passed since my arrival at EPFL and during all this time I met a lot of extraordinary people. It is thank to them and to all the experiences I lived here that I could grow from a personal and scientific point of view.

First, I would like to thank my supervisor Pascal Fua, who accepted me as part of his lab with great enthusiasm and who always demonstrated great trust in me. A special thank to my other advisor, Vincent Lepetit, for his ability to motivate and encourage me and for all the time and the ideas he put in following my PhD. Without the guidance of my two advisors this thesis would not have been possible.

I would as well thank the members of my thesis jury: Martin Rajman, Grégoire Malandain, Stéphane Mallat and Michaël Unser, for accepting to evaluate my work. In particular, I want to thank Stéphane Mallat for giving me the possibility to work with him and his team in Paris. During my time there, I had the chance to meet very smart and kind people, who made my stay refreshing and pleasant. Thank to Edouard, Irène, Mathieu, Chris, Vincent, Sira, Grégoire, Ivan and Carmine.

I thank as well all the incredible people of CVLab, in particular Jonas, Dat and Agata, with whom I had the privilege to share the office. Many thanks also to Carlos, Engin, Fethallah, Bugra, Przemyslaw, Pablo and Kwang for all the knowledge they transmitted me and their indispensable help. Many thanks to two extraordinary secretaries, Josiane and Ariane, for their kindness and for taking care so well of this peculiar group of engineers and scientists.

Thanks to the Italian group: Roberto, Alberto, Marco and Jean for making these years also an amusing adventure and for being a source of good mood also in the toughest moments. Many thanks to my family and to my adoptive French family for always welcoming me with open arms (and open fridge) to restore my heart with their love (and my stomach with their food). Finally, the greatest thank to Audrey, for being such a wonderful person and for sharing every single moment, the finest and also the roughest, of the last four years.

L'essentiel est invisible pour les yeux.

— Antoine de Saint-Exupéry

A Audrey.

Contents

List of Figures

List of Tables

CHAPTER 1

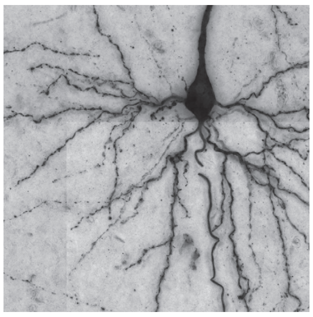
Introduction

For many tasks in scene analysis, there may not exist general solutions independent of purpose or intended application. However, for the task of linear delineation, one can easily find image subsets for which a panel of human observers would be almost unanimous in their interpretation without having to agree on the explicit criteria underlying their decision; our goal is to produce a computer system that can perform the delineation task at close to human levels for at least these more obvious cases, especially where semantic knowledge is not required.

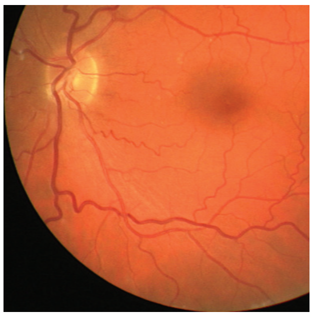
In this way M.A. Fischler and H.C. Wolf begin their 1983 "A General Approach to Machine Perception of Linear Structure in Imaged Data" [65]. 33 years later, these sentences are still relevant and help us to understand the fundamental nature of the linear delineation problem.

Automated linear delineation in images was one of the first problems addressed by Computer Vision scientists [166, 159, 63, 64]. This is in part due to the large number of applications of such a system (Sec. 1.1), but also because of the inherent ability of the human visual system to recognize linear structures without the need of "explicit criteria", and despite the huge variability that such structures show in images (Fig. 1.1)

Its basic nature makes linear delineation a very interesting but also challenging problem and, even though huge progress has been made in the field of Computer Vision and of linear delineation in particular, a "computer system that can perform the delineation task

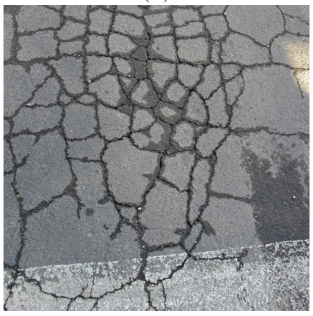


(a)

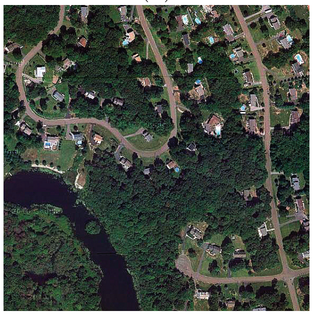


(b)

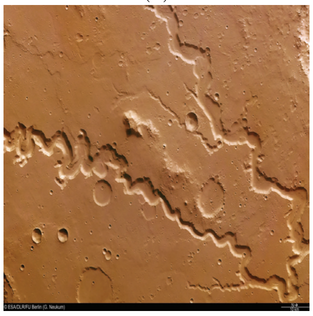
(c)



(d)



(e)



(f)

Figure 1.1: Linear structures are extremely common in natural and man made systems. Their aspect can vary remarkably depending on the nature of the structure or of the acquisition technique. Nevertheless, a human observer can easily detect them, with high consistency among different observers. (a) Minimum intensity projection of a brightfield image stack containing dendrites; (b) Retinal fundus image showing blood vessels (source: DRIVE dataset [185]); (c) RGB photograph of a fungal mycelium (source: flickr.com/photos/bushman_k/6177594429); (d) Cracking in asphalt pavement (source: en.wikipedia.org/wiki/File:Asphalt_deterioration.jpg); (e) Aerial image of a New York neighborhood; (f) Satellite image of the Martial Nanedi Valles (source: ESA, original image at sci.esa.int/mars-express/39161-nanedi-valles-valley-system/). This image, as most of the images in this thesis, is best viewed in color.

at close to human levels" still does not exist. In this thesis we set ourselves the same goals as the authors of [65] and we focus more specifically on the Centerline Detection task.

We present a novel way to look at the centerline detection problem, by reformulating it as a *regression* problem. Our approach is significantly more accurate and general than previous methods for the same task. Moreover, we show that using the centerlines detected with our method as input to complete delineation algorithms increases the final performance.

Before entering into the details of the method, which will be described in the next chapters, we first define our problem and describe the main challenges and some applications of a linear delineation system, in next section. Then, in Sec. 1.2, we list the main contributions of the thesis and outline its content in Sec. 1.3.

1.1 The Centerline Detection Problem: Definition, Applications and Challenges

Linear structures, also referred to as curvilinear structures or "line-like" structures, are extremely common in physical, biological and artificial systems. They can be rivers in satellite images, axons and dendrites in the brain, road networks or cracks in buildings. As a consequence their study is required in many fields, such as neuroscience, biology and cartography.

In addition, in the last decades, the capabilities and the efficiency of image acquisition systems, such as electronic microscopy, high-resolution cameras, etc., have considerably increased, making it possible to collect a very large amount of data for the study of linear structures. However, because no reliable fully-automated system for the extraction of curvilinear structure is available, the exploitation of such data is still limited by the time consuming and tedious task of manual annotation.

In order to build a reliable and fully automated linear delineation system, we start by defining our problem. We again rely on [65] and adopt their definition:

> We define linear delineation as the task of generating a set of lists of points, for a given 2-D image, such that the points in each list fall sequentially along what any reasonable human observer would describe as clearly visible "line-like" structure in the image.

We extend this definition, by considering not only 2D images, but general n-dimensional images, such as 3D biomedical image stacks or 4D spatiotemporal sequences. Moreover, the list of points generated by the computer system should correspond to the center (also called skeleton or ridge) of the linear structure, we will refer to it as the *centerline*. Finally, to each point in the list should be associated the value corresponding to the width (also called radius or scale) of the linear structure in that point. We will refer to as *scale-space* the $(n+1)$-dimensional space, where the first n dimensions are spatial coordinates, indicating physical points in the image, and the last dimension is the radial coordinate, indicating the value of the radius of the linear structure.

From the definition of [65], we see that linear delineation is a low level ability of the visual system: "any reasonable human observer" is used in the definition to intuitively denote a linear structure. This shows how basic the concept of linear structures for a human observer is, and as a consequence, how difficult it is to find a more formal definition. In fact, a general definition of a linear structure should account for all the possible sources of variability present in images. This variability can be due to the real aspect of the linear structure, or to artifacts, such as structured and unstructured noise, occlusions, discontinuities, irregular widths, etc.

For this reason, in this thesis, we will rely on the "reasonable human observer", rather than on a mathematical model, to discriminate linear structures and to evaluate the performance of a computer system. More precisely, we formulate centerline detection as a Machine Learning problem. A set of images containing linear structures will be labeled by human observers and used to train the system. Another set of images, also annotated by human observers, will be used to test the performance of the system by comparing the points it detects as centerlines, against the points labeled by the human observers. We will refer to the labeled images as *manual annotations* or *ground truth*.

The main applications we will consider in this thesis are the segmentation of biomedical data and the detection of roads in aerial images. However, our method is general and can be applied to any other image containing linear structures. Moreover, as we will show, it can also be applied to related problems, provided that training data is available, such as boundary and edge detection. In the following, we discuss in more detail some applications and use them to illustrate relevant challenges associated to the linear structure detection problem.

1.1.1 Neurite Tracing: The Data Size Challenge

Modeling the brain is one of the main challenges of this century [155]. Advances in this field could lead to the discovery of better treatments or of a cure for neurodegenerative diseases. Computer Science algorithms could be inspired by the functioning of biological neural networks to create systems with similar or better performance than animals brain.

More broadly, understanding the principles and the functioning behind the human brain would be a major advance for human knowledge, with important consequences both from the scientific, but also philosophical and ethical point of view.

Because of these, and many other reasons, big efforts have been made by the scientific community to better understand the brain [109, 132, 101, 119, 84, 36, 6, 133, 59]. A big part of the research in this field is dedicated to the study of the morphology and connectivity of neurons. This research relies on the acquisition and annotation of large amount of data from samples, with the intention to extract meaningful statistics from them.

Amazing imaging techniques [47, 142, 124, 87] have been developed to produce better and better quality images of the brain, under different aspects, at different scales and with greater and greater resolution. Considering the number of neuroscientific articles published every year and the amount of data generated per study [199], we can estimate that several terabytes of raw data are generated every year with these techniques.

One of the main obstacles today is represented by the time needed by human experts to accurately annotate the objects contained in the data. This step is necessary in order to extract relevant quantitative information from them.

A linear structure delineation algorithm would be of great help to automatically extract neurons, dendrites and axons from images or image stacks representing sections of the brain. Thus, to relieve neuroscientists from the tedious and time consuming task of manually annotating such data, and allowing them to mainly focus on the analytical and theoretical part of the research.

A delineation system of this kind should be robust enough to deal with the complex shapes that different neurons have and that the different imaging techniques show. Moreover, given the amount of data to be processed, efficiency is a fundamental requirement for a delineation algorithm to be of practical use in this field.

In this thesis, we will develop an accurate and general algorithm for linear structure extraction, by always keeping in mind the importance of fast computation. For example, in Chap. 4, we exploit separable convolution to speed up the extraction of features from n-dimensional images. In Chap. 6, we propose an efficient approximation techniques for the projection of images into the set of elongated structures. We will show that in this way we can obtain a considerable speed up over standard methods, with better accuracy.

1.1.2 Blood Vessel Reconstruction: Localization Accuracy

The reconstruction and analysis of blood vessels is required in many applications, such as computer assisted intervention or in the diagnosis of vascular diseases [203, 134, 217, 99, 4, 7].

For these applications, an accurate localization of the centerline and of the width of the blood vessels is of crucial importance for the correct analysis of the image and to help the medical doctor to take the most appropriate measures for the patient.

Low resolution, artifacts and different sources of noise can compromise the accuracy of standard detection techniques. Also, Machine Learning algorithms based on pixel classification suffer from low localization accuracy because of the similar aspect of a centerline pixel and pixels immediately next to it.

In this thesis, we reformulate centerline detection as a regression problem (Chap. 3). In this way, we enforce the output of our method to have well localized maxima corresponding to the centerline pixels. Thanks to our approach, we obtain results that are more accurate than classification and hand-crafted methods.

1.1.3 Road Detection: The Importance of Context

Another class of images where linear structures often appear is represented by aerial and satellite images. Roads, rivers, fields boundaries are some examples of linear structures present in such images. The automatic detection of these objects can be useful in numerous domains, from autonomous navigation to agriculture and environment monitoring [41, 79, 112, 208, 205]

One of the applications we focus on in this thesis is road centerline detection. The great diversity of aspects of roads and of objects in the background makes it hard to find a general road detector based only on local cues.

Studying the road detection problem, we realize how important it is to consider large portions of the image in order to be able to discriminate roads and non-roads pixels. In fact, many background objects, such as roof tops, have similar local appearance to roads. Moreover, occlusions, due to trees or cars, commonly occur.

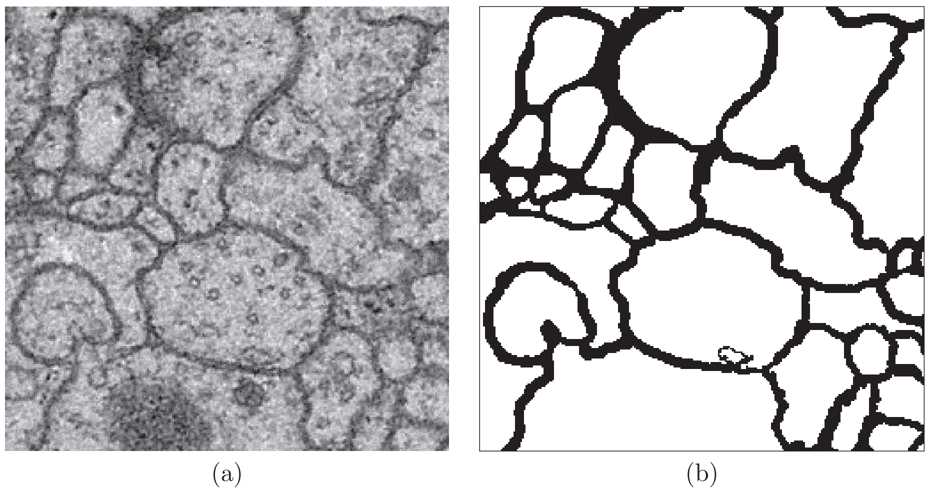


Figure 1.2: Cellular membranes in 2D slices of EM stacks appear as linear structures. Extracting the membranes from the images is a beneficial preprocessing step to correctly segment the different cells in the stack. In this kind of applications, preserving the topology of the membranes is more important than accurate localization or width estimation. (a) Sections from a serial section Transmission Electron Microscopy (ssTEM) dataset of the Drosophila first instar larva ventral nerve cord. (b) Human labeling of image (a), black pixels corresponds to membranes pixels. Source: ISBI dataset [35].

As a consequence, a global approach, or at least a large context, should be considered in order to reconstruct road networks reliably.

In this thesis, we propose a scale-space context-aware method for linear structures detection (Chap. 5). We show the advantage of using this context-based approach to obtain more robust and consistent detections of the centerlines and estimations of their radii.

1.1.4 Membrane Detection: Junctions and Topological Relevant Features

Membranes in 3D Election Microscope (EM) stacks appear as linear structures when considering 2D sections of the stack (Fig. 1.2). Detecting the membranes in these images is one of the fundamental prerequisites for correctly segment the volume in disjoint regions, each corresponding to a separate cell or organelle in the biological sample [38, 126, 12]. Many medical and biological studies rely on such segmentations [97, 98, 187, 33].

In this kind of applications, it is not accurate localization and accurate width estimation that matters the most, but rather the correct reconstruction of the connections of the linear structure or the correct separation of the different regions [95, 12]. In other words, the reconstruction returned by the computer system should have the same topology of the objects of interest, and small localization displacements are tolerated.

In such situation, some characteristic features of linear structure networks, such as junctions, branching points and crossings, become of great importance for the correct interpretation of the connectivity. Although rare events in the image, these topological relevant points are very important parts of line networks. Failing to correctly detect them can change the connectivity of the network and therefore generate a reconstruction that, even if close to the real structure on the pixel level, is completely wrong from the topological point of view.

Model-based methods [67, 113] often fail to robustly detect such points because their appearance deviate from the ideal model of the linear structure they rely on. Machine Learning methods [76, 204] are also less accurate on these points because of the sparsity of these objects in images and therefore their relative rare occurrence in the training set.

In order to accurately detect junctions and obtain more consistent results from the topological point of view, in this thesis, we refine the output score of a linear structure detector by projecting it onto the set of all possible linear structures ground truth images (Chap. 6). We will show that this projection can be approximated by a nearest neighbor projection of image patches, allowing us to efficiently obtain a solution.

1.2 Contributions

In this thesis, we consider the linear structure detection problem and, in order to tackle the challenges described in the previous section, we propose a general algorithm for automatically extracting the centerline and estimating the radius of the linear structures.

Previous approaches rely either on an ideal model of the linear structures or on binary classification techniques. Model-based approaches depend on filters designed to respond to locally cylindrical structures [67, 171, 110, 139, 114, 210], optimized for specific profiles [93]. They are accurate when the structures of interest in the image follow the ideal model they rely on. However, they lose accuracy in presence of noise, occlusions, irregular cross section, non-uniform intensity or other artifacts. In practice, it is difficult to define a model general enough to include all possible sources of variability that linear structures can have in images, without having to specify and tune a large number of parameters.

Machine learning-based approaches, although requiring manually annotated data for the training stage, can overcome the limitation of model-based ones. They mainly rely on a binary classification frameworks [169, 76, 23]. However, as we will show in Chap. 3, they show low localization accuracy when applied to centerline detection. This is because it is hard for the classifier to distinguish points on the centerline itself from those immediately next to it. Moreover, these approaches usually only focus on detecting the centerlines and do not provide an estimation of the radii.

In this thesis, we show that these problems can be solved by reformulating centerline detection and radius estimation in terms of a regression problem. More precisely, we train several regressors to return distances to the closest centerline in scale-space, each regressor being trained for a specific scale. In this way, performing Non-Maximum Suppression on their output yields both centerline locations and corresponding radii. We will show that, on very irregular structures, it outperforms the powerful Optimally Oriented Flux (OOF) approach with and without the anti-symmetry term [113, 114], which is widely acknowledged as one of the best among those relying on hand-designed filters; a very recent extension of it [194] designed to improve its performance on irregular structures; and a similarly recent classification-based method [23].

In order to extract features from the images used as input to the regressor, we consider a convolutional filter learning approach [163]. We show how standard tensor decomposition techniques can be used to learn n-dimensional filter banks that are adapted for the linear detection problem. Moreover, we introduce a general filter bank approximation scheme and show that large filter banks can be approximated accurately by a smaller set of separable filters. Thanks to this approach we can speed up the feature extraction step considerably.

Moreover, we introduce an additional refinement, inspired by the Auto-Context algorithm [193], which was originally proposed for image segmentation. We extend this method to a multiscale framework, considering both spatial and radial contextual information. More precisely, we use the output of the original regressors as features to a layer of new ones. By iterating this process, we can progressively correct earlier mistakes by exploiting contextual information across a widening portion of the image. In particular, this helps eliminate false detections on the background, to fill gaps in the linear structures and to obtain more accurate radii estimation.

Even though this approach is significantly more accurate than previous work, it essentially classifies individual pixels or voxels and does not explicitly model the strong relationship that exists between neighboring pixels. As a result, isolated erroneous responses, discontinuities, and topological errors are still present in the resulting score maps, in particular close to junctions. To overcome these limitations, we show that we can induce global spatial consistency on the regressor score map by systematically replacing pixel neighborhoods by their nearest neighbors in a set of ground truth training patches. This is in the spirit of algorithms for image denoising and inpainting that search for nearest neighbors within the image itself [44, 43, 128]. It is also closely related to structured learning approaches [74, 54]. By contrast, in our method we compute distances in terms of the ground truth and score image patches and we will show that this approach is more accurate than previous work, especially near junctions. Furthermore, assuming that the set of all admissible ground truth images is well represented by the set of training patches, we formally show that our method is equivalent to projecting the score map onto the set of all admissible ground truth maps.

We will also evaluate the ability of our method to trace the linear structures. In particular, we will demonstrate that feeding our output as input to a complete tracing algorithm [196] increases final performance.

Finally, the idea of using regression instead of classification is generic and can be applied to other problems. For example, in contour detection, due to low resolution, blurring, and other image artifacts, the exact boundary location is often hard to find. Training a classifier to separate boundary points from others typically produce multiple responses on the boundaries and poor localization accuracy. We will show that applying our approach to detect boundaries in natural images avoid these problems.

In summary, these are the main contributions of this thesis:

- We reformulate centerline detection as a regression problem. In particular, starting from the distance transform of the centerline, we define a function whose local maxima correspond to centerline pixels. Then, by training a regressor to predict this function from a given image, we can extract centerlines and corresponding radii by a simple Non-Maximum Suppression operation. This method is discussed in Chap. 3.
- To accelerate the convolutional feature extraction step, we develop a general technique to approximating an arbitrary filter bank as a linear combination of a smaller set of separable filters. Our method relies on tensor decomposition techniques and allows us to process large n-dimensional images with no significant significant loss in accuracy (Chap. 4).
- We show and exploit the importance of contextual information for the linear structure detection problem by introducing a scale-space context-aware approach for centerline detection and radial estimation (Chap. 5).
- Finally, we propose an efficient approximation method for projecting score maps onto the set of the ground truth images of elongated structures. We provide conditions for optimality of the projection and, by applying it to linear structure detection, we obtain more geometrically consistent results. This method is presented in Chap. 6.

1.3 Thesis Outline

The rest of the thesis is organized as follows: In Chap. 2, we discuss related work on centerline detection, linear structure segmentation and boundary detection. In particular, we propose a taxonomy for characterizing this class of methods. In Chap. 3, we formalize multiscale centerline detection as a regression problem. We also introduce the datasets and the evaluation metrics used in our experiments. The separable filter approximation framework, used for feature extraction, is described in Chap. 4. In Chap. 5, we describe our context-aware method and apply it in conjunction with a tracing algorithm to show the utility of an accurate centerline detection algorithm for a complete linear structure tracing pipeline. In Chap. 6, we describe how the particular structure of the set of elongated structures can be exploited to approximate image projections efficiently. We give sufficient conditions for the optimality of the projection. We apply this method to improve the results of our regression-based method, in particular close to junctions. Finally, to prove the generality of our approach, we apply it to boundary detection in natural images and 3D image stacks, showing an improvement also in this situation. We conclude the thesis with Chap. 7 by discussing possible further improvements and future work.

CHAPTER 2

Related Work

In this chapter, we review previous work about the centerline detection problem. We also discuss techniques designed for boundary detection in natural images and membrane detection in biomedical image stacks, which relate to our work. We start by describing the general characteristics of these methods and then describe in detail the ones most relevant to our work.

Centerline and boundary detection methods take an image as input and share the common goal of producing as output another image, called *score map* or *score image*, in which pixel values represent the likelihood that a centerline point or a boundary point lies at that pixel. Once the score map is computed, centerlines or boundaries can be directly extracted from the score map by a Non-Maximum Suppression (NMS) operation, followed by thresholding [157, 37]. Otherwise, in the case of centerline detection, the score map can be used as input to a tracing algorithm [5, 18, 214, 206, 19, 195] to obtain a graphical representation of the linear structure. In the case of membranes and boundaries, instead, the score map can be fed to a watershed or more advanced segmentation algorithm [72, 11].

In general, centerline and boundary detection algorithms are composed of two main steps: 1. A feature extraction step, which creates a representation of the image where

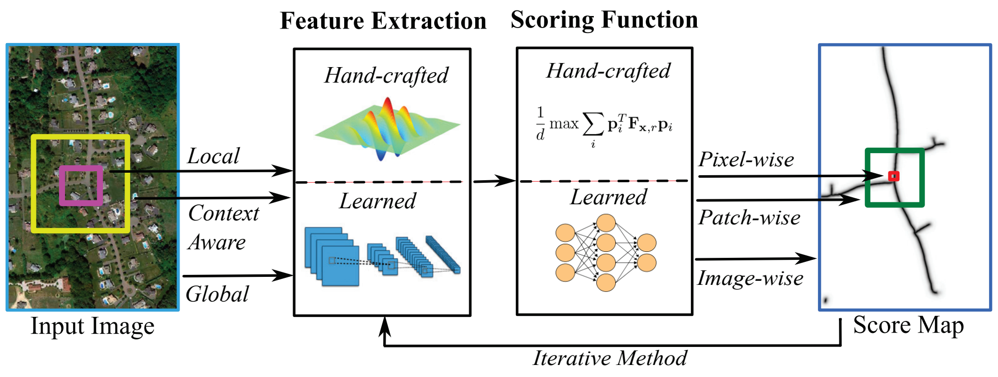


Figure 2.1: Our taxonomy of centerline and boundary detection pipelines. We divide the pipelines in two main steps: the feature extraction step and the scoring step. We distinguish between local, context-aware and global methods; between methods with hand-crafted or learned features or scoring function; and between pixel-wise, patch-wise and image-wise methods. See text for a detailed description.

the structures of interest can be easily discriminated from other structures present in the image; 2. A scoring step, in which, by taking advantage of the representation created by the feature extraction step, the structures of interest are enhanced and the background suppressed. These two steps can characterize specific algorithms in the following ways (Fig. 2.1):

1. **Feature Extraction Step**. *Hand-Crafted vs. Learned Features.* For each pixel in the input image, a set of features is extracted from a local neighborhood of the pixel (*e.g.* by convolution with a filter bank). Depending on the algorithm, the features can be hand-crafted, as for [67], learned, as for [23], or a combination of both, as in [162]. Hand-crafted features are based on an ideal profile of the linear structure or of the boundary and work best when the assumptions they rely on are satisfied. Learned features are general, in the sense that they do not make specific assumptions on the shape of the linear structure, but require training data and often need to be recomputed for every dataset.

   Local vs. Context-Aware Features. The size of the local neighborhood used to compute the features can vary from few pixels, as for early operators [56, 34] or be very large, as in the case of methods exploiting contextual information [38, 176]. In the limiting case, it can even coincide with the whole image [211, 165]. Using a small size has the advantage of being computationally efficient, but context-aware or global approaches can solve ambiguities that are not possible to discriminate only with local information. However, they have larger computational cost or rely on approximations to make computation feasible.

   Deep vs. Shallow Features. In Deep Learning methods, descriptors are computed by repeating iteratively simple feature operators. In this way, general functions can be represented using less parameters compared to a shallow architecture. This is the case of Convolutional Neural Networks (CNNs) [116], where convolutions and non-linearities are stacked in consecutive layers. Deep architectures generally work better than shallow ones for high level vision tasks [111, 125, 85], but require huge amount of training data in order to generalize well.

2. **Scoring Step**. *Hand-Crafted vs. Learned Score.* Once the features are computed, they are combined in a specific way to obtain the score map. In the case of hand-crafted methods, the score function is either designed by hand, as in [67], where the eigenvalues of the Hessian matrix are weighted using a specific combination of exponential functions, or deduced mathematically to fulfill some optimality criteria, as in [34, 93, 113].

   By contrast, in the case of Machine Learning based approaches [76, 216, 10, 160, 54], the scoring function is automatically learned. For example, in [160] Sparse Code Gradients are combined using a linear SVM to classify boundary pixels. Learning the function allows to automatically select the most relevant features for a given task and dataset. However, in opposition to hand-crafted functions, their results are not easy to interpret.

   Pixel-Wise vs. Patch-Wise Score. The score map is often computed for each pixel independently, as for pixel-wise classifiers [76, 160] and hand-crafted methods [67, 113]. This approach has the advantage of being computational efficient. However, it does not take advantage of the strong correlation that exists between nearby pixels. Other approaches instead directly predict the shape of the linear structure, or of the boundary, for an entire patch around a pixel. By averaging the patches on their overlapping areas, these methods obtain more consistent results [74, 54, 81]. In the limiting case, the whole image, or very large portions of it, can be directly predicted [211, 165].

Finally, steps 1 and 2 can be repeated iteratively, as in cascade of classifiers or Auto-Context based approaches [193, 176] to obtain more accurate results at every iteration (Fig. 2.1).

In the following we discuss some of these methods more in details. We divide centerline detection methods into two main categories, those that use hand-designed filters and those that learn them from training data. We review them in Sec. 2.1 and Sec. 2.2 respectively. We discuss work on boundary and membrane detection in Sec. 2.3.

2.1 Hand-Crafted Filters for Centerline Detection

Most of hand-crafted methods for centerline detection are based on the specification of a local descriptor designed to have a strong response when computed on line-like structures. Taking advantage of the symmetries of the problem, the local descriptor can be designed to be rotation invariant [67, 113, 194], or represented by a template that can be scaled and rotated[93]. In this case, by applying the template at different scales and orientations, one can obtain information about the radius and the angle of the line structure. Other approaches also parametrize curvature information in the descriptor [15, 131].

One of the simplest hand-crafted methods is given by [65]. In this work, only Gaussian smoothing is applied to the image. The result is interpreted as an approximated distance transform, where (bright) linear structures correspond to the peaks of the distance. Then, NMS is directly applied to the smoothed image. If the image corresponded to an exact distance transform of the centerlines, then this approach would output the exact centerline points. In Chap. 3, we build on this idea and try to predict the distance transform of the centerlines directly form the image data.

More recent Hand-Crafted filters for centerline detection can be divided into two main categories. The first one is made of Hessian-based approaches [67, 171, 110, 169, 139, 66, 51, 143]. They combine the eigenvalues of the Hessian to estimate the probability that a pixel or voxel lies on a centerline. They rely on the fact that, when computed on an elongated structure with an ideal Gaussian intensity profile, one of the eigenvalues will have much smaller absolute value than the others. The main drawback of these approaches is that the required amount of Gaussian blur to compute the Hessian may result in confusion between adjacent structures, especially when they are thick. Also, the radial estimation of these approaches is not accurate, especially for large scales [57].

This has led to the development of a second class of methods based on Optimally Oriented Flux (OOF) [113]. They rely on the second order derivatives of an n-dimensional ball and are less sensitive to the presence of adjacent structures. Moreover, the radius of the ball provides a reliable estimate of the tubular structure scale. Remaining difficulties, however, are that OOF can also respond strongly to edges as opposed to centerlines and that its performance degrades when the structures become very irregular. A number of schemes have been proposed to solve the first problem [2, 184, 152, 114, 210]. For example, in [114], an Oriented Flux Antisymmetric (OFA) term was added and has proved effective. There has been less work on improving OOF's performance on truly irregular structures, except for the very recent approach of [194] that attempts to maximize the image gradient flux along multiple radii in different directions instead of only one as in [113].

The method proposed in [219] can be seen as a mixture of these two classes. Hessian computation implicitly assumes an ellipsoidal model whereas in [219] the ellipsoid is explicitly fitted to the data. Because this is harder to do than fitting OOF balls, it is achieved by a learning a regression model from image data to ellipse parameters.

2.2 Learning Filters for Centerline Detection

Even if care is taken to add computational machinery to handle irregular structures [169, 194], the performance of hand-designed filters tends to suffer in severe cases such as the one depicted by Fig. 3.1. This is mostly because it is very difficult to explicitly model the great diversity of artifacts that may be present.

Some works therefore aim at segmenting linear structures in biomedical images [76, 216, 162, 23] or aerial ones [145, 207] by applying classification to label the pixels or voxels as belonging to the structure of interest or to the background. However, this is a problem simpler than the one we consider. It is not accurate to find the actual centerlines and radii from the segmentation even with post-processing operations. In particular, there is no guarantee that the classifier responses will be maximal at the centers of the structures. By contrast, if we can recover the centerlines and the corresponding thickness of the linear structures, it is straightforward to generate a segmentation from this data. This is therefore the approach we adopt in the thesis.

Other techniques, such as [90, 204, 218, 29], aim at extracting the centerlines, but still rely on binary classification to distinguish the image locations on centerlines from the rest. [218, 29] use Haar wavelets in conjunction with boosted trees to detect the centerlines of tubular structures at different scales. [90] uses spectral-structural features instead and SVMs to find road centerlines. In [204] co-occurrence features and the AdaBoost algorithm are used to detect the spinal column centerline.

These methods exhibit limited localization accuracy because points near the centerlines can easily be also classified as centerline points due to their similar appearance. As we will show, our approach based on regression rather than classification is more adapted to the problem at hand. Moreover, most of these machine learning based approaches do not consider the problem of radial estimation. Our approach instead returns a scale-space tubularity score, similar to that of hand-crafted methods [113, 194], designed to have a maximal response at centerline points along the spatial and radial dimensions.

Machine Learning based approaches rely on a feature extraction step, which facilitates the classification of the pixels in the image. Several kind of features have been used in combination with different classifiers. For example, the same hand-crafted features discussed in Sec. 2.1 can be used to train a Machine Learning system. Other popular choices are Haar wavelets, Gabor filters or steerable filters [68, 93, 76].

Automatic feature learning is another way to select features. It has long been an important area in Machine Learning and Computer Vision. Neural Networks [116], Restricted Boltzmann Machines [88], Auto-Encoders [24], Linear Discriminant Analysis (LDA) [27], and many other techniques have been used to learn features in either supervised or unsupervised ways. Among supervised feature learning techniques, Convolutional Neural Networks are the most commonly used in Computer Vision today, and they are discussed below. In [23] instead, a boosted tree based approach is used in combination with an LDA in order to find the filters giving the best separation of the data of a given node in the tree.

Among unsupervised methods, creating an overcomplete dictionary of features—sparse combinations of which can be used to represent images—has emerged as a powerful tool for many different purposes [40, 104, 209, 58, 128] and linear structure segmentation [164, 162] in particular.

However, for most such approaches, run-time feature extraction can be very time-consuming because it involves convolving the image with many non-separable non-sparse filters. In this work we will use an unsupervised convolutional filter learning approach to learn an expressive filter bank, thus taking advantage of the large quantity of unlabeled training data of the biomedical domain. Then, we will approximate this filter bank with a smaller set of separable filters, thus greatly reduce computational complexity.

Even if powerful features in combination with pixel-wise methods can produce remarkable results, they do not explicitly model the strong relationship that exists between neighboring pixels. As a consequence, discontinuities and inconsistencies may occur in their output. To overcome these limitation contextual information [193] or structured prediction based approaches [107] can be exploited to obtain more consistent results. For example, the authors of [80], inspired by [121], use a latent tree model to classify filament fragments in biomedical images. In this thesis, we propose a novel and efficient patch-wise prediction method which approximates the projection of a pixel-wise score map onto the set of linear structures ground truth images.

Finally, in the last years Deep Convolutional Networks [116] have become very popular thanks to their impressive results on many benchmark datasets [39, 38, 111, 165] and they have been recently applied to the problem of vessel segmentation in retinal images [71, 120]. They do not require a specific set of features, but directly learn them from the training data. Moreover, thanks to their particular architecture, they can consider large portion of the image in input and exploit contextual information. Typically, a Deep Convolutional Network is trained using back-propagation to minimize the classification error.

However, because of the huge number of parameters to be optimized, Deep Networks require extremely large amounts of labeled training data and computational power to reach state-of-the-art performance. When the input image is too large or only limited amounts of training data are available, which is the typical situation in the biomedical domain, their applicability and performance are reduced.

By optimizing the features in an unsupervised way and by adopting an iterative prediction approach, our method is easier to train. Moreover, it can efficiently process large input volumes.

Structured and contextual information, as well as Convolutional Neural Networks, has also been applied to detect boundaries in natural and medical images. We discuss these approaches in the next section.

2.3 Boundary and Membranes Detection

Edge detection is one of the most widely studied problem in Computer Vision. Boundary detection methods can be divided in the same two classes as for centerline detection. Early attempts at solving it belong to the first one and are based on filters designed to respond to specific image intensity profiles [135]. The resulting algorithms [34, 50, 154, 149] are still in wide usage. However, attention has recently shifted to classification based methods [10, 52, 160, 127, 121, 53, 176], which have produced significant improvements.

More specifically, in [10] gradients on different image channels are fed to a logistic regression classifier to predict contours in natural images. In [160], SVMs are trained to predict contours from features computed using sparse coding. In [52], a boosting algorithm is used to predict the probability of a boundary.

Deep Convolutional Networks have also been applied to the boundary and membrane detection problem. Some of them try to classify local patches to predict whether the central pixel of the patch lies on a boundary point or not [38, 177, 26, 91]. However, their performance is comparable to shallower architectures [54], probably because of the small size of the training set they rely on for training [92, 10] and because of the local nature of these approaches. In fact, a large field of view is necessary to correctly interpret the content of the image and discriminate object boundaries.

In order to improve performance of local classification based approaches, cascades of classifiers [62, 86, 189, 100, 175, 176, 193] have been used by several authors. Typically, in such frameworks, one or more classifiers are applied sequentially to the input signal to capture more contextual information and improve classification rate at each iteration. In these architectures, each layer is trained with a set of features extracted from the output score map of the previous layer, the first layer score map being the raw image. The training is easier than Deep Learning models since each classifier is treated separately. For example the approach of [176] relies on a cascade of classifiers at different resolutions to predict a boundary map. It is related to our approach in the sense that we also use cascades but, as for centerline detection, we will show that we outperform it because regression is more appropriate than classification in this context.

However, none of these methods explicitly model the relationship between nearby pixels. In particular, the response of Convolutional Neural Networks [116] can be spatially inconsistent because they typically treat every pixel location independently, thus relying only on the fact that neighboring patches share pixels to enforce consistency. This problem can be mitigated by applying post processing operations, such as median filtering, or by averaging the output of several models, trained independently [38].

Patch-wise prediction methods [121, 53, 74, 54, 81] are an efficient way to overcome these problems. For example, the method of [54] relies on structured learning, resulting in an accurate and extremely efficient edge detector. It is inspired by the work of [107, 108] where the structured random forest framework is introduced for image labeling purposes, predicting for every pixel an image patch, instead of single pixel probabilities. However, it is specific to the particular kind of classifier used for learning and is difficult to generalize.

Recently, Nearest Neighbors search in the space of local descriptors obtained with a Convolutional Neural Network was used for boundary detection purposes [74]. Given an image patch, the algorithm computes a corresponding descriptor and then looks for the Nearest Neighbor in a dictionary built from the training set. While effective, this approach strongly depends on the specific dictionary learned by the network. Therefore, when it fails, it is difficult to understand why. By contrast, in this thesis we propose an efficient patch-wise projection method based on Nearest Neighbors search in the space of the final output, rather than of intermediate image features. We will show empirically that this approach works better than previous patch-wise methods, especially near junctions. Moreover, unlike other approaches, we provide optimality conditions, under which our method returns solutions with the same geometrical properties of the ground truth images, thus giving insights and indications on how to improve the results.

Membrane detection approaches in biomedical image stacks are based on the same principles of boundary detection methods for natural images. Membranes are the 3D equivalent of contours in image stacks. They are important for 3D volume segmentation, especially in a biomedical context [201, 9, 72]. Early approaches to detecting them relied on hand-crafted filters optimized to respond to ideal sheet-like structures. In [172, 146, 137, 141] for example, the eigenvalues of the Hessian matrix are combined to obtain a score value that is maximal for voxels lying on a 2D surface. Similarly, the eigenvalues of the Oriented Flux matrix [113] can be combined to obtain a score that is less sensitive to adjacent structures. These hand-crafted methods have the same limitations described in the case of centerline detection.

More recent approaches have focused on machine learning techniques. For example, a Convolutional Neural Network and a hierarchical segmentation framework combining Random Forest classifier and watersheds are used in [96] and [8] respectively to segment neural membranes. Even though both of these methods produce excellent results, they are designed for tissue samples prepared with an extra-cellular die that highlights cell membranes while suppressing the intracellular structures, thus making the task comparatively easy.

The approach of [175] was also applied segmenting membranes in Electron Microscope images. It was improved in [176], where, thanks to a hierarchical architecture, they obtain lower pixel error than Deep Convolutional Networks on the ISBI 2012 Challenge [92].

Finally, very recent techniques, based on a Fully Convolutional architecture [125], overcome the limitation of local methods by considering as input to the network, not only local patches, but the whole image [211]. They have the double advantage of considering global contextual information and to output a segmentation mask for the whole image, making the final segmentation more consistent. In [211] the 16 layers VGG architecture [178], pre-trained on the ImageNet [45] dataset, was modified and fine-tuned on the BSDS500 dataset [10] in order to output boundary score maps. The semantic information encoded in the network and the large field of view make this approach the state-of-the-art for boundary detection. However, since they rely on binary classification they suffer of the thick boundary problem.

A similar idea is used in [165] to classify membrane pixels in 2D slices of Electron Microscopy data. In this case, data augmentation is used during training to avoid overfitting. This approach outperform previous techniques, based on CNN classification of local patches [38] both in accuracy and run time efficiency.

These techniques could be potentially combined with the regression based approach introduced in this thesis and extended to operate also on 3D volumes in order to obtain an even more accurate centerline detector. A limiting factor, however, are the memory capabilities of modern graphical processors (GPUs), needed to train and run these large models. As a consequence, it is not possible to train them, unless by considering small 3D input volumes, thus losing the advantage of using a deep architecture and global contextual information.

CHAPTER 3

Multiscale Centerline Detection by Learning a Scale-Space Distance Transform

In this chapter, we formalize the centerline detection problem and propose a regression-based approach to accurately extract centerlines and radii from images. As stated in the previous chapter, most existing techniques rely on filters designed to respond to locally cylindrical structures [67, 171, 110, 139, 114, 210], optimized for specific profiles [93], or learnt [169, 76, 23]. They compute a scale-dependent measure that, ideally, should be maximal at the centerline of linear structures when computed at the right scale.

Among these approaches, the learning-based ones tend to outperform the hand-designed ones when the linear structures become very irregular and deviate from the idealized models on which their design is based. Some works only aim at segmenting the linear structures from the background [23], and it is not clear how to reliably extract the centerlines from the segmentation. Others focus on the centerlines, but they typically rely on classification and this results in poor localization accuracy. As shown in Fig. 3.1, this is because it is hard for the classifier to distinguish points on the centerline itself from those immediately next to it.

In this chapter, we show that this problem can be solved by reformulating centerline detection and radius estimation in terms of a regression problem. More precisely, we train several regressors to return distances to the closest centerline in scale-space, each regressor being trained for a specific scale. In this way, performing Non-Maximum Suppression on their output yields both centerline locations and corresponding radii. We will show that, on very irregular structures, it outperforms the powerful OOF approach with and without

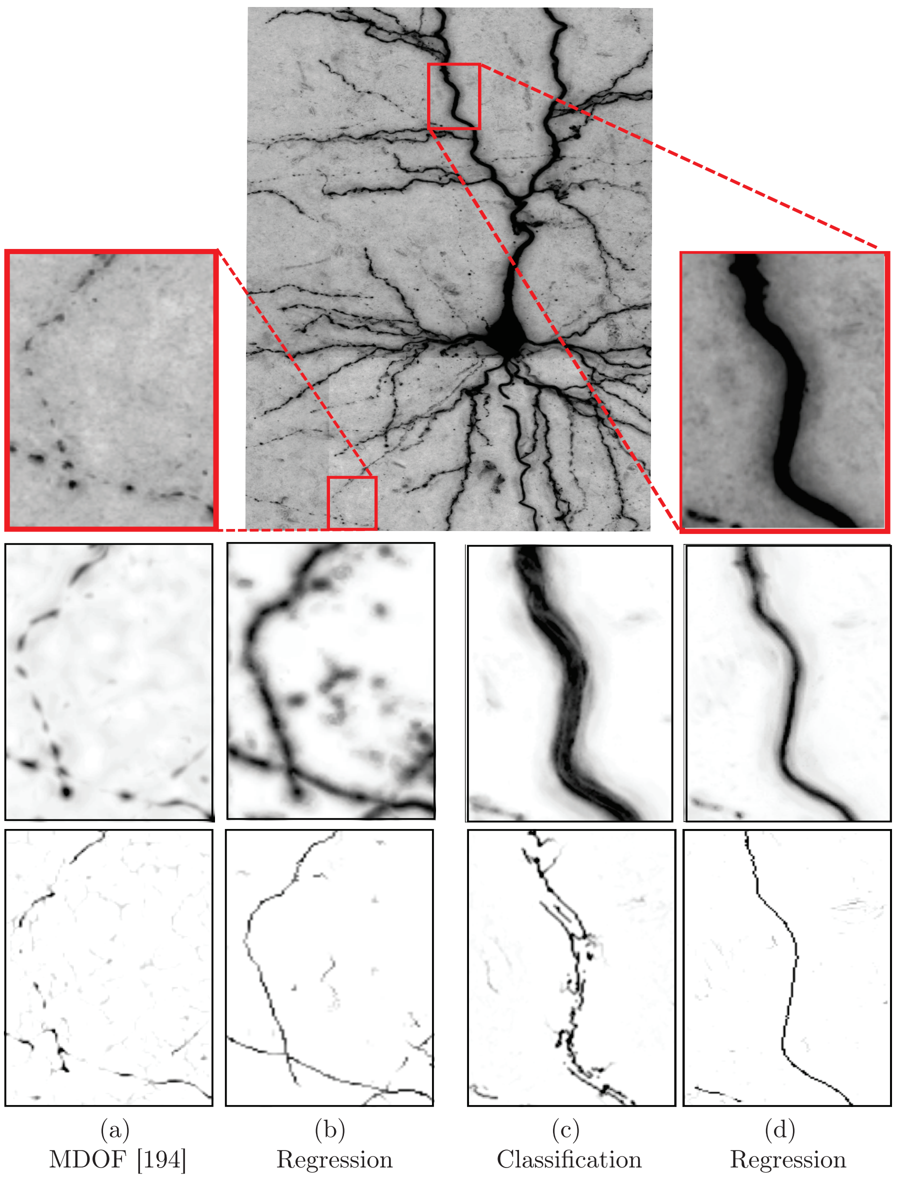


Figure 3.1: Detecting dendrites in a 3D brightfield image stack. **Top row:** Minimal intensity projection with two enlarged details. **Middle row:** Comparison of the responses of our method against a recent model based approach [194] and a classification based one [23]. **Bottom row:** Centerlines detected after performing Non-Maximum Suppression on the response images. (a) Model-based methods have troubles modeling highly irregular structures. (c) Classification-based approaches respond on the whole body of the tubular structure and do not guarantee maximal response at the centerline. (b,d) Our regression-based method combines robustness against image artifacts and accurate centerline localization. In the last two rows, images have been inverted for visualization purposes.

anti-symmetry term [113, 114], which is widely acknowledged as one of the best among those relying on hand-designed filters, as well as a recent extension of it [194] designed to improve its performance on irregular structures, and a similarly recent classification-based method [23].

The chapter is organized as follows: In Sec. 3.1 we formalize centerline detection as a Machine Learning problem using a standard classification-based formulation. Then, in Sec. 3.2 we introduce our regression-based approach, showing how a modified distance transform of the centerline can be used to learn a function to achieve more accurate localization. We first introduce our algorithm in the case of single scale detection. Then, in Sec. 3.3, we consider the multiscale case. In Sec. 3.4, we describe the features used to train our model. Finally, in Sec. 3.6, we show empirically the superiority of our approach by evaluating it on several 2D and 3D biomedical and natural images datasets, both for centerline detection and radial estimation.

Part of the content of this chapter has been previously published in [179].

3.1 Centerline Detection as a Classification Problem

In this section we formalize the centerline detection problem as a Supervised Learning problem. First, we introduce the commonly used *classification*-based formulation. Then, we argue why classification is not the best formalism for this problem and, in the next section, we show how this limitations can be overcome by rewriting centerline detection as a *regression* problem.

Let $I(\mathbf{x})$, be an n-dimensional image containing curvilinear structures of various radii, where $\mathbf{x} \in \mathbb{R}^n$ are the pixel coordinates. Let $Y_I(\mathbf{x})$ be the binary image, of the same size of I, corresponding to the centerline pixels, that is, Y_I is such that $Y_I(\mathbf{x}) = 1$ if pixel $\mathbf{x}$ is on a centerline and $Y_I(\mathbf{x}) = 0$ otherwise. We will refer to Y_I as the *binary ground truth* or the *manual annotation* of I.[1] In the remainder, unless there are ambiguities, we will simply write Y for Y_I.

A classification-based approach to finding the centerlines involves learning a function $y(\cdot)$ mapping I to Y_I. Following our taxonomy of Chap. 2, in this section we consider $y(\cdot)$ to be a local pixel-wise classifier, *i.e.* $y(\cdot)$ predicts the class of each pixel $\mathbf{x}$ independently, starting from a feature vector $f(\mathbf{x}, I) \in \mathbb{R}^J$ computed from a neighborhood of size s surrounding

[1] We assume here that the location of the centerline is well defined and that there exists a unique $Y_I(\mathbf{x})$ for every I. We will discuss in Sec. 3.6.2 how to account for uncertainty in the location of the centerlines and in Sec. 5.6 on how to deal with multiple manual annotations.

pixel $\mathbf{x}$ in image I. More precisely, the ideal classifier y is given by

$$y(f(\mathbf{x}, I)) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is on a centerline,} \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

The function $y(\cdot)$ is typically approximated by minimizing a loss $\mathcal{L}$ over a set of training samples. Given a set of training images $\{I_k\}_{k=1}^{K}$ and the corresponding binary ground truth images $\{Y_k\}_{k=1}^{K}$, $y(\cdot)$ is approximated by a function $\psi^*(\cdot)$, obtained by minimizing the loss $\mathcal{L}$ over training samples $\{f_i, \ell_i\}_{i=1}^{Q}$, with features $f_i = f(\mathbf{x}_i, I_i)$ and labels $\ell_i = Y_i(\mathbf{x}_i)$:

$$\psi^* = \underset{\psi \in \mathcal{H}}{\arg\min}\, \mathcal{L}(\{f_i, \ell_i\}_i; \psi), \tag{3.2}$$

where $\mathcal{H}$ is a space of functions where the minimization is carried out.

Usually, to make the optimization easier, the loss $\mathcal{L}$ is taken to be a sum over the samples: $\mathcal{L} = \sum_i L(f_i, \ell_i; \psi)$. In the case of probabilistic interpretation of the loss, this is equivalent to assuming the samples to be independent and identically distributed [82].

The choice of L and $\mathcal{H}$ depends on the particular algorithm considered. For example in Linear Support Vector Machines [42], L is the Hinge Loss and $\mathcal{H}$ the space of real-valued linear functions on $\mathbb{R}^J$. In the AdaBoost algorithm [69], instead, L is the Exponential Loss and $\mathcal{H}$ is given by a linear combination of decision stumps.

As discussed above, learning directly the function $y(\cdot)$ is hard because points on the centerline itself, for which $y(\cdot)$ should return 1, and their immediate neighbors, for which it should return 0, look very similar (Fig. 3.1). One way to solve this is to train $y(\cdot)$ to return 1 for all points within a given distance from the centerline. However, in practice, even if $y(\cdot)$ is allowed to return floating point values between zero and one, using for instance an SVM-style classifier, there is no guarantee that its value will be maximal at the centerline itself. This makes finding its accurate location, for example by using Non-Maximum Suppression, problematic.

Ideally, a way to overcome this limitation would be to constraint the output of the classifier to have local maxima corresponding to centerline locations. For example, we could use a structured learning approach, such as SSVM [192], to learn a function $y(\cdot)$ ranking the training samples so that $y(f(\mathbf{x}_i, I_i)) < y(f(\mathbf{x}_j, I_j))$ if and only if the distance of $\mathbf{x}_i$ to a centerline point in image I_i is larger than the distance of $\mathbf{x}_j$ to a centerline point in I_j. This approach has the advantage that we do not need to specify a function to learn, instead, it only imposes the centerline points to be local maxima.

Table 3.1: Main mathematical notations used in Chap. 3.

Notation	**Meaning**
$I(\mathbf{x})$	Input image (resp. volume) at pixel (resp. voxel) $\mathbf{x}$
$f(\mathbf{x}, I)$	Feature vector computed on image I, at pixel $\mathbf{x}$
$Y_I(\mathbf{x})$	Binary ground truth image of the centerlines in I
C	Set of centerline pixels for a given image
$y(f(\mathbf{x}, I))$	Ideal pixel-wise classifier: $y(f(\mathbf{x}, I)) = 1$ iff $\mathbf{x} \in C$
$\mathcal{D}_C(\mathbf{x})$	Euclidean distance transform of the set C at pixel $\mathbf{x}$
$d_C(\mathbf{x})$	Ideal regressor response. Exponential scaling of $\mathcal{D}_C$
$\varphi(f(\mathbf{x}, I))$	Actual regressor response
$y(\cdot; r), \mathcal{D}_C(\cdot; r), d_C(\cdot; r), \varphi_r$	As above, but for centerlines corresponding to tubular structures of radius r

However, learning a function using this ranking formulation is difficult in practice. In fact the associated optimization problem is extremely large to solve and involves a number of constraints that is quadratic in the number of pixels. To solve this issue and make optimization easier, in the next section we consider a fixed function satisfying the constraints we want and then we try to learn it.

More precisely, our solution is to learn instead $y(\cdot)$ as a *regressor* whose values decrease monotonically as the distance of point $\mathbf{x}$ to the centerline increases. Then, as required, the local maxima of this function coincide with the centerline and, as shown in Fig. 3.2, we can rely on simple Non-Maximum Suppression to localize them. We will show in Sec. 3.6 that this solution is significantly more robust than both classification-based and hand-crafted methods.

Moreover, many automated and semi-automated tracing algorithms [25, 148, 197] rely on the extraction of local maxima from a tubularity measure as an initial step. Our method is designed to return a score with a well defined maximum along the centerlines and therefore it can be used as input to improve the accuracy of these methods. We demonstrate the advantage of using our method in this context in Chap. 5.

In the next section, we describe our method for structures whose scale is assumed to be known *a priori*. We then relax this constraint to handle structures of arbitrary scale in Sec. 3.3. The main notations used in this chapter are summarized in Table 3.1.

3.2 Centerline Detection as a Regression Problem

Let us momentarily assume that the linear structures have a known radius r. Let $C = \{\mathbf{x} : Y(\mathbf{x}) = 1\}$ be the set of centerline points and $\mathcal{D}_C$ the corresponding Euclidean distance

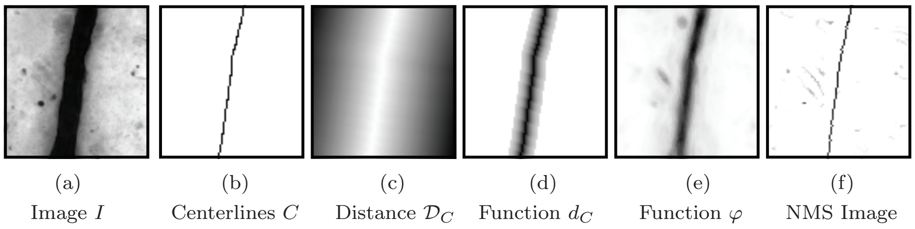


Figure 3.2: Learning a regressor for centerline detection. (a) Raw image; (b) Ground truth centerlines; (c) The distance transform to the centerline is used to discriminate points close to it; (d) The function we want to learn is maximal at the centerlines and it is thresholded to a constant value when the local window used to compute features does not contain any centerline points; (e) The function learned with our method; (f) Centerline detected after Non-Maximum Suppression (NMS) on function φ. In images from (b) to (f), white indicates lower values.

transform [166], that is, $\mathcal{D}_C(\mathbf{x})$ is the metric distance from location $\mathbf{x}$ to the closest location in C:

$$\mathcal{D}_C(\mathbf{x}) = \min_{\mathbf{x}' \in C} \|\mathbf{x} - \mathbf{x}'\|_2, \tag{3.3}$$

where $\|\cdot\|_2$ is the standard Euclidean norm: $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$. We assume here, as in [166], that C is non-empty. Image 3.2(c) shows the distance transform for a simple centerline image.

Our goal is to learn a function that is maximal for $\mathbf{x}$ on the centerline and whose value decreases monotonically as $\mathbf{x}$ moves away of it. The function $d_C(\mathbf{x}) = -\mathcal{D}_C(\mathbf{x})$ has this property, see Fig. 3.3. In theory, given training data, we could learn a regressor that takes $f(\mathbf{x}, I)$ as input and returns $-\mathcal{D}_C(\mathbf{x})$ as output. However, in practice, we learn a different function for the two following reasons.

First, because our feature vectors $f(\mathbf{x}, I)$ are computed using local neighborhoods of size s, a regressor could only learn it for points that are close enough to the centerlines for their neighborhood to be affected by it. For this reason, it makes sense to threshold d_C when $\mathcal{D}_C$ is greater than a given value d_M, which is a function of the neighborhood size s. This yields the modified function

$$d_C(\mathbf{x}) = \begin{cases} 1 - \frac{\mathcal{D}_C(\mathbf{x})}{d_M} & \text{if } \mathcal{D}_C(\mathbf{x}) < d_M \\ 0 & \text{otherwise,} \end{cases} \tag{3.4}$$

which takes values between 0 and 1, see Fig. 3.3. In our implementation, we set $d_M = s/2$, which means that d_C is uniformly 0 for points whose corresponding neighborhood does not overlap the centerline.

Second, a regressor trained to associate to a feature vector $f(\mathbf{x}, I)$ the value of $d_C(\mathbf{x})$ can only do so approximately. As a result, there is no guarantee that its maximum is

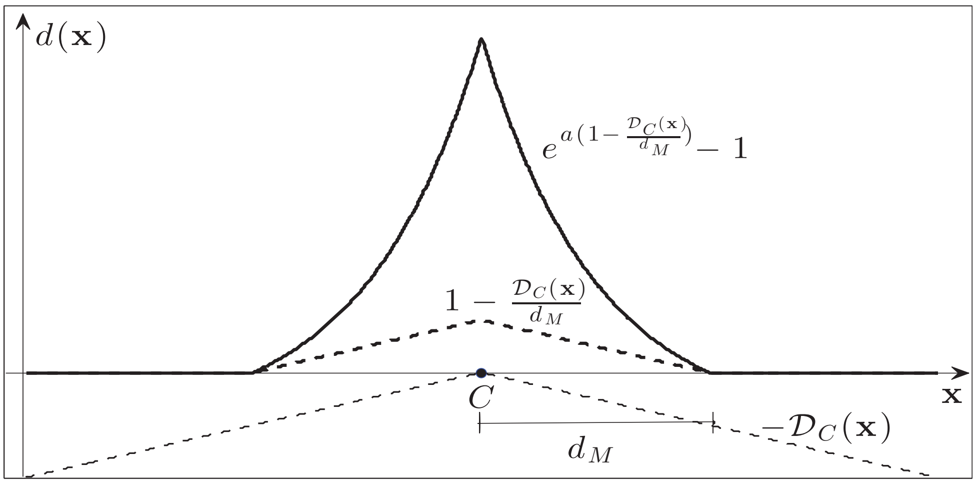


Figure 3.3: The function d_C in the case of $\mathbf{x} \in \mathbb{R}$. If a centerline point is located in C, the function we want to learn is obtained from the distance transform $\mathcal{D}_C$, after thresholding and scaling. The vertical axis has been scaled for visualization purposes.

exactly on the centerline. To increase robustness to noise, we have therefore found it effective to train our regressor to reproduce a distance function[2] whose extremum is better defined. In our actual implementation, we take it to be

$$d_C(\mathbf{x}) = \begin{cases} e^{a(1-\frac{\mathcal{D}_C(\mathbf{x})}{d_M})}-1 & \text{if } \mathcal{D}_C(\mathbf{x}) < d_M \\ 0 & \text{otherwise,} \end{cases} \tag{3.5}$$

where $a > 0$ is a constant that controls the exponential decrease rate of d_C close to the centerline, see Fig. 3.3. In all our experiments, we set $a = 6$.

The regression method we use to learn function d_C is the GradientBoost algorithm [82]. It can be viewed as a generalization of the AdaBoost algorithm and it can efficiently approximate very complex functions.

Given training samples $\{(f_i, d_i)\}_i$, where $f_i = f(\mathbf{x}_i, I_i) \in \mathbb{R}^J$ is the feature vector corresponding to a point $\mathbf{x}_i$ in image I_i and $d_i = d_C(\mathbf{x}_i)$, GradientBoost approximates $d_C(\cdot)$ by a function of the form

$$\varphi(f(\mathbf{x}, I)) = \sum_{t=1}^{T} \alpha_t h_t(f(\mathbf{x}, I)) \,, \tag{3.6}$$

where $h_t : \mathbb{R}^J \to \mathbb{R}$ are weak learners and $\alpha_t \in \mathbb{R}$ are weights. The function φ is built iteratively, selecting one weak learner and its weight at each iteration, to minimize a loss function $\mathcal{L}$ of the form $\mathcal{L} = \sum_i L(d_i, \varphi(f_i))$. In our experiments we consider $L(\cdot)$ to be the

[2]Formally, function d_C of Eq. (3.5) is not a distance. However, with an abuse of terminology, we will refer to it as a distance transform in the rest of the manuscript.

squared loss function $L(d_i, \varphi(f_i)) = \frac{1}{2}(d_i - \varphi(f_i))^2$. We also experimented with the l_1 loss and Huber loss functions and the results proved to be very similar.

Alg. 1 shows the pseudo-code of the GradientBoost algorithm in the case of a generic loss function. In the case of squared loss L, the residuals r_i^t of Step 4, are equal to $r_i^t = d_i - \varphi^{t-1}(f_i)$, while the weights α_t can be computed in closed form as

$$\alpha_t = \frac{\sum_i h_t(f_i)\left(d_i - \varphi^{t-1}(f_i)\right)}{\sum_i \left(h_t(f_i)\right)^2}. \tag{3.7}$$

As usually done with GradientBoost, we use regression trees as weak learners h_t since they achieve state-of-the-art performance in many applications [82]. The regression trees are learned one split as a time, as in [82].

Unless otherwise stated, in all our experiments we used $T = 250$ trees of depth 2. Fig. 3.2 shows the output of the learned function for a sample image. For simplicity, unless there are ambiguities, we will write $\varphi(\mathbf{x})$ instead of $\varphi(f(\mathbf{x}, I))$ and $h_t(\mathbf{x})$ instead of $h_t(f(\mathbf{x}, I))$. Moreover, we will refer to the image obtained by applying the regressor φ to every pixel of an image I as the *score map* or *score image* corresponding to I.

Algorithm 1 GradientBoost Training

```
1: **Input**: Training Set $\{(f_i, d_i)\}_i$, Number of Iterations $T$
2: Set $\varphi^0 = 0$
3: **for** $t = 1$ to $T$ **do**
4:     Let $r_i^t = -\frac{\partial L(d_i,\phi)}{\partial \phi}\Big|_{\phi=\varphi^{t-1}(f_i)}$
5:     Find weak learner
           $h_t(\cdot) = \arg\min_{h(\cdot)} \sum_i \left(h(f_i) - r_i^t\right)^2$
6:     Find weight $\alpha_t$
           $\alpha_t = \arg\min_{\alpha} \sum_i L\left(d_i, \varphi^{t-1}(f_i) + \alpha h_t(f_i)\right)$
7:     Let $\varphi^t(\cdot) = \varphi^{t-1}(\cdot) + \alpha_t h_t(\cdot)$
8: **end for**
9: **Return** $\varphi(\cdot) = \varphi^T(\cdot)$
```

In next section we describe how radial information can be encoded in the score map, and how the radius of the linear structures can also be learned from a modified multiscale distance transform.

3.3 Multiscale Centerline Detection

In the previous section, we focused on structures of known radius. In general, however, structures of many different radii are present in the same image and we would like to estimate the scale of every centerline point (Fig. 3.4(a)).

To generalize our approach to this multiscale situation we assume that at every centerline point is associated a scale value $r = r_{\mathbf{x}} > 0$. In the case of ideal tubular structures, the cross section of the linear structure is a circle, therefore the radius r is simply defined as the radius of the cross section. In case of ellipsoidal or irregular cross section, many possible definitions of radius are possible, for example the radius can be defined as the radius of the smallest circle containing the cross section. Our method does not make any particular assumption on how the radius is defined, as we rely on manual annotations to determine its value, independently on how it was computed. Provided that the manual radial annotation is consistent across images, this makes our approach more general and more robust than hand-crafted methods, which depend on a specific model of the tubular structures.

Given centerline points and corresponding radii, we consider the *scale-space binary ground truth*, $Y(\mathbf{x}; r)$ where $Y(\mathbf{x}; r) = 1$ if and only if $\mathbf{x}$ is a centerline point with associated radius equal to r, and 0 otherwise.

Now the set of centerline points C is the set of $(\mathbf{x}; r)$ $(n+1)$-dimensional vectors of centerline points and corresponding radii. We redefine the function d_C of Eq. (3.5) as

$$d_C(\mathbf{x}; r) = \begin{cases} e^{a\cdot(1-\frac{\mathcal{D}_C(\mathbf{x};r)}{d_M})} - 1 & \text{if } \mathcal{D}_C(\mathbf{x}; r) < d_M, \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

where $\mathcal{D}_C(\mathbf{x}; r)$ is the *scale-space distance transform* of C:

$$\mathcal{D}_C^2(\mathbf{x}; r) = \min_{(\mathbf{x}',r')\in C} \|\mathbf{x} - \mathbf{x}'\|_2^2 + w(r - r')^2 \ , \tag{3.9}$$

where $w > 0$ is used to weight the scale component differently from the space component. In practice w depends on the image resolution and the range of scales. In Sec. 3.6 we discuss the choice of w.

If we consider the maximum projection of $d_C(\mathbf{x}; r)$ along the radial component, we obtain a function of $\mathbf{x}$, whose local maxima are the centerline points for all the values of r. Therefore, if we train a regressor to output the values of d_C, the problem of multiscale centerline detection is reduced to the problem of finding local maxima in the projected image, see Fig. 3.4. Moreover, function $d_C(\mathbf{x}; r)$ is defined so that points in C are local

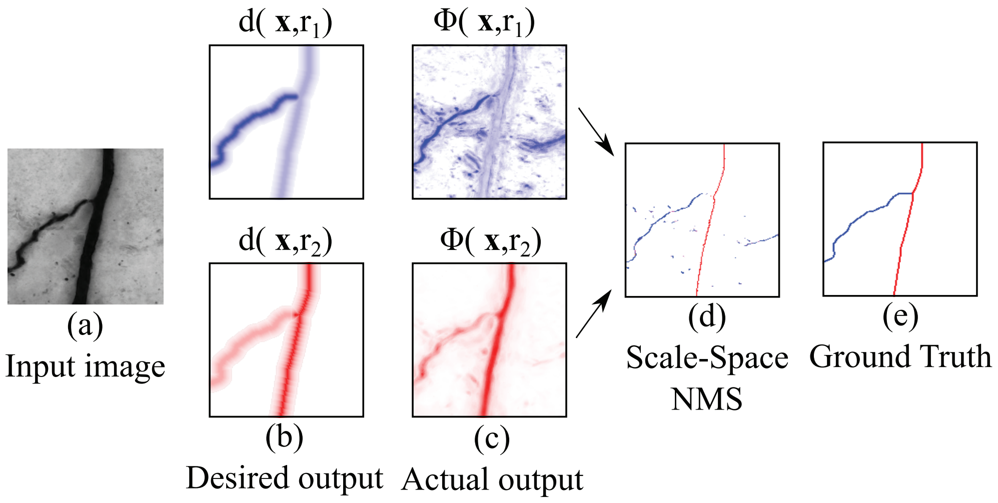


Figure 3.4: Multiscale centerline detection. (a) Input image containing linear structures at different scales. We want to learn a function with local maxima at centerline points along the spatial and radial axes. (b, top): Values of d_C for the smaller radius r_1, (b, bottom): values for the larger radius r_2. (c) The learned multiscale approximation $\{\varphi_{r_i}\}_i$ for r_1 and r_2. (d) The centerlines and the radii are detected with Non-maximum Suppression in the scale-space. (e) Ground truth centerlines. The radial values are color-coded.

maxima of d_C not only along the spatial dimensions, but also along the radial component, as shown in Fig. 3.4(b). Therefore, we can easily find the scale corresponding to a centerline point as the one that gives the maximal value for that point, Fig. 3.4(d).

We now want to learn a regressor that returns the values of this new d_C function. The simplest way would be to discretize the range of possible scales r into a finite set of scales and to use the fixed-radius method of Sec. 3.2 to learn one regressor φ_r for each scale in this set. This approach, however, decreases the number of training samples available to train each regressor, which in our experience severely impairs performance.

An alternative approach is to rely on scale-space theory [123] to train a single regressor φ_{r^0} for radius r^0. By properly scaling and normalizing the convolutional filters used to compute the feature vectors $f^{r^0}(\mathbf{x}, I)$, we can use φ_{r^0} to find the centerlines for all the other radii. The advantage of this approach is that we can exploit all training samples to train φ_{r^0} by rescaling them to have a radius equal to r^0. However, this assumes that the aspect of tubular structures is scale invariant. When this is not the case, the results are less accurate, especially for large differences between the actual radius of the structure and r^0.

We therefore adopt a hybrid approach. We learn a set of regressors $\{\varphi_{r_i}\}_{i=1}^R$ for a small set of regularly sampled radii. We then apply the scale-space approach for intermediate radii and use the closest r_i to the scale we want to predict. In Sec. 3.6 we discuss how these radii are selected.

As in the single-scale case, we build the functions φ_{r_i} using GradientBoost. In next section, we describe the set of features we use to train the GradientBoost algorithm. More details about the computation of the multiscale features are given in Sec. 3.4.1.

3.4 Efficient Features for Centerline Detection

As discussed in Chap. 2, there are many possible ways to compute the feature vectors $f(\mathbf{x}, I)$ of Eq. (3.1). In order to select the most appropriate set of features for our problem, we consider two main factors: first, the kind of data we want to process; and second, the learning algorithm the features will be fed to.

From these consideration, we deduce the following properties that our features should satisfy:

- The features must be expressive enough to discriminate linear structures with irregular and diverse appearance, from other similar background structures, both in 2D and 3D images.
- Since we use the GradientBoost algorithm for learning, we should use a large and diverse pool of features. In fact, Boosting algorithms have the best performance when they can choose, at each iteration, the optimal feature for splitting the regression trees, among a large set of candidates.
- The features extraction step must be efficient enough to process large datasets of 2D and 3D images.

In summary, we need a large number of expressive features that can be computed efficiently.

Previous work [162] has shown that learning a set of convolutional filters via sparse coding techniques can produce expressive features that perform well on linear structures. The number of such features is a parameter of the method, therefore, it can be, in theory, arbitrary large. This gives us the first two properties we wanted, namely expressive features and a large number of them. In order to also make features extraction computational efficient, we consider a separable filter approximation scheme. Thanks to this algorithm, we can approximate large filter banks with a smaller set of separable filters, thus greatly reducing computational time, with small or no loss in performance. In the rest of this section we briefly describe how the feature vector is computed. We will discuss in detail how the filter banks are learned in Chap. 4.

Formally, the features $f(\mathbf{x}, I)$ we consider in our method are of the form

$$f(\mathbf{x}, I) = [(\mathbf{f}_1 * I)(\mathbf{x}), \ldots, (\mathbf{f}_J * I)(\mathbf{x})]^\top , \tag{3.10}$$

where the $\mathbf{f}_j$'s are convolutional filters learned with an unsupervised learning algorithm, as in [162]. This method is described in Sec. 4.2. In the case of 2D images, in all our experiments, we used $J = 121$ filters. In the case of 3D volumes, the number of possible orientations of the tubular structures is significantly larger and therefore more filters should be used. We found it most effective to learn first a filter bank of $J = 121$ filters and then extend it by rotating the learned filters at different orientations, 14 in practice.

To speed up the convolutions required to compute the descriptor, we approximate the $\mathbf{f}_j$'s by decomposing them as a linear combination of separable filters. This approach is described in detail Sec. 4.3, it involves learning a set of filters $\{\mathbf{s}^k\}_{k=1}^K$, with $K < J$, such that:

$$\mathbf{f}_j = \sum_{k=1}^{K} w_j^k \mathbf{s}^k . \tag{3.11}$$

In this way, the convolutions $(\mathbf{f}_j * I)(\mathbf{x})$ can be written as

$$(\mathbf{f}_j * I)(\mathbf{x}) = \sum_{k=1}^{K} w_j^k (\mathbf{s}^k * I)(\mathbf{x}) , \tag{3.12}$$

where the convolutions $\mathbf{s}^k * I$ are now computed with separable filters. Moreover, the filters $\mathbf{s}^k$ are shared among all the non-separable ones and only the coefficients w_j^k's depend on the specific filter $\mathbf{f}_j$. As we will show in Chap. 4, this greatly reduce the computational cost, compared to Eq. (3.10), even compared to using the Fast Fourier Transform to compute the convolutions. In our experiments we use $K = 49$ in the 2D case and $K = 80$ in the 3D case.

In the 2D case, thanks to the separable filters approximation we can divide the computational time by about half. However, it is in the 3D case that the computational reduction is fundamental. In fact, using $J = 121 \times 14$ filters for feature extraction would be impractical without considering separable filters. For example, let us consider the time needed to compute the convolution of an image stack of size $768 \times 1436 \times 77$, such as those used in our experiments of Sec. 3.6, with a filter of size $21 \times 21 \times 21$. This convolution requires about 231.6 seconds when computed in the spatial domain[3]. Multiplying this time by the number of filters J, we see that the total time for computing the features would be too large. When using the Fast Fourier Transform (FFT) for computing the convolutions, the time drops to 9.6 seconds. However, by using our separable filter approximation scheme, the time

[3]The times are esimated using a multi-thread MATLAB implementation.

can be further reduced to 3.4 seconds for each separable convolution. Considering the linear combination we have a total of 4.5 seconds per feature map. Moreover, computing the convolution using FFT requires additional memory usage, since the FFT is computed using the size of the input image stack. By contrast, in Sec. 3.5 we describe how we can take advantage of the characteristics of the GradientBoost algorithm and of our separable approximation scheme to avoid explicitly computing all the J features at once, thus avoiding memory issues.

More details about the filter learning schemes used in our experiments are given in Chap. 4. In the next section, we discuss in details the computation of multiscale features.

3.4.1 Multiscale Features

In the previous section, we have described how the feature vector is computed by convolutions with a bank of filters learned form training images. The features are used as input by the regressors of Sec. 3.3 to approximate the distance function of the centerlines.

As described in Sec. 3.3, we train a different regressor for every scale in a set $\{r_i\}_{i=1}^{R}$. If we suppose that the images used in Eq. (4.1) to learn the filters contain linear structures at every scale r_i, then, the learned filters will be expressive enough to discriminate the linear structures we want to detect and the corresponding scales.

For this reason, we do not learn a separate filter bank for each regressor φ_{r_i}, but instead we use always the same filter bank for every r_i.

Once the regressors at scale r_i have been trained, for every i, we can approximate the response of the regressors at a generic scale r by using scale-space theory. More precisely, given a scale r_{i_0} for which a regressor $\varphi_{r_{i_0}}$ is known, the response of the regressor φ_r at pixel $\mathbf{x}$ in image I, can be approximated by

$$\varphi_r(\mathbf{x}) \approx \varphi_{r_{i_0}}(f(\mathbf{x}, \sigma_{\frac{r_{i_0}}{r}}(I))), \tag{3.13}$$

where $\sigma_{\frac{r_{i_0}}{r}}(I)$ is image I rescaled by the factor r_{i_0}/r.

In practice, for a given r, we chose the corresponding r_0 to be the closest scale to r in $\{r_i\}_i$, that is

$$r_{i_0} = \arg\min_{i=1,\dots,R} |r - r_i|. \tag{3.14}$$

In this way we limit the artifacts due to re-sampling and to the fact that the linear structure appearance, for large difference of scales, is not scale invariant.

Using Eq. (3.13) is efficient when $r_{i_0} < r$, that is when image I is down-sampled. In the opposite case, convolving an up-sampled image can become excessively costly, in particular for large 3-dimensional volumes.

To avoid this situation, we notice that, since the features $f(\mathbf{x}, I)$ are given by convolutions with linear filters $\mathbf{f}_j$, computing the convolution with the scaled image is equivalent, up to a scaling factor, to computing the convolution between the original image and the rescaled filter:

$$\left(\mathbf{f}_j * \sigma_{\frac{r_{i_0}}{r}}(I)\right)(\mathbf{x}) \approx \gamma_{r_{i_0},r}\left(\sigma_{\frac{r}{r_{i_0}}}(\mathbf{f}_j) * I\right)(\mathbf{x}). \tag{3.15}$$

The constant $\gamma_{r_{i_0},r}$ is equal to the ratio between the area of the original filter and the area of the rescaled filter. More precisely, if $\mathbf{f}_j$ is a n-dimensional filter with size $d_1 \times \ldots \times d_n$ and $\sigma_{\frac{r}{r_{i_0}}}(\mathbf{f}_j)$ is the rescaled filter with size $d_1\frac{r}{r_{i_0}} \times \ldots \times d_n\frac{r}{r_{i_0}}$, we have

$$\gamma_{r_{i_0},r} = \frac{d_1 \cdot \ldots \cdot d_n}{d_1\frac{r}{r_{i_0}} \cdot \ldots \cdot d_n\frac{r}{r_{i_0}}} = \left(\frac{r_{i_0}}{r}\right)^n. \tag{3.16}$$

Thanks to this remark, we can efficiently compute the features also for scales $r < r_{i_0}$.

3.5 Implementation Details

In this Section we give some implementation details about the code and the algorithms used in our experiments. The code was mainly implemented in MATLAB and partly in C++ and it is available at the following webpage: `http://cvlab.epfl.ch/software/centerline-detection`.

Computation of the Distance Transform In order to train our regression-based approach we need to compute the Euclidean distance transform of a binary $(n+1)$-dimensional image, as indicated in Eq. (3.9). For this computation we use the algorithm proposed in [138], which is designed to compute the exact Euclidean distance transform for binary images of arbitrary dimensions.

The algorithm runs in linear time and we used the implementation given in the ITK filter `itkSignedMaurerDistanceMapImageFilter`. This function also allows the user to specify the spacing of the image, which is particularly useful for 3D biomedical data, where the spacing along the z-axis is usually different than the spacing along the xy dimensions. Moreover, we also used this option when computing the scale-space distance transform of Eq. (3.9) in order to set the parameter w. In fact, weighting the radial component of the multiscale distance transform by a factor w is equivalent to using a spacing equal to $\sqrt{w}$

along the radial dimension of the scale-space binary ground truth.

Since, we are only interested in the thresholded distance transform, its computation could be optimized even further. However, we did not find necessary to do this in our case since the ITK function was fast enough and the distance transform needed to be computed only once for each train image.

Instead, in order to save memory usage, we used a sparse representation of the ground truth images. In fact, since linear structures occupy only a small fraction of the pixels (or voxels) in the image, after computing the thresholded distance transform, most of the pixels (or voxels) in the ground truth are zeros.

Sampling For classification problems where the probability of appearance of one class is much lower than the other classes, it is common to train the classification algorithm by sampling the same amount of training data for every class and then re-weight the output of the final classifier to take into account the bias introduced by the sampling.

This problem is particularly relevant for the centerline classification problem, where positive samples are really sparse in the image or in the 3D volume, compared to negative ones.

In the case of a generic regression problem, there is no notion of positive and negative samples. However, in our case, we can divide the samples in two classes: those that lie completely on the background, far from any centerline point, and those that are close to a centerline point. For the first kind of points, the value of our regression ground truth d_C is zero, while for the second kind, is greater than zero.

Therefore, similarly to what is done for binary classification approaches, we found it effective to train our regressors by sampling half of the training samples form location close to the centerlines and half from locations far from centerline points.

Moreover, in order to train a multiscale regressor at scale r_i, we also consider training samples with scales r such that $|r - r_i| < tol$, where tol depends on the set of training scales. In practice, we rescale samples corresponding to scale r by a factor r_i/r, in order to map them to the scale r_i, we want to train the regressor for. In this way, we can exploit more training data for every scale. Similarly, to further augment the training, we randomly rotate the training samples lying close to a centerline.

Gradient Boost In order to train the GradientBoost regressors of Sec. 3.2 efficiently, we consider the following techniques. First, at every boosting iteration only half of the training samples are used to learn the regression tree at that iteration. In addition, we also randomly sample the features used to learn a split in the tree, by using always 500 features at most. This approach is known as Stochastic GradientBoost [70].

Moreover, to make learning more robust, we use the shrinkage regularization technique, which is known to improve the model generalization ability [82]. It amounts at multiplying the weights α_t at Step 7 of Alg. 1 by a constant $\eta < 1$. We use shrinkage factor equal to 0.1 in all our experiments.

The GradientBoost implementation we used in our experiments is the one of `https://sites.google.com/site/carlosbecker/resources`.

Computing Features The convolutional features are learned separately for every dataset used in the experiments. Then, each filter bank is approximated with separable filters as described in Sec. 3.4. We used the code available at the webpage `http://cvlab.epfl.ch/software/filter-learning`.

In the case of 3D datasets, pre-computing all the non-separable features at once, would require too much memory. However, thanks to the Stochastic GradientBoost formulation we do not need to explicitly compute all of them. In our implementation, we only pre-compute the convolution with the separable filters. Then, at each boosting iteration, we linearly combine them to produce the random subset of features considered at that iteration. Although this implies that some features are recomputed several times, it makes it possible to explore a much larger pool of features, without having memory issues.

Non-Maximum Suppression Applying our method to an n-dimensional image, yields an $(n+1)$-dimensional one, with n spatial dimensions and one scale dimension. Our method is designed to respond maximally at the centerlines in scale-space. To find these local maxima, we first compute a n-dimensional image by keeping for each location the maximum along the radii, and saving the radius corresponding to the maximum. We then perform a Canny-like Non-Maximum Suppression by keeping only the locations that correspond to a local maximum along a line perpendicular to the local orientation, and within a neighborhood of width defined by the radius. We estimate the orientation using the eigenvectors of the OOF matrix [113], which we found more robust than the Hessian. Results from this Non-Maximum Suppression step are shown in Fig. 3.5.

3.6 Experimental Evaluation

In this section, we first introduce the datasets and the parameters used to test our centerline detection method. Then, we describe our evaluation methodology and we discuss our results.

3.6.1 Datasets and Parameters

Our method depends on few parameters, namely: the radial weight w in Eq (3.9); the size s of the filters used to extract the features; the range of sampled scales and the number of trained regressors.

The range of scales sampled for the different datasets is automatically determined from the ground truth data and was always sampled uniformly. We optimized the other parameters by a cross validation procedure on small volumes. We found experimentally that the radial weight w should be larger for images containing thin structures and smaller for larger scales. We tested our method on the 2D road images and 3D biological image stacks depicted by Fig. 3.5. More specifically we used the following datasets:

- **Aerial**: Aerial images of road networks. We used a training set composed of 7 images and used 7 others for testing. We sampled 10 scales ranging from 5 to 14. We trained 4 regressors at scales 6, 8, 11 and 13 and learned filters of size $s = 21$. We set w to 1.
- **Brightfield**: A dataset of 3D image stacks acquired by brightfield microscopy from biocityne-dyed rat brains. We used 3 images for training and 2 for testing. We sample 12 scales corresponding to radii from 1 to 12 microns. We trained 2 regressors at scales 2 and 8. We learned filters of size $s = 21$ and used $w = 1$.
- **VC6**: Three dimensional brightfield micrographs of biocytin-labeled cat primary visual cortex layer 6 taken from the DIADEM challenge data [13]. We used 3 images for training and 2 for testing. We sampled 6 scales from 1 to 6, trained 3 regressors at scales 1, 3, and 5. We used $w = 7$ and $s = 11$.
- **Vivo2P**: Three dimensional *in vivo* two-photon images of a rat brain, capturing the evolution of neurons in the neocortex. We used 2 images for training and 3 sequences of 3 images for testing. We sampled 3 scales, 0.6, 0.7, and 0.8 microns. We trained one regressor at scale 0.7, using $w = 1$ and $s = 21$.

The Aerial dataset is challenging because of the similar aspects of roads and other background objects and because of the occlusions of the roads due to cars or trees. The

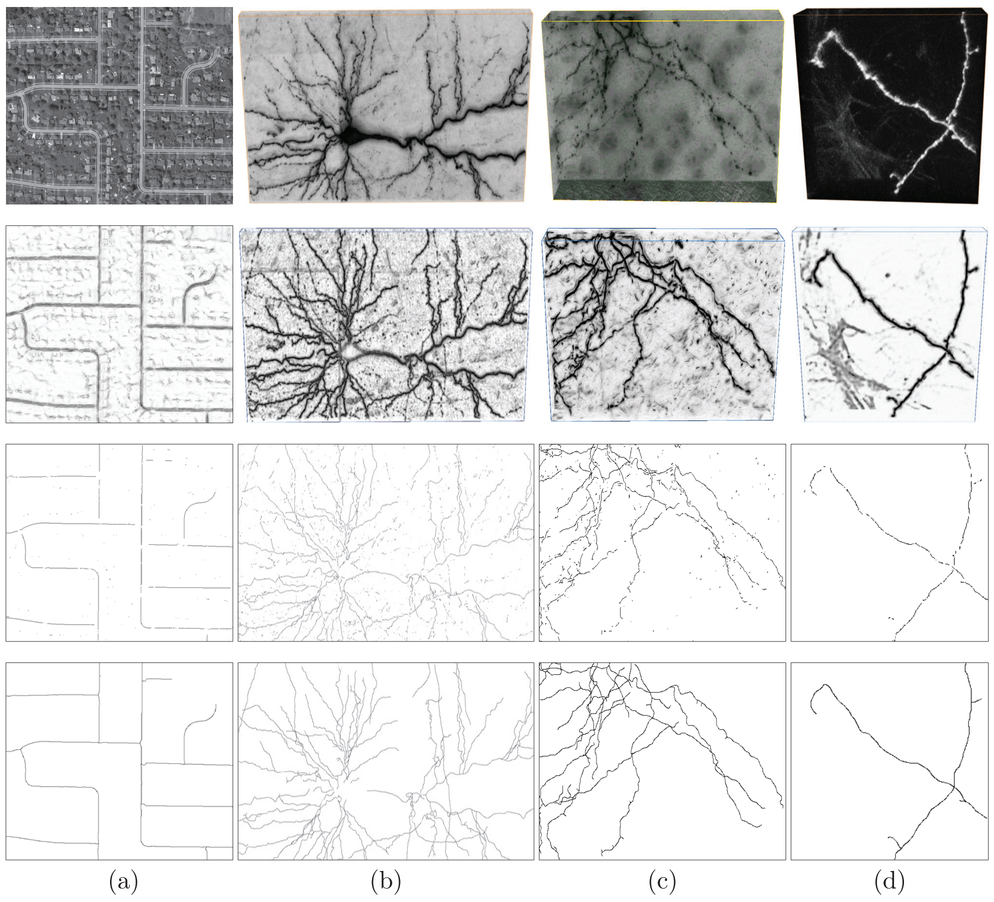


Figure 3.5: Centerline Detection Results. (a) Aerial image. (b) Brightfield image stack. (c) VC6 image stack. (d) Vivo2P volume. In each case, we show from top to bottom the original image, the maximum projection along the radial component of our regressor's output, centerlines detected by thresholding after Non-Maximum Suppression, and ground truth centerlines. In the last three rows, values have been inverted for visualization purposes

Brightfield and VC6 datasets main difficulties are due to the irregular shape of the dendrites and the structured noise caused by the staining process. The Vivo2P dataset instead shows a low signal-to-noise ratio and presents some very faint linear structures that were not annotated by the human experts.

For training, we randomly sampled 100 000 image locations within the distance d_M to the centerline and other 100 000 from points further than d_M to the centerline. During the boosting iterations half of the samples were randomly used to learn the weak learner.

For the Aerial dataset the average size of the test images is $5.87 \cdot 10^5$ pixels with standard deviation $2.28 \cdot 10^5$. For the Brightfield dataset the images are composed of $(8.58 \pm 0.12) \cdot 10^7$ voxels, for the VC6 dataset of $(2.65 \pm 1.52) \cdot 10^7$ and for Vivo2P of $(2.41 \pm 0.87) \cdot 10^6$ voxels. The running time in our MATLAB implementation is of several hours for training and from few minutes to few hours for testing.

3.6.2 Evaluation

We compare our approach against three of the most powerful model-based methods for centerline detection. Optimally Oriented Flux (OOF) [113], Oriented Flux with Oriented Flux Antisymmetry [114] (OOF+OFA), and Multidirectional Oriented Flux [194] (MDOF). Moreover, to prove the importance of our regression approach compared to Classification, we also train a GradientBoost classifier to segment the centerlines from the rest of the images, thus emulating the approach of [23]. We use the same features and the same parameters used for regression. The only difference is that for Classification training is done using the binary ground truth and the exponential loss.

For evaluation we consider Precision-Recall (PR) curves analysis [136]. As usually done to evaluate methods extracting one-pixel-wide curves [136, 145, 194], we introduce a tolerance factor ρ to compute the curves. More precisely, a predicted centerline point is considered a true positive if it is at most ρ distant from a ground truth centerline point. We generate PR curves for all the methods for different value of ρ. The results are shown in Fig. 3.8 and show that our approach clearly outperforms the others on all datasets.

We also evaluate the accuracy of the radii we estimate. Again we follow the same evaluation methodology of [194]. We start by thresholding the image after Non-Maximum Suppression at different values. Then, for each point in the thresholded image, we construct a sphere using the corresponding estimated radius. In this way we obtain for every threshold value a full segmentation of the tubular structures, which we can compare to the ground truth. Fig. 3.7 shows some examples of segmentations obtained in this way.

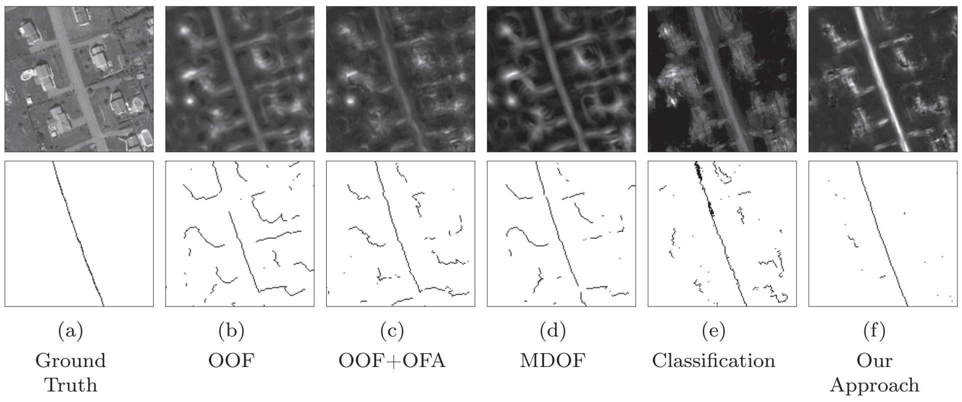


Figure 3.6: Centerline detection on Aerial images. **Top row**: Raw image and responses returned by the different methods. **Bottom row**: Ground truth and extracted centerlines. In the bottom row, values have been inverted for visualization purposes.

Since the ground truth data itself can be inaccurate, we introduce also in this case a tolerance factor δ, and eliminate from comparison points that are closer than $\delta\, r$ from the surface of a ground truth tube of radius r.

Fig. 3.9 shows the Precision-Recall results for different values of δ. In this case also, our method outperforms all the others for all the relevant ranges of precision and recall. Qualitative results are shown in Fig. 3.5 and Fig. 3.7.

We observe the biggest improvement for the Aerial dataset. There, model-based methods do worst because they respond strongly to bright polygonal objects such as houses, as can be seen in Fig. 3.6. Learning-based methods can be taught to discount them, and in this case, Classification does better than hand-crafted methods, but still not as well as our approach.

On the Brightfield and VC6 datasets, our approach still does best, but Classification does worst, especially in Brightfield case, due to the presence of very wide branches. As shown in Fig. 3.1, in such cases, the maximum response is not necessarily on the centerline and Non-Maximum Suppression behaves badly. Our regression-based approach avoids this problem. As observed in [194], the antisymmetric term introduced by OFA degrades the results with respect to OOF for very irregular structures. However, with and without it, OOF is more sensitive than our algorithm to strong artifacts and image noise, which are hard to ignore for hand-crafted methods. Only for the segmentation results on the Brightfield dataset and for very high recall values, does the precision of our approach significantly degrade. This is due to the sensitivity of our method to thin and faint structures. It is needed to detect the smallest branches but, inevitably, makes it also respond, albeit weakly, to noise. Moreover, in this range of recall values, the centerline localization accuracy of the other methods becomes very low, making their results essentially meaningless.

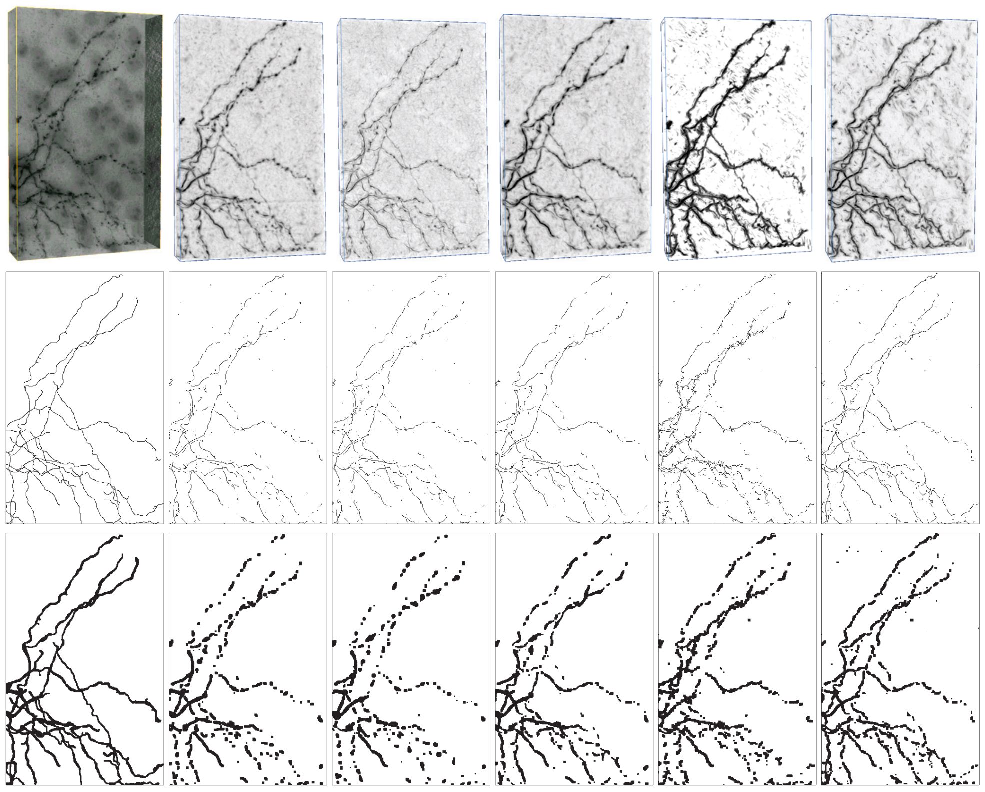
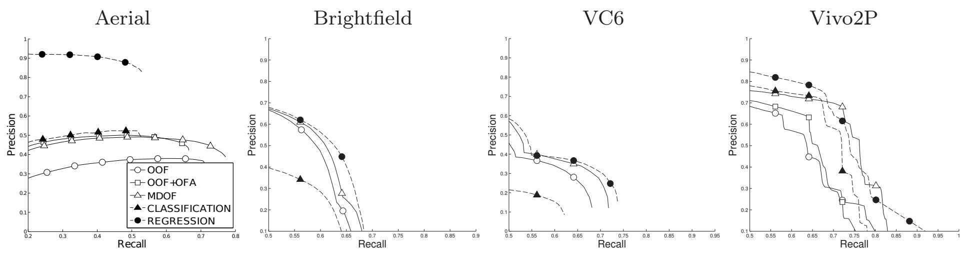
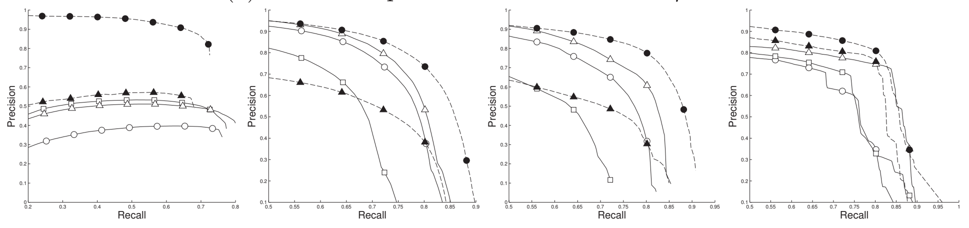


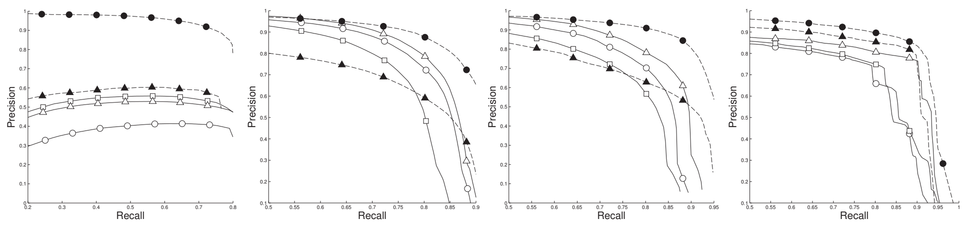
Figure 3.7: Centerline detection and Segmentation on a test image of the VC6 dataset for the different methods. **First row**: Original image and maximum projection along the radial dimension of the multiscale score map; **Second row**: Ground truth centerlines and centerlines detected by applying Non-Maximum Suppression to the score maps; **Third row**: Maximum intensity projection along the z-axis of the segmentation ground truth and the segmentations obtained with the different methods. The results and ground truth images have been inverted for visualization purposes.



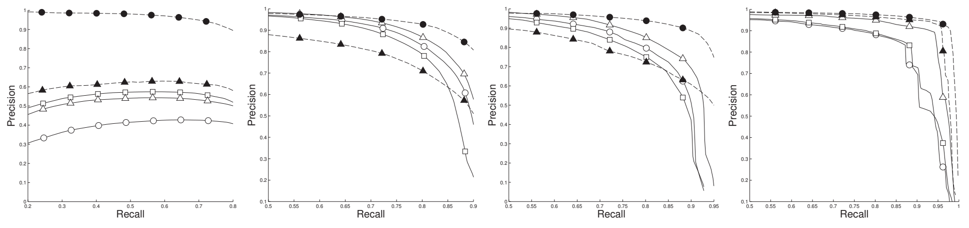
(a) Centerline precision-recall curves for $\rho = 1$.



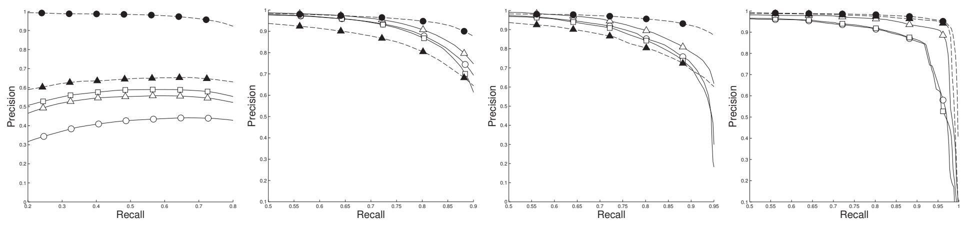
(b) Centerline precision-recall curves for $\rho = 2.0$.



(c) Centerline precision-recall curves for $\rho = 3$.

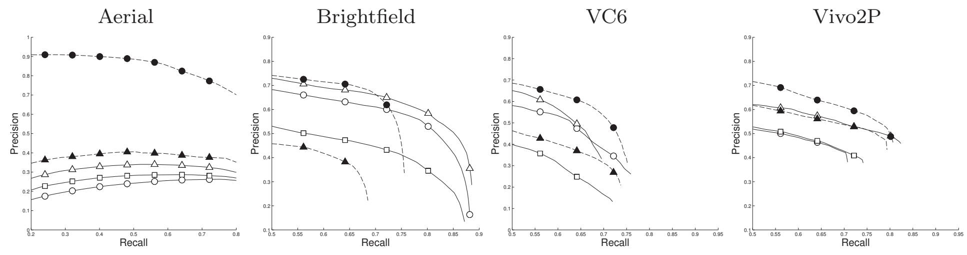


(d) Centerline precision-recall curves for $\rho = 4$.

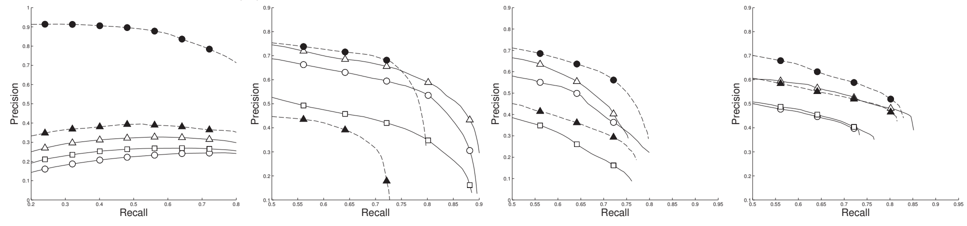


(d) Centerline precision-recall curves for $\rho = 5$.

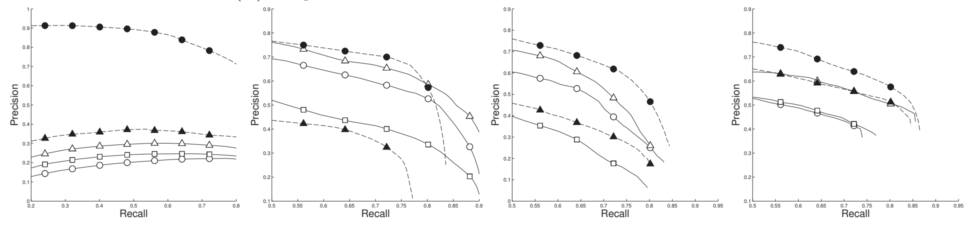
Figure 3.8: Precision-Recall curves of for centerline detection for different tolerance values. Our method outperforms the others on all the datasets we considered.



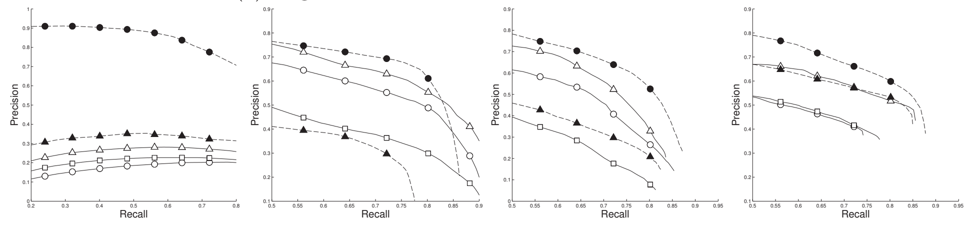
(a) Segmentation Precision-Recall for $\delta = 0.1$.



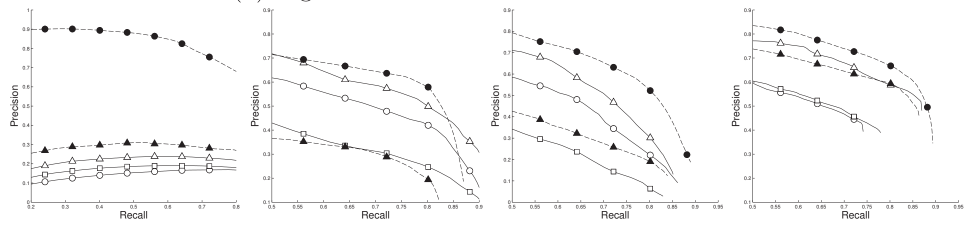
(b) Segmentation Precision-Recall for $\delta = 0.2$.



(c) Segmentation Precision-Recall for $\delta = 0.3$.



(d) Segmentation Precision-Recall for $\delta = 0.4$.



(e) Segmentation Precision-Recall for $\delta = 0.5$.

Figure 3.9: Precision-Recall curves of for segmentation for different tolerance values. Our method outperforms the others on all the datasets we considered. Legend in Fig. 3.8(a).

Conclusion

In this chapter we have introduced an efficient regression-based approach to centerline detection, which we showed to outperform both methods based on hand-designed filters and classification-based approaches.

Our approach is very general and applicable to other linear structure detection tasks when training data is available. For example, given a training set of natural images and the contours of the objects present in the images, our framework is able to learn to detect such contours in new images as was done in [10]. We consider this application in Chap. 5.

However, the approach presented in this chapter only relies on local features. Therefore, it can not solve ambiguous situations that need information beyond the size of the local filters in order to be correctly discriminated. This is particularly relevant for applications such as detecting object boundaries in natural images, or for detecting linear structures with severe occlusions.

In Chap. 5 we will extend our method and make it context-aware. By including image contextual information, we can further improve the results presented in this chapter and also obtain more accurate results for boundary detection, compared to previous work. Moreover, we show the advantage of using our method also when used in conjunction with a tracing algorithm.

Before introducing our context-aware method, in Chap. 4 we will describe in detail the filter learning scheme used in our feature extraction step of Sec. 3.4. In particular, we will introduce a separable filter learning approach that can be used to speed up the computation of convolutions with filter banks. We will prove the effectiveness of this approach by using it on different Computer Vision tasks and prove its superiority to previous methods.

CHAPTER 4

Learning Separable Filters

In the previous chapter, we have described a regression-based approach for centerline detection. Following the taxonomy of Chap. 2, this method is composed of a feature extraction step and a scoring step. The scoring step was described in detail in Chap. 3, and it is based on the idea of learning a set of regressors in order to approximate a multiscale distance transform. Thanks to this formalism, we obtain more accurate results compared to hand-crafted and classification-based methods.

In this chapter, we focus on the feature extraction step. As mentioned in Sec. 3.4, we rely on an unsupervised learning scheme [162] in order to automatically learn a set of filters from training data. Then, we obtain the features used as input to the regressor by convolving the filters with an input image.

Since computing convolutions between a large number of filters and an n-dimensional image can become computationally prohibitive, we will introduce in this chapter an approximation scheme based on separable filters. We show that, by expressing a full rank filter bank as linear combination of separable ones, we can reduce computational time considerably, with no significant loss in performance.

This idea was first introduced in [163], where the separable filters are learned by minimizing an objective function that includes low-rank constraints. This approach delivers the desired run-time accuracy and efficiency but at a high-computational cost during the learning phase. Moreover, the low-rank constraints, being soft constraints, do not guarantee that the resulting filters are of rank one. In this situation, separability can be imposed after the optimization by truncating the singular values of the filters. Here, we propose a new approach based on Tensor Decomposition. Our experiments show that this second approach converges faster during learning and it is more accurate.

The rest of the chapter is organized as follows. In Sec. 4.1, we review previous work on separable filter learning. In Sec. 4.2, we briefly introduce the convolutional filter learning approach used to derive the filter banks we used for feature extraction in Chap. 3. In Sec. 4.3, we describe how a generic filter bank can be accurately approximated by a linear combination of separable ones. To show the advantage of the separable approximation, in Sec. 4.4 we discuss computational complexity and in Sec. 4.5 we apply it to two different Computer Vision tasks, to show its generality and efficiency. We conclude in Sec. 4.5.3, by comparing our Tensor Decomposition approach with the one of [163].

Part of the content of this chapter has been previously published in peer-reviewer conferences and journals [163, 181].

4.1 Related Work on Separable Filter Learning

Automatic feature learning has long been an important area in Machine Learning and Computer Vision. Many techniques have been used to learn features in either supervised or unsupervised ways [116, 88, 24, 27]. Sparse dictionary learning techniques has emerged as a powerful tool for object recognition [40, 104, 209] and image denoising [58, 128], among others.

However, for most such approaches, run-time feature extraction can be very time-consuming because it involves convolving the image with many non-separable non-sparse filters. It was proposed several years ago to split convolution operations into convergent sums of matrix-valued stages [190]. This principle was exploited in [153] to avoid coarse discretization of the scale and orientation spaces, yielding steerable separable 2D edge-detection kernels. This approach is powerful but restricted to kernels that are decomposable in the suggested manner, which precludes the potentially arbitrary ones that can be found in a learned dictionary or a hand-crafted ones to suit particular needs.

After more than a decade in which the separability property has been either taken for granted or neglected, there is evidence of renewed interest [130, 156]. The scope of these papers is, however, limited in that they are restricted to specific frameworks, while our approach is completely generic, since it can be applied to any filter bank. Nonetheless, they prove a growing need for fast feature extraction methods.

Among other recent feature-learning publications, very few have revisited the problem of run-time computational efficiency. The majority of those advocate exploiting the parallel capabilities of modern hardware [60, 144].

An interesting recent attempt at reducing computational complexity is [167], which involves learning a filter bank by composing a few atoms from an handcrafted separable dictionary. Our approach is in the same spirit, but it is more general as we also learn the atoms as well. As shown in the results section, this results in a smaller number of separable filters that are tuned for the task at hand.

In [83], separable dictionaries are learned through a classical sparse coding approach, not in a convolution-based one. The authors show that by using separable items as compared to unstructured ones, it is possible to deal with larger images. However, the dimensions of the images used as input to their method are smaller than those typically handled by convolutional sparse coding approaches. Moreover, it has been shown in [163] that directly learning separable filters yields worse results than those of their unstructured counterpart.

Here, we overcome this limitation by introducing separability at a later stage of the learning process. We first learn a set of non-separable filters and then approximate them as linear combinations of a small set of separable ones, which are specific for the particular application.

Finally, the authors of [30] propose a way to reduce the time it takes to learn non-separable filters. This makes their approach complementary to ours, as their method can be incorporated into our pipeline when we learn the set of non-separable filters, as described in the next section.

4.2 Learning Centerline Features

Most dictionary learning algorithms operate on image patches [150, 128, 40], but convolutional approaches [104, 118, 213, 161] have been introduced as a more natural way to process arbitrarily-sized images. In our work, we consider the convolutional extension of Olshausen and Field's objective function proposed in [161].

Formally, J filters $\{\mathbf{f}_j\}_{1 \leq j \leq J}$ are computed as

$$\operatorname*{arg\,min}_{\{\mathbf{f}_j\},\{\mathbf{m}_i^j\}} \sum_i \left(\left\| I_i - \sum_{j=1}^{J} \mathbf{f}_j * \mathbf{m}_i^j \right\|_2^2 + \lambda_1 \sum_{j=1}^{J} \left\| \mathbf{m}_i^j \right\|_1 \right) , \tag{4.1}$$

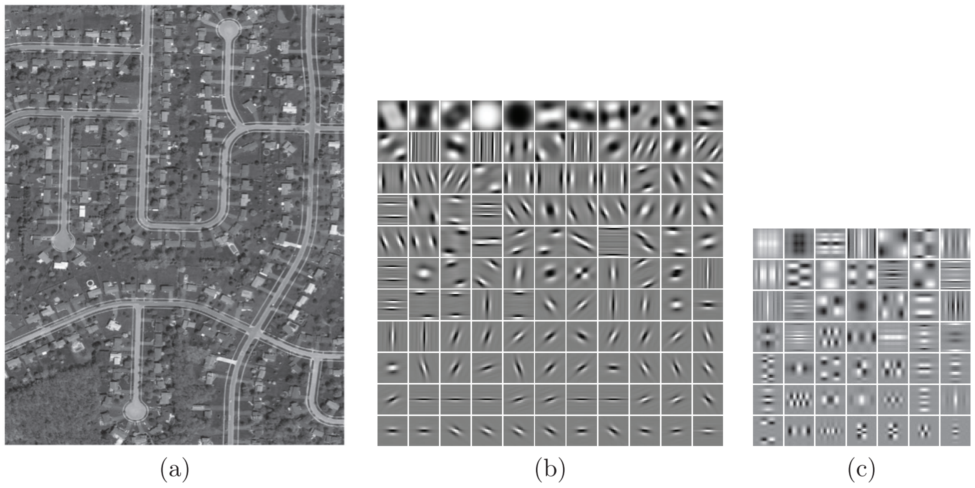


Figure 4.1: Filter banks learned on the Aerial Dataset of Sec. 3.6.1. (a) Example image from the dataset. (b) Full-rank filters learned using the convolutional sparse coding approach of Eq. (4.1). (c) Separable filters approximating the filter bank in (b), learned by solving Eq. (4.6).

where, I_i is an input image, $\{\mathbf{m}_i^j\}_{1\leq j\leq J}$ is the set of feature maps extracted during learning and λ_1 is a regularization parameter. The ℓ_1 norm $\|\cdot\|_1$ is used to impose sparsity on the entries of the feature maps.

A standard way to solve Eq. (4.1) is to alternatively optimize over the $\mathbf{m}_i^j$ representations and the $\mathbf{f}_j$ filters. Stochastic gradient descent is used for the latter, while the former is achieved by first taking a step in the direction opposite to the ℓ_2-penalized term gradient and then applying the soft-thresholding operation on the $\mathbf{m}_i^j$s. Soft-thresholding is the proximal operator for the ℓ_1 penalty term [14]; its expression is $\text{prox}_\lambda(x) = \text{sgn}(x)\max(|x| - \lambda, 0)$. Proximal operators allow to extend gradient descent techniques to some non-smooth problems.

Fig. 4.1(b) shows filters learned on a set of Aerial images. As we can see, some of the the filters are well localized in the frequency domain, responding to oriented features at a particular scale. While other filters are more specialized for the particular dataset, like for example those composed of two parallel lines, which are tuned to have a strong response when convolved with the the bright road lines.

The filters learned by solving Eq. (4.1) are used to extract the features used in Chap. 3, as indicated in Eq. (3.10). To speed up the convolutions required to compute this descriptor, we rely on an approximation technique, which decomposes the filters $\{\mathbf{f}_j\}_{j=1}^J$ as linear combinations of a set of separable filters. This approach is described in the following section.

4.3 Learning Separable Filters

While the convolutional formulation of Eq. (4.1) achieves state-of-the-art results [164], the required run-time convolutions are costly because the resulting filters are not separable. Quantitatively, if $I \in \mathbb{R}^N$ and $\mathbf{f}_j \in \mathbb{R}^{d_1 \times d_2}$, extracting the feature maps requires $\mathcal{O}\left(N \cdot d_1 \cdot d_2\right)$ multiplications. By contrast, if the filters were separable, the computational cost would drop to a more manageable $\mathcal{O}\left(N \cdot (d_1 + d_2)\right)$. This cost reduction becomes even more desirable in biomedical applications that require processing large 3D image stacks.

We therefore rely on an approximation scheme which makes our filters separable without compromising their descriptive power. This approach takes advantage of the fact that arbitrary filters of rank K can be expressed as linear combinations of K separable filters [153]. The solution we propose is general as it can be applied to any filter bank, and not only to filters learned with Eq. (4.1).

We can formulate our problem as finding a decomposition of the filters $\mathbf{f}_j$'s of the form

$$\mathbf{f}_j = \sum_{k=1}^{K} w_j^k \mathbf{s}^k \, , \tag{4.2}$$

where the filter $\mathbf{s}^k$'s are separable. The filters $\mathbf{s}^k$'s are shared among all the non-separable ones and only the coefficients w_j^k's depend on j. In this way, convolving the image with all the $\mathbf{f}_j$'s at run-time amounts to convolving it with the separable $\mathbf{s}^k$ filters and then linearly combining the results, without any further convolutions.

Separable Filters by Minimizing the Nuclear Norm

In [163] it was first proposed to obtain such decomposition by solving the optimization problem

$$\operatorname*{arg\,min}_{\{\mathbf{s}^k\},\{w_k^j\}} \sum_j \left\| \mathbf{f}_j - \sum_{k=1}^{K} w_j^k \mathbf{s}^k \right\|_2^2 + \lambda_* \sum_{k=1}^{K} \left\| \mathbf{s}^k \right\|_* \, , \tag{4.3}$$

where, the first term in the equation ensures that the separable approximation is close to the original filters, while the second term enforces the filters $\mathbf{s}^k$'s to have low rank. In the 2D case, $\| \cdot \|_*$ is the nuclear norm, which is the sum of its singular values and is a convex relaxation of the rank [61]. Computing it involves a Singular Value Decomposition (SVD) of the filters $\mathbf{s}^k$'s.

In the n-dimensional case, the optimization is similar to the 2D case, but instead of the SVD, the Canonical Polyadic Decomposition (CPD) [106] of the n-dimensional filter

$\mathbf{s}^k$ is considered. We define the CPD in more detail below when we introduce a Tensor Decomposition approach for learning separable filters.

The method of Eq. (4.3) produces a small set of separable filters that approximate the original ones. However, it requires introducing an additional regularization parameter λ_* that can be difficult to tune. Moreover, its convergence rate is slow, especially when trying to approximate high-rank filters. Finally, it does not guarantee that the filters $\mathbf{s}^k$'s have rank one at the end of the optimization and often hard thresholding needs to be applied on the singular values of the $\mathbf{s}^k$'s filters after convergence.

For these reasons, we introduce an alternative approach to finding the separable filters $\mathbf{s}^k$'s and weights w_j^k's of Eq. (4.2), which relies on Tensor Decomposition. In Sec. 4.5.3, we show that this method is easier to optimize, it is more accurate and has less parameters.

Separable Filters by Tensor Decomposition

Low rank Tensor Decomposition techniques have been used in many Computer Vision applications [105], [21], [20], [173] to obtain speed up for various applications. In this section, we show how Tensor Decomposition can be used in a general framework to obtain the decomposition of Eq. (4.2) for an arbitrary filter bank, thus speeding up convolutions.

We first describe the method for 2D filters, then we show how to generalize it to filters of arbitrary dimension. We start by stacking the J filters $\mathbf{f}_j \in \mathbb{R}^{d_1 \times d_2}$ into a 3D tensor $\mathcal{F} \in \mathbb{R}^{d_1 \times d_2 \times J}$, where the j-th slice of $\mathcal{F}$ corresponds to the filter $\mathbf{f}_j$, as shown in Fig. 4.2.

Writing the slices of $\mathcal{F}$ as linear combinations of rank-one matrices is equivalent to writing the tensor $\mathcal{F}$ as a linear combination of rank-one tensors

$$\mathcal{F} = \sum_{k=1}^{K} \mathbf{a}^k \circ \mathbf{b}^k \circ \mathbf{w}^k \ , \tag{4.4}$$

where $\mathbf{a}^k$ is a vector of length d_1, $\mathbf{b}^k$ a vector of length d_2 and $\mathbf{w}^k$ a vector of length J. The symbol $\circ$ corresponds to the tensor product, that for vectors is also referred to as outer product. Such a decomposition is called Canonical Polyadic Decomposition (CPD) [106] of the tensor $\mathcal{F}$ and the right-hand side of Eq. (4.4) is called Kruskal form of the tensor. We refer to K as the rank of the Kruskal tensor.

If tensor $\mathcal{F}$ can be written in the form of Eq. (4.4), we obtain

$$\mathbf{f}_j = \sum_{k=1}^{K} w_j^k \mathbf{s}^k, \quad \forall j, \tag{4.5}$$

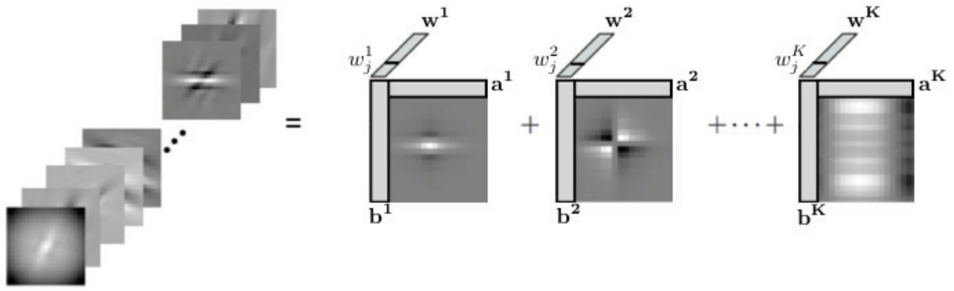


Figure 4.2: Tensor Decomposition for learning separable filters. Left: A bank of two-dimensional filters is stacked together to form a 3-dimensional tensor. Right: The tensor is decomposed in the sum of K rank-one tensors. Thus, the original filters are approximated by the weighted sum of the separable filters $\mathbf{s}^k = \mathbf{a}^k \circ \mathbf{b}^k$.

where the separable filters are given by the $\mathbf{a}^k$ and $\mathbf{b}^k$ components of the CPD, that is, $\mathbf{s}^k = \mathbf{a}^k \circ \mathbf{b}^k$. The coefficients w_j^k's necessary to reconstruct the filter j are given by the j-th component of the $\mathbf{w}^k$'s vectors.

In general, for a given K, we have no guarantee that the decomposition of Eq. (4.4) exists. Thus, we will compute the best approximation of this form by optimizing

$$\min_{\{\mathbf{a}^k,\mathbf{b}^k,\mathbf{w}^k\}_k} \left\| \mathcal{F} - \sum_{k=1}^{K} \mathbf{a}^k \circ \mathbf{b}^k \circ \mathbf{w}^k \right\|_2^2 . \tag{4.6}$$

To this end, we use the CP-OPT algorithm of [1], implemented in the MATLAB tensor toolbox, in which Eq. (4.6) is solved by conjugate gradient descent. Fig. 4.1(c) shows an example of filters learned with this approach.

The rank K of the decomposition is the only parameter of the method. It determines the number of separable filters used to approximate the original filter bank.

Similarly, for the n-dimensional case, let $\{\mathbf{f}_j\}_{j=1}^{J}$ be a set of filters, with $\mathbf{f}_j \in \mathbb{R}^{d_1\times\cdots\times d_n}$ $\forall j$. Let $\mathcal{F}$ be the $(n+1)$-dimensional tensor formed by stacking the $\mathbf{f}_j$'s along the $(n+1)$-th dimension, that is $\mathcal{F}_{i_1,i_2,\ldots,i_n,j} = (\mathbf{f}_j)_{i_1,i_2,\ldots,i_n}$. Applying CPD of rank K to $\mathcal{F}$, yields

$$\mathcal{F} \approx \sum_{k=1}^{K} \mathbf{a}^{k,1} \circ \mathbf{a}^{k,2} \circ \cdots \mathbf{a}^{k,n} \circ \mathbf{w}^k. \tag{4.7}$$

Therefore, for all $j = 1,\ldots,J$, the separable approximation of $\mathbf{f}_j$ is given by $\mathbf{f}_j \approx \sum_{k=1}^{K} w_j^k \mathbf{s}^k$, with $\mathbf{s}^k = \mathbf{a}^{k,1} \circ \mathbf{a}^{k,2} \circ \cdots \mathbf{a}^{k,n}$.

In Sec. 4.5.3, we show that the Tensor Decomposition approach of Eq. (4.6) has faster convergence and gives a better approximation of the original filter bank, compared to the approach of Eq. (4.3).

4.4 Computational Complexity

In this section, we compare the computational complexity of different methods used to compute convolutions of an image with a filter bank. First, we start by introducing the competing strategies used in the evaluation. Then, we study the case of 2D filters and finally, the generalization to the n-dimensional case.

Competing Strategies

In the following, we will refer to the non-separable filters obtained by minimizing the objective function of Eq. (4.1) as *NON-SEP*. To provide a separable-filters-based baseline, we also compute separable filters by approximating each *NON-SEP* filter by the outer product of its first left singular vector with its first right singular vector, computed using SVD in 2D and by rank-1 CPD for the nD case. This is the simplest way to approximate a non-separable filter by a separable one. We refer to these strategies as *SEP-SVD* and *SEP-CPD* respectively. For completeness sake, we reimplemented *NON-SEP* using the Fast Fourier Transform to perform the convolutions. This approach is known to speed-up convolutions for large enough filters and we will refer to it as *NON-SEP-FFT*.

SEP-COMB and *SEP-TD* denote the separable filters whose linear combinations can be used to approximate the non-separable *NON-SEP* filters as described in Sec. 4.3. More specifically, *SEP-COMB* refer to those that have been learned by minimizing the nuclear norm, as in Eq. (4.3) and *SEP-TD* to those obtained by Tensor Decomposition, by solving Eq. (4.6).

Finally, although the *SEP-COMB* and *SEP-TD* filters can be used to write the non-separable ones as linear combinations of them, explicitly computing the coefficients of these combinations is not always necessary. For example, when the filters output is to be fed to a linear classifier for classification purposes, this classifier can be trained directly on the separable-filters output instead of that of the non-separable ones. This approach, which we will refer to as *SEP-COMB** and *SEP-TD**, further simplifies the run-time computations because the linear combinations coefficients are then learned implicitly at training-time.

The different methods described in the previous section are summarized in Tab. 4.1. Here, we provide an analysis of their computational complexities in terms of the number of multiplications required to perform the necessary run-time convolutions.

Table 4.1: Summary of the different methods used for our experiments, as described at the beginning of Sec. 4.4.

Method Name	**Filter Bank**	**Run Time Computations**
NON-SEP	Non-separable filters learned from Eq. (4.1)	Spatial convolutions
NON-SEP-FFT	Non-separable filters learned from Eq. (4.1)	FFT convolutions
SEP-SVD	Approximation of *NON-SEP* by truncated SVD in 2D	Separable convolutions
SEP-CPD	Approximation of *NON-SEP* by rank-one CPD in 3D	Separable convolutions
SEP-COMB	Separable filters learned from Eq. (4.3)	Separable convolutions + linear combinations
SEP-TD	Separable filters learned from Eq. (4.6)	Separable convolutions + linear combinations
*SEP-COMB**, *SEP-TD**	As *SEP-COMB* and *SEP-TD*	Separable convolutions

The 2D Case

Let $I \in \mathbb{R}^{N_1 \times N_2}$ be an image we want to convolve with a filter bank $\{\mathbf{f}_j\}_{j=1}^{J}$, with $\mathbf{f}_j \in \mathbb{R}^{d_1 \times d_2}$, for all j.

In the *NON-SEP* case, J convolutions are computed in the spatial domain and each one requires $N_1 \cdot N_2 \cdot d_1 \cdot d_2$ multiplications, for a total of

$$\textit{NON-SEP}_{nop} = J \cdot N_1 \cdot N_2 \cdot d_1 \cdot d_2 \tag{4.8}$$

multiplications.

In the *NON-SEP-FFT* case, the convolutions are performed in the frequency domain, which involves the following steps:

- Padding I and $\mathbf{f}_j$ with zeros to have the same size $m_1 \times m_2$, where m_i is the closest power of 2 larger than $(N_i + d_i - 1)$, for $i = 1, 2$;
- Computing real-to-complex FFT on the padded image and the J filters;
- Multiplying the resulting Discrete Fourier Transform of the image by that of each filter;
- Computing complex-to-real Inverse FFT (IFFT) on the results.

To decrease the total computational cost of the convolutions, we can precompute the FFT of the filters, at the cost of using more memory.

Assuming that each FFT and IFFT requires $c \cdot m_1 \cdot m_2 \cdot \log_2(m_1 \cdot m_2)$ complex multiplications, where c depends on the specific FFT algorithm being used, and that each complex multiplication requires 3 real multiplications, this yields a total of

$$\begin{aligned} \textit{NON-SEP-FFT}_{nop} &= 3 \cdot c \cdot m_1 \cdot m_2 \cdot \log_2(m_1 \cdot m_2) \\ &+ 3 \cdot J \cdot m_1 \cdot m_2 \\ &+ 3 \cdot J \cdot c \cdot m_1 \cdot m_2 \log_2(m_1 \cdot m_2) \end{aligned} \tag{4.9}$$

multiplications. In the experiments, the value we used for the constant c is 2.

When the filters are separable, the cost of a spatial convolution reduces to $N_1 \cdot N_2 \cdot (d_1 + d_2)$. If the filter bank is composed of K filters, this represents

$$\text{SEP*}_{nop} = K \cdot (N_1 \cdot N_2 \cdot (d_1 + d_2)) \tag{4.10}$$

multiplications. This is the total cost for *SEP-SVD* and *SEP-CPD*, as well as *SEP-COMB** and *SEP-TD**. In the cases of *SEP-COMB* and *SEP-TD*, one must account for the additional cost of linearly combining the results to approximate the J non-separable filters. This requires $N_1 \cdot N_2 \cdot K \cdot J$ more multiplications, for a total of

$$\text{SEP}_{nop} = K \cdot N_1 \cdot N_2 \cdot (J + d_1 + d_2) \tag{4.11}$$

multiplications.

In Fig. 4.3(a), we plot the values of $\textit{NON-SEP}_{nop}$, $\textit{NON-SEP-FFT}_{nop}$, SEP*_{nop}, and SEP_{nop}, normalized by the number of pixels in the image, as a function of the size $d = d_1 = d_2$ of the filters in the range $[3, 25]$. A 2D test image of size 488×488 is considered and convolved with $J = 121$ non-separable filters and $K = 25$ separable ones. Notice that the size of the image is chosen so that the size considered to compute the FFT is a power of 2 for the maximum value of the filters $d = 25$. In this way the zero-padding required is minimal, which is at the advantage of the FFT based approach.

Note that these theoretical curves are very similar to those observed experimentally, shown in Fig. 4.4. Our code relies on the MATLAB `conv2` function for spatial 2D convolutions and on the fftw library for the frequency domain convolutions. Observe that these functions can be run in parallel to further reduce the cost of the convolutions, as shown in Fig. 4.4.

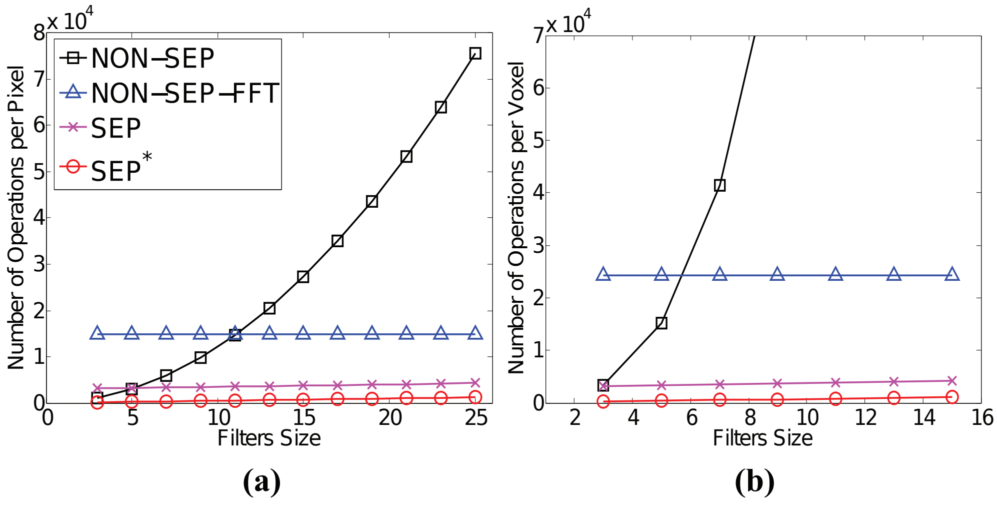


Figure 4.3: Number of operations per pixel to compute convolutions, as a function of the filters size. **(a)** An image of 488×488 pixels is convolved with a filter bank of $J = 121$ non-separable filters and $K = 25$ separable ones. **(b)** A volume of $114 \times 114 \times 50$ voxels is convolved with a filter bank of $J = 121$ non-separable filters and $K = 25$ separable ones. The theoretical values are very similar to the experimental time shown in Fig. 4.4 and they show that the number of operations needed to compute the convolutions with our approach is smaller than both spatial convolution and FFT based convolution.

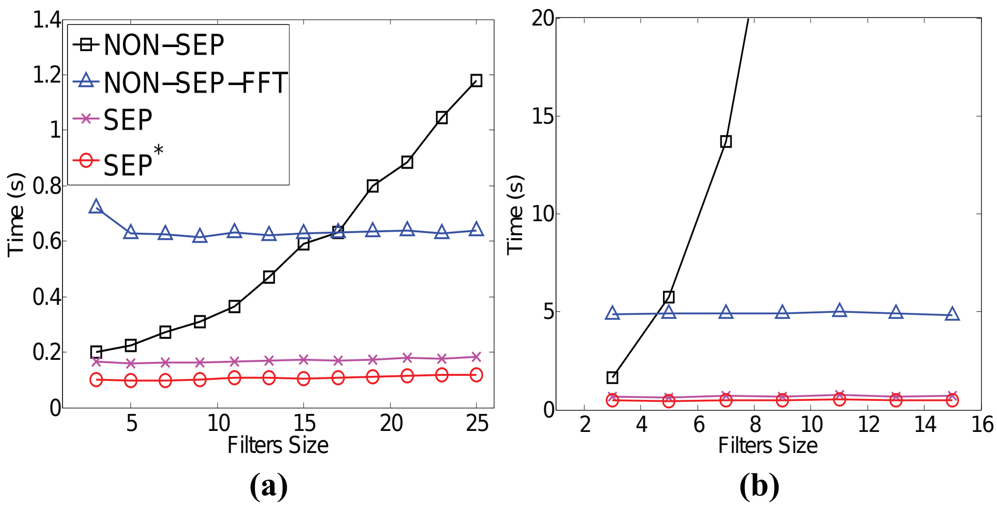


Figure 4.4: Time needed to compute convolutions using a multi-thread MATLAB implementation, as a function of the filters size. **(a)** An image of 488×488 pixels is convolved with a filter bank of $J = 121$ non-separable filters and $K = 25$ separable ones. **(b)** A volume of $114 \times 114 \times 50$ voxels is convolved with a filter bank of $J = 121$ non-separable filters and $K = 25$ separable ones. Using parallel computation the time needed to compute the convolution is further reduced and our methods are still the most efficient ones. The times are averaged over 5 repetitions.

The nD Case

The generalization to any dimension is straightforward. If $\mathbf{x} \in \mathbb{R}^{N_i \times, \ldots, \times N_n}$ is an n-dimensional image and $\mathbf{f}_j \in \mathbb{R}^{d_1 \times, \ldots, \times d_n}$ an n-dimensional filter, the cost of a non-separable convolution becomes $\prod_{i=1}^{n} N_i \cdot d_i$. The cost of a separable convolution is $(\prod_{i=1}^{n} N_i) \cdot (\sum_{i=1}^{n} d_i)$ and the cost of a FFT is $c \cdot m \cdot \log_2(m)$, where m is the product, over index i, of the closest larger powers of 2 of $N_i + d_i - 1$.

Fig. 4.3(b) and Fig. 4.4(b) illustrate that using separable filters is even more advantageous for the 3D case. Here the size of the filters is between 3 and 15 and a $114 \times 114 \times 50$ volume is considered. Again the size of the volume is taken at the advantage of the FFT based approach.

4.5 Experimental Evaluation

In this section, we show the generality of our separable filter approach by applying it to two different Compute Vision tasks involving convolution with a filter bank. First we consider a voxel classification task of biomedical images, by extending the approach of [164] to the 3D case. Then, we consider an image classification problem with Convolutional Neural Networks [116]. We compare the performance and computational complexity that results from using either separable filters or non-separable ones and different strategies for deriving them.

We will show that our separable filters systematically deliver a substantial speed-up at no significant loss in performance. This is in line with our theoretical analysis of the previous section. The code and parameters for all these experiments are publicly available at `http://cvlab.epfl.ch/software/filter-learning`.

4.5.1 Detection of Curvilinear Structures

In this section, we apply the separable filter approximation schemes described in Sec. 4.3 to the problem of segmenting linear structure in 3D biomedical volumes. We consider as baseline the approach of [164], which uses a filter bank learned with Eq. (4.1) to extract the features used for the segmentation task. The goal of this section is therefore to achieve the same level of performance of [164], but much faster.

Notice that, even if [164] consider the problem of segmenting the linear structures, the features used in this task are the same as the ones we use in Chap 3 to train our regressors

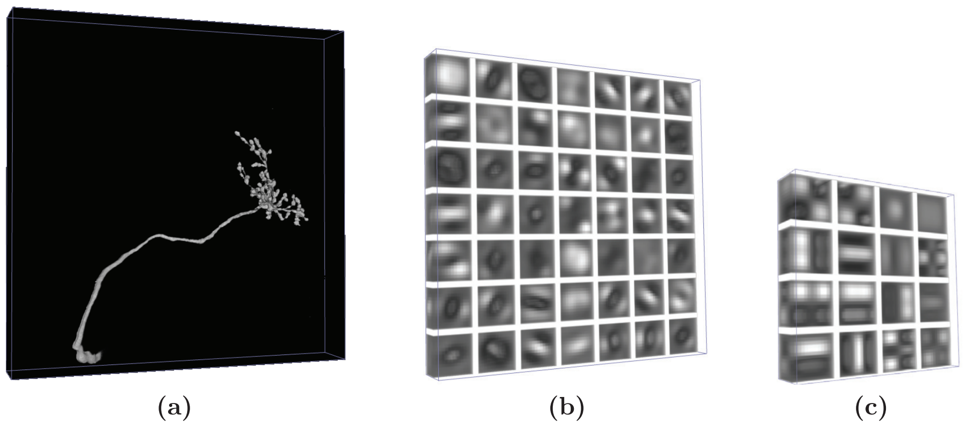


Figure 4.5: Filters learned for the voxel classification task of Sec. 4.5.1. **(a)** Example volume from the OPF dataset. **(b)** Non-separable filters learned with Eq. (4.1). **(c)** Separable filters learned with the Tensor Decomposition approach of Eq. (4.6) to reconstruct the filters in **(b)**.

for centerline detection. This is why we consider it for validating the use of separable filters also in our case.

As described in Sec. 1.1, biomedical image processing is a particularly promising field of application for Computer Vision techniques as it involves large numbers of 2D images and 3D image stacks of ever growing size, while imposing strict requirements on the quality and the efficiency of the processing techniques. Here, we demonstrate the power of our separable filters for the purpose of identifying curvilinear structures.

As discussed in Sec. 3.4, in [164] it was showed that convolving images with non-separable filter banks learned by solving the problem of Eq. (4.1) and training an SVM on the output of those filters outperforms previous methods for the task of linear structures segmentation [170, 77]. Unfortunately, this requires many such non-separable filters, making it an impractical approach for large images or image stacks, whose usage is becoming standard practice in medical imaging. Here we show that our approach solves this issue.

The dataset we use for evaluation is composed of 3D volumes of Olfactory Projection Fibers (OPF) from the DIADEM challenge [13], which were captured by a confocal microscope. We first learned the non-separable 3D filter bank made of 49 $13 \times 13 \times 13$ pixel filters and then approximate them with 16 separable filters, as described in Sec. 4.3. An image from the datasets and the filters learned on it are shown in Fig. 4.5.

We train a classifier to use these filters using ℓ_1-regularized logistic regression. Since this classifier does not require us to compute the linear combination of the separable filter outputs, we can rely on the *SEP-COMB** and *SEP-TD** approach for our experiments. For training we used a set of 200000 samples, randomly selected from 4 train images.

Table 4.2: Analytic measure of the performance of the voxel classification task over the OPF dataset. The VI and RI values are compared on the classification thresholded at the value found using the F-measure. For the learning-based approaches, a training set of 200000 randomly selected samples and a ℓ_1-regularized logistic regressor classifier have been used. Approaches that use a separable filter basis have been found to reduce the computational costs by a factor of 30 in classifications tasks.

Method	**AUC**	**F-measure**	**VI**	**RI**	**Time[s]**
		OPF:Image 4			
OOF	0.997	0.531	0.012	0.998	193.05
NON-SEP-FFT(49)	0.997	0.571	0.013	0.998	339.01
SEP-CPD(49)	0.997	0.567	0.013	0.998	40.06
SEP-COMB(16)*	0.997	0.570	0.013	0.998	11.08
SEP-TD(16)*	0.997	0.567	0.013	0.998	11.08

We use *NON-SEP* as our baseline. We compare *SEP-COMB** and *SEP-TD** against *NON-SEP-FFT*. For completeness, we compare our results to those obtained using the Optimally Oriented Flux [113], which we will refer to as *OOF*, and *SEP-CPD*.

As evaluation metric we use: Area Under Curve (AUC), which represents the area subtended by the ROC curve; the F-measure [200]; the Rand Index (RI) [198] and the Variation of Information (VI) [140]. The first three measures assumes values in $[0, 1]$, the higher, the better, while VI assumes values in $[0, \infty)$, the lower the better.

The results are reported in Tab. 4.2. *SEP-COMB** and *SEP-TD** are 30 times faster than *NON-SEP-FFT* for virtually the same accuracy. They are 4 times faster than *SEP-CPD*, but as *OOF*, *SEP-CPD* is worse in terms of accuracy.

4.5.2 Convolutional Neural Networks

In recent years, Convolutional Neural Networks (CNNs) have become increasingly popular and have been shown to improve upon the previous state-of-the-art for many challenging tasks. However, they are computationally intensive and their wide acceptance has been achieved only after the introduction of powerful GPU implementations. In this section, we show that replacing the non-separable filters they typically use by separable ones can help alleviate this problem.

Recent works such as [49, 46] have addressed this issue, albeit in a different way. The authors of [46] focus on reducing complexity at training time. They show that thanks to the correlation present in the weights, it is possible to optimize only a small fraction of the parameters and predict the remaining ones starting from them, without losing accuracy at test time.

Table 4.3: Handwritten digit recognition on MNIST dataset with Convolutional Neural Networks. Different kernel sizes are used in the first and second convolutional layers to evaluate the effect of kernel size on the classification performance and the execution time. The classification results and the execution times are reported for separable and non-separable filters. By using our separable filters approximation scheme, we can divide the computational times by 2, at the cost of a negligible accuracy loss.

MNIST					
Kernel Size		Misclassification Rate		Execution Time	
1^{st} Layer	2^{nd} Layer	*SEP-TD*	*NON-SEP*	*SEP-TD*	*NON-SEP*
5	5	5.27%	5.17%	27.33	28.77
5	9	4.84%	4.17%	24.9	44.61
9	9	3.47%	3.17%	21.9	48.54

Although less effective, the approach of [49] is more similar in spirit to ours. They reduce the complexity of the convolutional layers at test time by using smaller but non-separable operators, reducing computation by a 1.6 factor. By contrast, by using separable filters learned with our method, we show up to a factor 3 speed up.

Finally, the three approaches could be combined to achieve an even greater-speed up. A network could be trained using the approach of [46], then the decomposition [49] could be applied to obtain a smaller set of filters, to be turned into separable ones by through approach.

In our experiments, we considered the following two datasets:

- The MNIST dataset [117] is a standard Machine Learning benchmark that consists of 70000 images of hand-written digits. The training set contains 60000 images and the test set 10000. For training we use a batch size of 50 with a learning rate of 1 for 5 epochs.
- We built a Drone Detection dataset that consists of 40×40 images in which a drone rotorcraft may or may not appear. The task is to determine whether the drone is present or not, with a view to automated visual collision avoidance in swarms of such drones. The training and test datasets both contain 10950 images. Representative samples are shown in Fig. 4.7. Note that they are low-resolution and subject to motion-blur. For training we use a batch size of 10 with a learning rate of 1 for 100 epochs.

We consider an architecture consisting of 4 fully connected hidden layers for the MNIST dataset and 5 fully connected hidden layers for the Drone Detection dataset.

We trained the networks using different kernel sizes in order to study the influence on

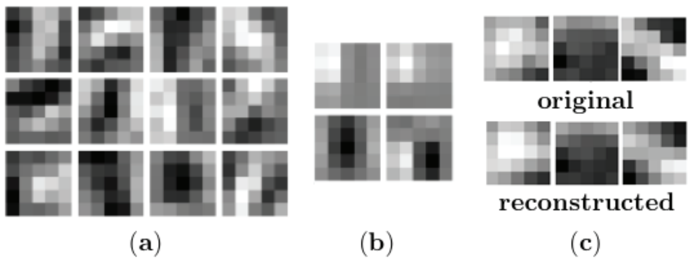


Figure 4.6: Filters learned for digit classification with Convolutional Neural Networks. **(a)** 12 filters going out of a node of the second convolutional layer. **(b)** A set of 4 separable filters obtained after Tensor Decomposition. **(c)** Comparison of the original and the approximated filters.

the performance and the execution time. For our experiments, we used a publicly available Deep Learning MATLAB toolbox [151].

The first layer consists of 6 feature maps connected to the single input layer via 6 kernels. The second layer is a 2-by-2 downsampling layer. The third layer consists of 12 feature maps connected to the 6 downsampling layers via 72 kernels. The fourth layer is again a 2-by-2 downsampling layer.

For digit classification, the feature maps obtained at the last layer are concatenated into feature vectors and fed into the last layer, which has 10 output neurons in order to do multiway classification between 10 handwritten digit characters.

For drone detection, one more convolution layer consisting of 24 feature maps fully connected to the fourth layer is added. These feature maps are fed into 2 output neurons in order to discriminate whether the image contains a drone or not.

To obtain separable kernels, we apply the *SEP-TD* approach at each convolution layer. For the MNIST dataset, the 6 kernels in the first layer are approximated using 3 separable filters. In the second layer, we group the 12 filters corresponding to each outgoing feature map together and approximate them by 4 separable filters independently.

For the Drone Detection dataset, 4 separable filters are used to approximate 6 filters in the first convolution layer. In the second convolution layer, 12 filters at each outgoing feature map are approximated with 5 filters independently. In the third convolution layer, the 24 filters at each outgoing node of the third convolution layer are approximated by 9 filters.

Using separable filters speeds up the Convolutional Neural Network without loss in accuracy. The execution times and the misclassification rates are reported in Tables 4.3 and 4.4 for different kernel sizes. In particular, for a kernel size of 9, classification becomes

Table 4.4: Drone detection task with Convolutional Neural Networks. As for the MNIST dataset, we considered different kernel sizes in the convolutional layers. Using separable filters the execution time can be reduced up to a factor of 3 without decreasing accuracy.

Drone Detection						
Kernel Size			Misclassification Rate		Execution Time	
1st Layer	2nd Layer	3rd Layer	*SEP-TD*	*NON-SEP*	*SEP-TD*	*NON-SEP*
5	5	5	0.17%	0.16%	9.72	17.55
5	5	9	0.09%	0.08%	19.69	44.33
5	9	9	0.02%	0.02%	36.35	102.15

Figure 4.7: Some positive class training images from the Drone Detection dataset. A Convolutional Neural Network is trained to classify images containing a rotorcraft drone. We use separable filters to speed up the execution time of the convolutional layers. See text for more details.

two to three times faster. Note that the purpose of these experiments is not necessarily to achieve state-of-the-art performance in a given task using CNNs, but to prove that our approach can be used on an arbitrary CNN to speed up convolutions without loss in accuracy, which is what it does.

Moreover, notice that, in order to further improve performance, one could fine tune the last fully-connected layer, while keeping fixed the layers approximated by separable filters.

Fig. 4.6 shows the filters learned in the second convolutional layer learned on the MNIST dataset and the 4 separable filters used to approximate them. Fig. 4.6(c) presents a visual comparison between the original and the reconstructed filters.

Finally, we observe that even though our approach is able to speed up the execution of Convolutional Neural Networks at test time, the main bottle neck associated to these models is the time required for training. However, the idea of decomposing filter banks as linear combination of separable filters could be adapted also to speed up the training of the network. We discuss this possibility in detail in Chap 7.

4.5.3 Comparison between *SEP-COMB* and *SEP-TD*

In Sec. 4.3, we showed that an arbitrary filter bank can be approximated by linear combinations of separable filters. We also proved that such decomposition can be used in several Computer Vision tasks to decrease the computational complexity without substantial changes in accuracy.

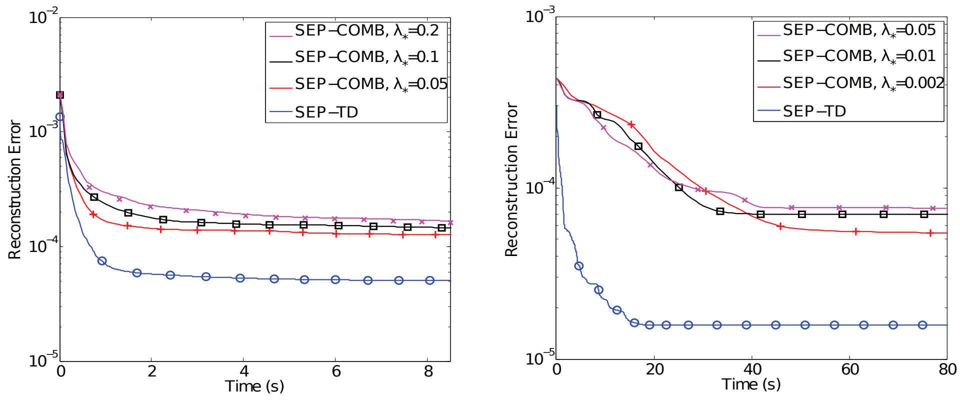
In this section, we compare *SEP-COMB* and *SEP-TD* in terms of approximation error and learning time. We start by studying the convergence rate of the two different approaches. We compute the reconstruction error as the Root Mean Square Error between the original filter bank and the filter bank approximated by the separable filters, and plot it as a function of the learning time. For *SEP-COMB*, we also considered different values of the parameter λ_*. The results are shown in Fig. 4.8. From the figure, we can see that *SEP-TD* converges faster and has lower reconstruction error both in the 2D and 3D case.

Notice that, as expected, for *SEP-COMB* the error decreases as the value of λ_* decreases. In fact, with lower values of λ_*, the squared error term in Eq. (4.6) has more weight. However, lower values of λ_* penalize less high rank filters and therefore, in this situation, it is more difficult to obtain separable filters at the end of the optimization.

We then considered a second series of experiments by computing the error as a function of the number of separable filters used in the approximation. The results are shown in Fig. 4.9. Also in this case we see that *SEP-TD* constantly returns better approximations than *SEP-COMB*, for all the values of λ_* considered and both in 2D and in 3D.

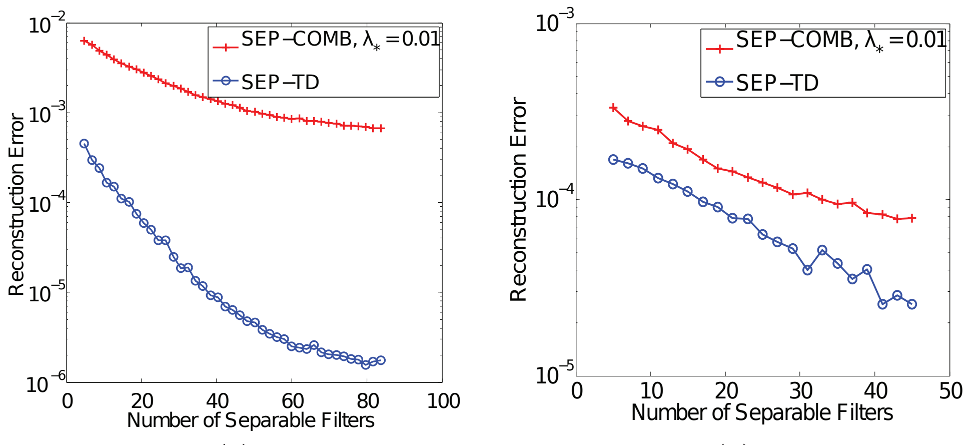
To summarize, the advantages of *SEP-TD* compared to *SEP-COMB* are:

- **Parameter reduction**: The only parameter of *SEP-TD* is the number of separable filters used to approximate the original one, while *SEP-COMB* relies on a regularization parameter.
- **Faster convergence**: The *SEP-TD* approach converges faster than *SEP-COMB*. The advantage of using *SEP-TD* rather than *SEP-COMB* is more pronounced in the 3D case. Moreover, the *SEP-COMB* approach does not guarantee that the filters actually have rank-1 after convergence and thresholding on the singular values might be applied.
- **Lower approximation error**: This can be explained by the fact that in the *SEP-TD* approach, a non-separable filter is explicitly written as the sum of the products of 1D filters. This approach provides a better approximation quality then *SEP-COMB*, which relies on a soft constraint to make the filter ranks low.



(a) **(b)**

Figure 4.8: Comparison of the reconstruction errors of *SEP-TD* and *SEP-COMB* as a function of the learning time for approximating **(a)** a 2D non-separable filter bank and **(b)** a 3D non-separable filter bank. The performance of the *SEP-COMB* approach depends on the specified regularization parameter λ_*. Small regularization parameters yield a smaller reconstruction error. *SEP-TD* does not need to satisfy an additional constraint and yields a smaller reconstruction error compared to *SEP-COMB* with a faster convergence. In the 3D case, the difference is even more pronounced.



(a) **(b)**

Figure 4.9: Comparison of the reconstruction errors of *SEP-TD* and *SEP-COMB* as a function of the number of separable filters used to approximate **(a)** a 2D non-separable filter bank of 121 filters and **(b)** a 3D non-separable filter bank 49 filters. The results are averaged over 10 repetitions. Also in this case we observe that *SEP-TD* returns more accurate results than *SEP-COMB* for every number of separable filters. Moreover, when the number of separable filters increase, the error decreases faster for *SEP-TD* than *SEP-COMB*.

Conclusion

In this chapter, we have proposed a learning-based filtering scheme applied to the extraction of curvilinear structures, along with two learning-based strategies for obtaining a basis of separable filters to approximate an existing filter bank. Thanks to this approximation, we get the same performance as with the original filter bank. Moreover, we also considerably reduce the number of filters, and thus, the number of convolutions. We presented two optimization schemes. In the first one the separable filters are learned by lowering their ranks. In the second one, which proved to be more efficient and accurate, the filters are obtained by Tensor Decomposition.

Our techniques bring to learning approaches one of the most coveted properties of hand-crafted filters, namely separability, and therefore reduce the computational burden traditionally associated with them. Moreover, designers of hand-crafted filter banks do not have to restrict themselves to separable filters anymore: they can freely choose filters for the application at hand, and approximate them using few separable filters with our approach.

For example, in [202] our separable filters approximation scheme was applied to a filter bank learned to detect feature points. Thanks to the separable approximation, the authors obtained an improvement of both speed and accuracy compared to the original non-separable case.

Moreover, after the introduction of separable filters approximation scheme, many other works considered the use of separable filters, in particular for speeding up Convolutional Neural Networks [94, 115, 103, 122]. This shows once more the relevance of our approach.

The features extracted with the method described in this chapter achieve state-of-the-art performance when used as input to our regression-based method of Chap. 3 for centerline regression. However, since the features extracted with these filters are local, they are not able to discriminate ambiguous situations, where contextual information is required.

In the next chapter, we show how the convolutional filters described in this chapter can be used to extend our regression-based method of Chap. 3 to a context-aware method. Thanks to this new approach we can further improve the accuracy of our method for centerline detection and radial estimation.

CHAPTER 5

Leveraging Contextual Information for Centerline Detection

In Chap. 3, we have reformulated centerline detection as a regression problem. We have introduced a supervised learning method, which consists in learning a set of pixel-wise regressors to approximate a scale-space distance transform, whose local maxima correspond to centerline locations and radii.

Performing Non-Maximum Suppression on the regressors output yields a large improvement in the accuracy for centerline and radii estimation, compared to previous methods. However, the method of Chap. 3 only relies on local information. Therefore, it is not able to discriminate linear structures in ambiguous situations, where a larger context is required. As a consequence, false detections on the background or gaps in the linear structures are present in the output score map.

In this chapter, we extend our method by taking advantage of contextual information. We consider context not only in the spatial domain, by widening the portion of the image considered to extract the features, but also in the radial dimension, by incorporating information across scales. Using the taxonomy we introduced in Chap. 2, the method presented in this chapter can be considered a scale-space context-aware method.

We achieve this by adapting the Auto-Context algorithm [193], which was originally proposed for image segmentation, to a multiscale regression framework. More precisely, we use the outputs of the regressors as features to a layer of new ones. By iterating this process, we can progressively correct earlier mistakes by exploiting information across a widening portion of the image and across scales.

We validate our approach with extensive numerical experiments and show that it gives better performance both for pixel-wise evaluation and also when used in conjunction with a tracing algorithm, compared to state-of-the-art methods.

Moreover, our approach is very generic and also performs well on contour detection. We show an improvement above pixel-wise and patch-wise contour detection algorithms on the BSDS500 dataset [10].

The rest of the chapter is organized as follows: In Sec. 5.1, we underline the importance of context for the centerline detection problem and show some failure cases of the local method of Chap. 3. Then, we introduce an iterative regression method and show how to incorporate spatial and radial context in our algorithm in Sec. 5.2 and Sec. 5.3, respectively. The features used as input to our iterative regression method are described in Sec. 5.4. Experimental results for centerline detection and tracing are provided in Sec. 5.5, while in Sec. 5.6 we apply our method to boundary detection.

Part of the content of this chapter has been previously published in [182].

5.1 The Importance of Context for Centerline Detection

Contextual information is fundamental to solve many Computer Vision problems [186, 188, 158, 73]. While local information might be sufficient for many low-level vision tasks, such as detecting changes in intensity values, estimating local orientation, denoising, etc., for higher-level tasks, information from a larger portion of the image is needed. For example, texture analysis, detection of occluding contours and object recognition are tasks that, in general, can not be solved by local methods alone.

For the problem of linear structure detection, local methods are adequate when the structures clearly appear in the image as elongated segments, strongly emerging from the background and when no other locally elongated structure is present in the image. However, in situations, such as those of Fig. 5.1, where occlusions, low contrast or structured noise appear, local information is not enough (Fig. 5.1(c)). On the contrary, by considering a larger portion of the image, these ambigous situations can be easily discriminated (Fig. 5.1(b)).

One of the main difficulties in using context in a Computer Vision system is how to take advantage of this large amount of information while keeping computation efficient.

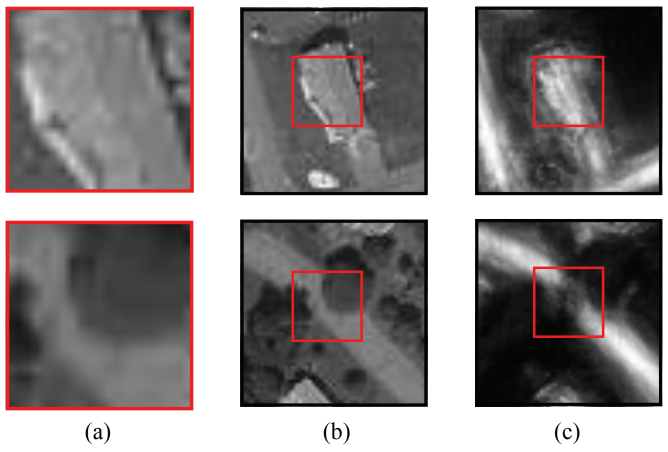


Figure 5.1: Importance of context for centerline detection. (a) Two local patches of size 21×21. The patch in the top row appears as a line-like structure, although it corresponds to a roof top. The patch in the bottom row is centered on a road pixel, but it is partially occluded by a tree. (b) Enlarged views of the patches in (a), of size 55×55. Thanks to the additional contextual information road and non-road segments are recognizable. (c) Score map obtained with the method of Chap. 3. Since the method only relies on local information, it makes mistakes on the pixels corresponding to the patches in (a).

Many solutions have been proposed in the literature. A classical, and still in wide use, method is the Hough transform [89, 55], which combines the response of local detectors using a voting technique. Other approaches, instead, rely on a set of context-aware features [183, 22] that can be computed efficiently. Among them, pooling techniques [28, 75] are a standard way to incorporate context. They are typically applied to a first layer of locally extracted features and they proved to be effective to obtain local invariance and discriminate texture [32]. In Auto-Context-like methods [193], contextual features are extracted from the output of a classifier. Iterating this process, the receptive field of the method is enlarged at every new iteration.

In order to detect centerlines, we will use the same principles of the methods described above. First, inspired by [22], we will use a set of contextual features. For every pixel, our regressor will use the features extracted not only from a local window centered at that pixel, but also from pixels within a given radius from it. We will describe in Sec. 5.4 how this can be done efficiently.

Second, we remark that if we are given an approximated score map estimating the location of the centerline, and in the score map there is a location with a strong response, but which is isolated, *i.e.* no other centerline is detected around it, then it is likely that this isolated response corresponds to a false detection. Similarly, if in the score map there is a line-like and continuous response with only a small gap in it, where the response of the detector is weaker, we can conclude that in reality also the pixels in the gap belong to the linear structure. In Sec. 5.2, we show how the Auto-Context method can be used to implement efficiently these observations and improve the score map returned by our regressors.

Finally, the same reasoning described above can be done for the response of the regressor at the different scales. As a consequence, in Sec. 5.3, we describe how we can improve radial estimation by aggregating radial contextual information.

5.2 Adding Spatial Context

In this section, we first briefly recall the method and the notations used in Chap. 3. Then, we describe how contextual information can be used to improve the performance of our method in the case of structures with fixed radius. The multiscale case is considered in the next section.

Given an image I, containing linear structures and the binary ground truth Y, corresponding to the set of centerline points C, finding the centerlines can be formulated as a binary classification problem of learning a mapping between a feature vector $f(\mathbf{x}, I)$ extracted from a local neighborhood of pixel $\mathbf{x}$, and the value of Y at $\mathbf{x}$.

Learning such a classifier, however, can be difficult in practice because of the similar aspect of nearby pixels to the centerline and ambiguities on the exact location of a centerline due to low resolution, occlusions and blurring.

To address this difficulty, in Chap. 3 we have replaced the binary ground truth Y by the modified distance transform d_C of Y:

$$d_C(\mathbf{x}) = \begin{cases} e^{a(1-\frac{\mathcal{D}_C(\mathbf{x})}{d_M})} - 1 & \text{if } \mathcal{D}_C(\mathbf{x}) < d_M \\ 0 & \text{otherwise} \end{cases}, \tag{5.1}$$

where $\mathcal{D}_C$ is the Euclidean distance transform of the set C, $a > 0$ is a constant that controls the exponential decrease rate of d_C close to the centerline and d_M a threshold value determining how far from a centerline d_C is set to zero. An example of the function d_C computed on a small patch is shown in Fig. 3.2.

Function d_C has a sharp maximum along the centerlines and decreases as one moves further from them (Fig. 3.3). We apply the GradientBoost algorithm [82] to learn a regressor $\varphi(\cdot)$ that associates the feature vector $f(\mathbf{x}, I)$ to $d_C(\mathbf{x})$. In this way, we induce the output

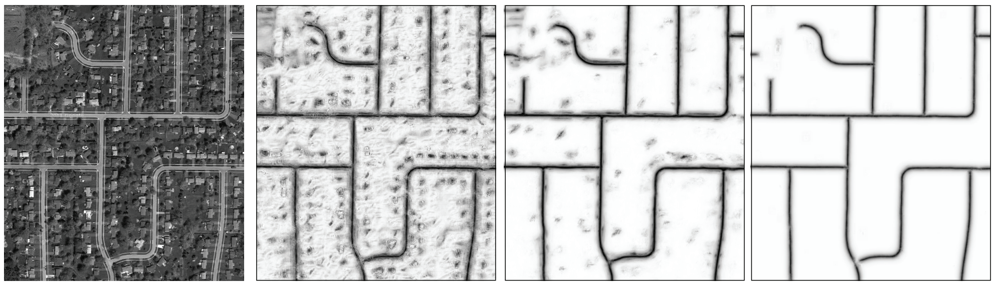

(a) Input Image I
(b) Score map $\varphi^{(0)}$
(c) Score map $\varphi^{(M)}$
(d) Final approximation Φ

Figure 5.2: Improvement obtained by our context-aware iterative regression method. (a) Input image $I(\mathbf{x})$; (b) Score map $\varphi^{(0)}(\cdot)$ obtained for $M = 0$, which corresponds to the method of Chap. 3; (c) Score map $\varphi^{(M)}(\cdot)$ with $M = 2$ obtained using our context-aware approach, as described in Sec. 5.2; (d) Score map $\Phi(\cdot)$ obtained by adding the multiscale learning step described in Sec. 5.3. Both the iterations and the last multiscale regressor help to remove false detections on the background and to obtain a better localization accuracy of the centerlines. For (b), (c), and (d) we show the maximum projection along the radial dimension for visualization purposes.

of the regressor to have a unique local maximum in the neighborhood of the centerlines. As a result, centerline points can be easily extracted by Non-Maximum Suppression. This approach is more robust to small displacements and returns centerlines that are better localized compared to classification-based methods.

However, as explained in the previous section, using $\varphi(\cdot)$ to predict function $d_C(\cdot)$, may result in incorrect large values on the background or missed parts of the linear structures, since only local information is used for prediction purposes.

These mistakes can be avoided by including more contextual information in the algorithm, as is done in the so-called Auto-Context algorithms [193, 176] for classification purposes. To this end, we use the score map $\varphi(\mathbf{x})$ to extract a new set of features able to discriminate isolated responses on the background and to fill gaps in the detected structures. These new features are added to the original ones to train a new regressor. By iterating this process we obtain a sequence of regressors able to include more and more contextual information in the learning algorithm and to correct the mistakes done at the previous iterations.

In the following, we describe how the regressors are learned, assuming we are given two set of features f and g, extracted from the image and from the score map respectively. In Sec. 5.4, we describe in detail how the set of features is computed.

Let $\{(f_i, d_i)\}_i$ be the set of training samples used to learn the first regressor $\varphi^{(0)}(\mathbf{x}) = \varphi(\mathbf{x})$, where $f_i = f(\mathbf{x}_i, I_i) \in \mathbb{R}^J$ is the feature vector corresponding to a point $\mathbf{x}_i$ in image I_i and $d_i = d_C(\mathbf{x}_i)$, as described in Sec. 3.2.

Now, let us consider a new feature vector $g(\mathbf{x}, \varphi^{(0)})$, extracted from the score map

$\varphi^{(0)}(\mathbf{x})$ and let $\{(f_i, g_i, d_i)\}_i$ be the new training set, with $g_i = g(\mathbf{x}_i, \varphi_i^{(0)}) \in \mathbb{R}^{J'}$ and $\varphi_i^{(0)} = \varphi^{(0)}(f(\mathbf{x}, I_i))$. In order to learn a new regressor, able to improve the performance of the previous one, we apply again the GradientBoost algorithm to learn a better approximation of the function $d_C(\cdot)$:

$$\varphi^{(1)}(f(\mathbf{x}, I), g(\mathbf{x}, \varphi^{(0)})) = \sum_{t=1}^{T} \alpha_t^{(1)} h_t^{(1)}(f(\mathbf{x}, I), g(\mathbf{x}_i, \varphi_i^{(0)})) \ . \tag{5.2}$$

We iterate this process M times learning a series of regressors

$$\{\varphi^{(m)}(f(\mathbf{x}, I), g(\mathbf{x}, \varphi^{(m-1)}))\}_{m=0,\ldots,M}. \tag{5.3}$$

The final output $\varphi^{(M)}(\cdot)$ will be used as approximation of $d_C(\cdot)$. For the sake of brevity, we will write $\varphi^{(m)}(\mathbf{x})$ instead of $\varphi^{(m)}(f(\mathbf{x}, I), g(\mathbf{x}, \varphi^{(m-1)}))$.

To prevent overfitting, which is a known weakness of Auto-Context frameworks, we adopt several strategies. First, at the beginning of each Auto-Context iteration new pixel locations are sampled from the train images to build a new training set. Second, at each boosting iteration t we learn the weak learner h_t using only a random subset of the whole training set, as in Stochastic GradientBoost [70]. Finally, as discussed in Sec. 5.4, the features used to learn a weak learner are also subsampled at each boosting iteration. Fig. 5.2(c) shows the advantage of using iterating regression.

In practice, the method converges fast: the performance does not improve beyond the second iteration and we therefore set $M = 2$ in all our experiments.

5.3 Adding Scale-Space Context

In this section, we consider the centerline detection problem in the case where structures at several scales are present in the image. Now, at every centerline pixel is associated a radial value r that we also want to estimate. In Chap. 3, this was achieved by learning a set of regressors $\{\varphi_{r_i}(\cdot)\}_{i=1}^{R}$ to approximate the multiscale version of function d_C, defined in Eq. (3.8).

As in the single-scale case, and as summarized in Fig. 5.3, we use the Auto-Context strategy to improve the accuracy of our method and create a sequence $\{\varphi_{r_i}^{(m)}(\cdot)\}_{i,m}$ of scale-space regressors.

Potentially, all the outputs at every scale, obtained at iteration $m-1$ could be used to train the regressors for the next iteration. However, this would increase the learning and

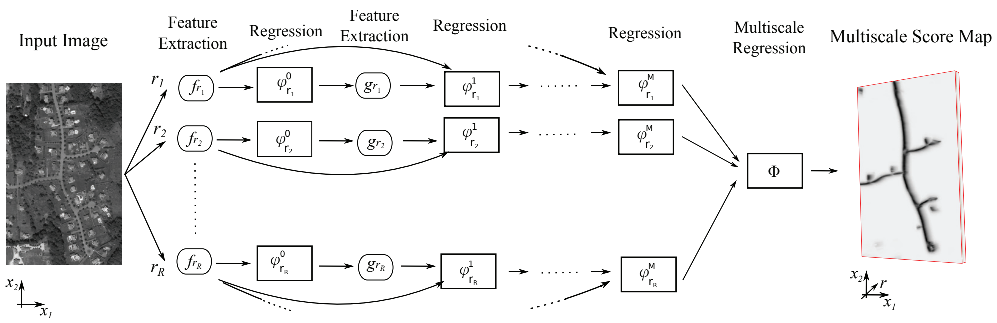


Figure 5.3: Iterative regression for multiscale centerline detection. For each value of radius r, the input image is convolved with a bank of filters to extract a set of image features. The features are used as input to regressors $\varphi_r^{(0)}$. The outputs of the regressors are then convolved with other filter banks to extract new features. These features are fed together with the image features to a second layer of regressors $\varphi_r^{(1)}$. This process is iterated M times. In the final step, the output of the regressors is fed to Φ, a multiscale regressor, that computes the final score map.

test times considerably. We therefore adopt a different strategy and divide the learning process into simpler subproblems. At the first M iterations the regressors at different scales are learned independently, that is for each r_i, $\varphi_{r_i}^{(m)} = \varphi_{r_i}^{(m)}(f^{r_i}(\mathbf{x}, I), g(\mathbf{x}, \varphi_{r_i}^{(m-1)}))$. Then, as a final step, we take the score maps obtained at all scales $\{\varphi_{r_i}^{(M)}(\mathbf{x})\}_{i=1}^{R}$ and use them to train a last multivariate regressor $\Phi(\cdot)$, where now $\Phi(\mathbf{x}) \in \mathbb{R}^R$.

We build the function $\Phi(\cdot)$ again with GradientBoost:

$$\Phi(\mathbf{x}) = \sum_{t=1}^{T'} \alpha_t h_t(\{\varphi_{r_i}^{(M)}(\mathbf{x})\}_{i=1}^{R}) , \tag{5.4}$$

where now the weak learners $h_t(\{\varphi_{r_i}^{(M)}(\mathbf{x})\}_{i=1}^{R}) \in \mathbb{R}^R$ return a vector of values, each component corresponding to a different scale. We use $\Phi(\cdot)$ as the final approximation of the scale-space function d_C.

This last step imposes consistency and smoothness on the values returned by the previous regressors, which were trained independently and improve localization and radial estimation accuracy. Fig. 5.2(d) shows the advantage of this last step on a sample road image.

5.4 Scale-Space Context-Aware Features

In this section, we discuss the image features we feed as input to the regressors described in Sec. 5.2 and Sec. 5.3.

In order to extract the image features $f(\mathbf{x}, I)$, we again rely on the convolutional filter learning framework described in Sec. 4.2. However, compared to the method of Chap. 3, we exploit additional image information by also considering locations within a certain distance of location $\mathbf{x}$. This produces a much larger pool of possible features

$$f(\mathbf{x}, I) = \{(\mathbf{f}_j * I)(\mathbf{x} + \mathbf{l})\}_{j,\mathbf{l}}\,, \tag{5.5}$$

where the $\mathbf{f}_j$ are the learned filters and with $\mathbf{l} \in \mathbb{R}^n$, $\|\mathbf{l}\| \leq \Lambda$ for some fixed $\Lambda > 0$. For example, for a 121-filter bank, setting $L = 13$ results in approximately $64,200$ possible features. We handle this potentially large number by considering at each boosting iteration only a random subset of all possible locations and convolutional features. The GradientBoost algorithm will automatically select the most relevant features and locations. Moreover, randomly subsampling them has the added benefit of reducing overfitting.

We use the same filter banks as in Chap. 3. To speed up the computation, we again approximate the filters $\{\mathbf{f}_j\}_j$ with a set of separable ones, as described in Chap. 4.

After iteration m, we extract a new set of features from the score map $\varphi^{(m)}(\cdot)$ that will be used in the next iteration. As for the image information, we use the method of Sec. 4.2 to learn filters specifically trained to extract features from $\varphi^{(m)}(\cdot)$.

Ideally we should learn a set of filters on the score maps at each iteration. In practice, however, this would be prohibitively expensive. In biomedical imagery, the background is relatively uniform and the score maps produced by the regressors exhibit characteristics similar to those of the original images. We therefore use the same filters as for the image to extract features from the score maps. In other kinds of images, such as aerial or natural images, which contain objects other than the linear structures, we learn instead a bank of filters from the ground truth training images $d_C(\mathbf{x})$. This produces filters able to detect linear structures and junctions similar to those we would have learned on the score maps.

Neighbor locations are again considered to capture more context. Formally, the feature vector on the score image at the iteration m is given by $\{\mathbf{g}_j * \varphi^{(m)}(\mathbf{x} + \mathbf{l})\}_{j,\mathbf{l}}$. To keep the computational complexity under control, we subsample the set of features at each boosting iteration also in this case.

In the case of multiscale detection, at the last iteration, the function Φ is learned from all the previously computed maps $\{\varphi_{r_i}^{(M)}\}_{i=1}^{R}$ and we do not use features extracted from the original image anymore. This compensates for the increased number of score map features. Moreover, image features are not really needed here because the purpose of this last step is to enforce score consistency over the different scales.

5.5 Experimental Evaluation

In this section, we first introduce the datasets and the parameters used to test our centerline detection method, in Sec. 5.5.1. We describe our evaluation methodology and we discuss our results. We then evaluate the improvement brought by our method when used in conjunction with automated tracing algorithms. We first evaluate the accuracy of our method for tracing paths on the linear structure, in Sec. 5.5.2. Then, in Sec. 5.5.3 we evaluate its performance when using it for reconstructing the whole graph of the linear structure. Finally, in the next section, we apply our algorithm to the problem of boundary detection in natural images.

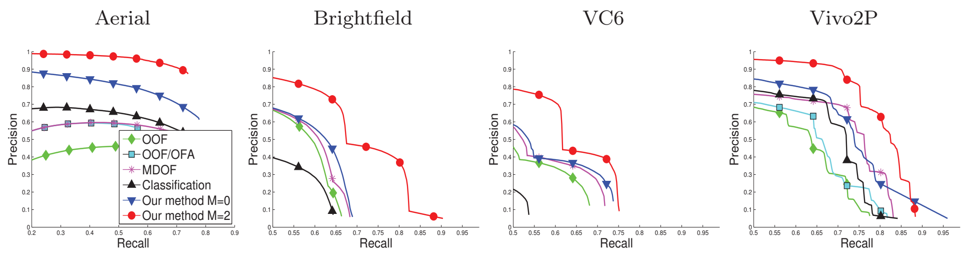
5.5.1 Pixel-Wise Evaluation

In this section we repeat the pixel-wise evaluation of Sec. 3.6, for centerline detection and radial estimation, as done for the method of Chap. 3.

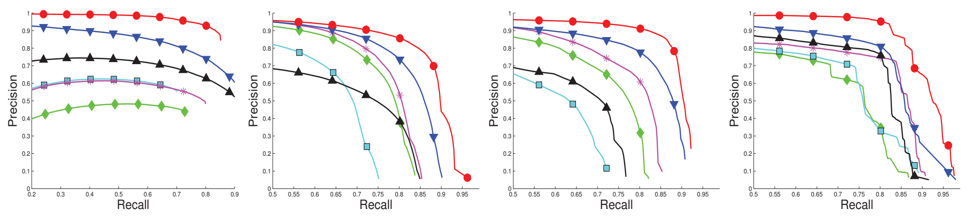
We use the same datasets and the same evaluation metrics as in that chapter, with the only exception that we use an extended version of the Aerial dataset. The new Aerial dataset is composed of 13 images for training and 13 images for testing. For this dataset, we sample 10 scales ranging from 5 to 14. We trained 2 regressors at scales 6 and 9.

For iterative regression, we always used $M = 2$ iterations and the maximum contextual radius Λ was set to 13. For the Aerial dataset, we learned a bank of 36 convolutional filters on the distance transform ground truth and applied it to the score map to extract the features used during iterative regression. In order to train the final regressor Φ, more samples were needed to obtain good performance. We therefore used 10^6 samples for this purpose, half of them sampled close to the centerlines (within a distance d_M) and the other half far from it. All the other parameters are the same as in Chap. 3.

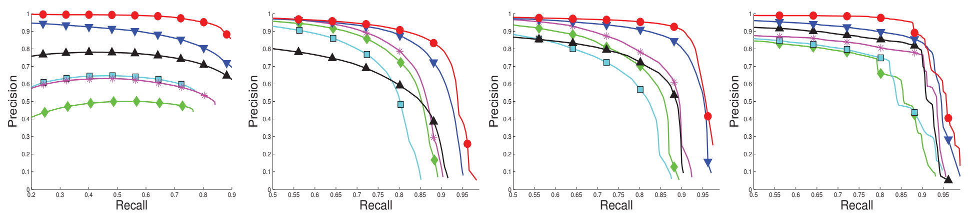
Fig. 5.4 and Fig. 5.5 show the Precision-Recall curves of the different methods, computed for different tolerance factors ρ and δ. From the figures, we notice that iterative regression consistently brings an improvement, both for centerline localization accuracy and for radial estimation. This confirms the importance contextual information to solve the problem. Qualitative results are given in Fig. 5.7 and Fig. 5.6. In particular, from Fig. 5.7 we see the advantage of using our scale-space contextual method on a Brightfield stack. Thanks to spatial context we obtain more continous centerlines, while thanks to radial context the radial estimation is smoother.



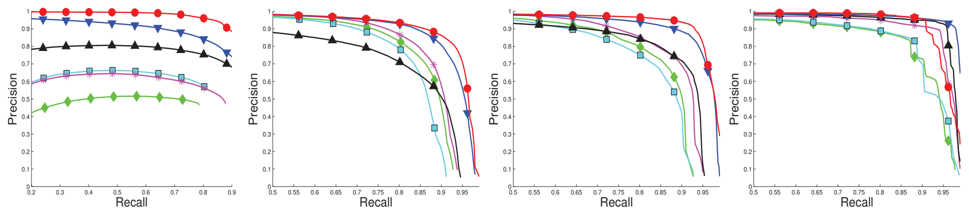
(a) Centerline precision-recall curves for $\rho = 1$.



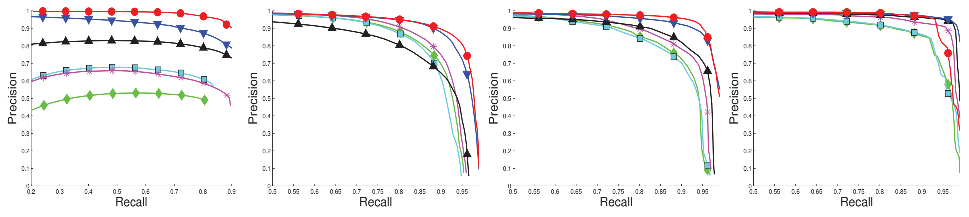
(b) Centerline precision-recall curves for $\rho = 2.0$.



(c) Centerline precision-recall curves for $\rho = 3$.

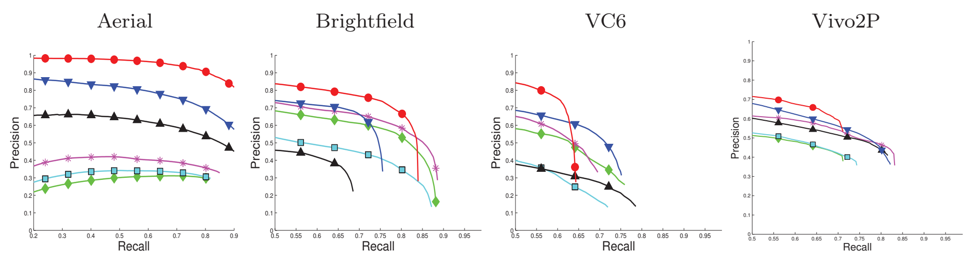


(d) Centerline precision-recall curves for $\rho = 4$.

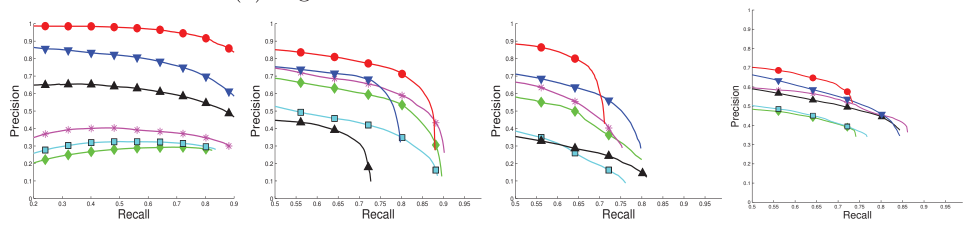


(d) Centerline precision-recall curves for $\rho = 5$.

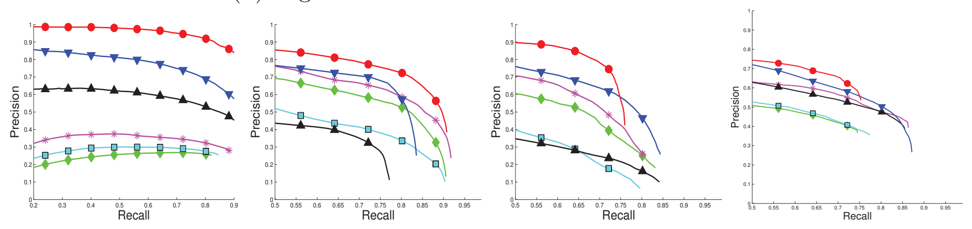
Figure 5.4: Precision-Recall curves for centerline detection for different tolerance values. Our method outperforms the others on all the datasets we considered. Using iterative regression further improves the results.



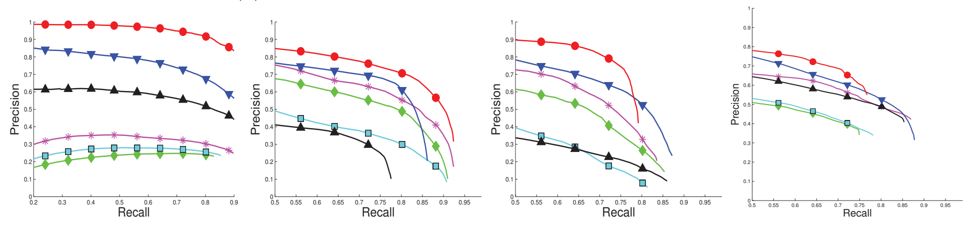
(a) Segmentation Precision-Recall for $\delta = 0.1$.



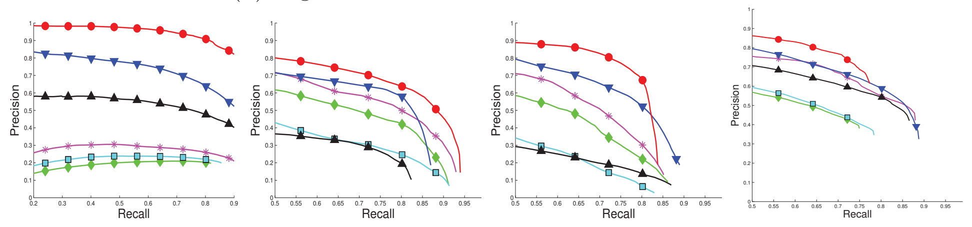
(b) Segmentation Precision-Recall for $\delta = 0.2$.



(c) Segmentation Precision-Recall for $\delta = 0.3$.



(d) Segmentation Precision-Recall for $\delta = 0.4$.



(e) Segmentation Precision-Recall for $\delta = 0.5$.

Figure 5.5: Precision-Recall curves for segmentation for different tolerance values. Our method outperforms the others on all the datasets we considered. Using iterative regression further improves the results. Legend in Fig. 5.4(a).

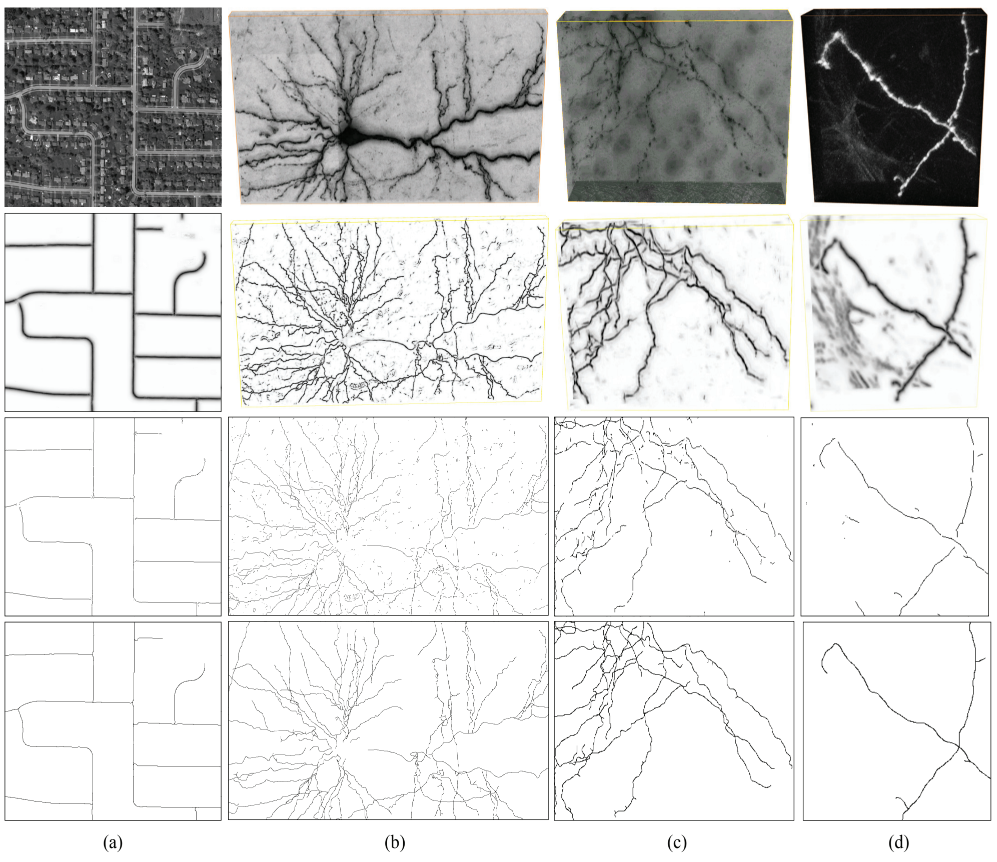


Figure 5.6: Centerline Detection Results. (a) Aerial image. (b) Brightfield image stack. (c) VC6 image stack (d) *In vivo* two-photon (Vivo2P dataset) volume. In each case, we show from top to bottom the original image, the maximum projection along the radial component of our regressor's output, centerlines detected by thresholding after Non-Maximum Suppression, and ground truth centerlines. In the last three rows, the images have been inverted for visualization purposes.

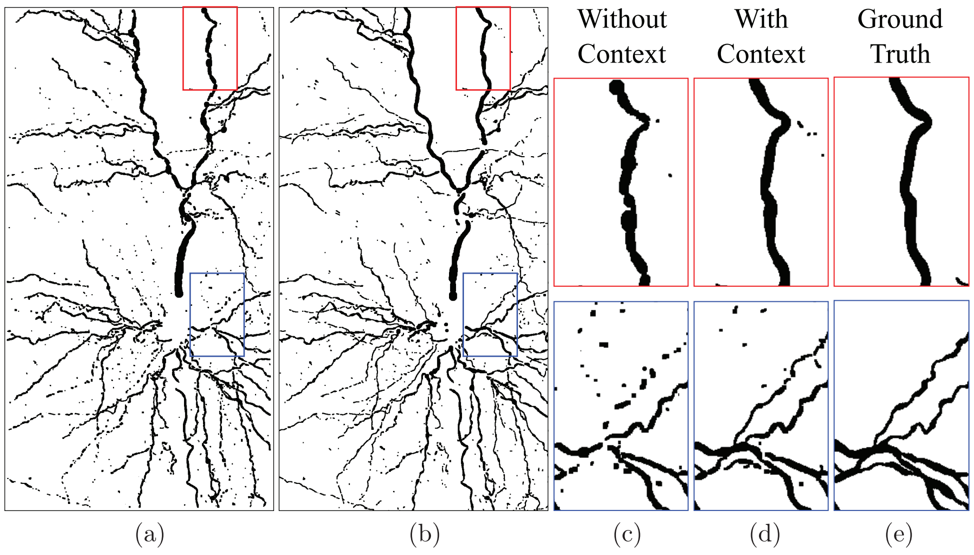


Figure 5.7: Maximum intensity projection of the segmentation results on a Brightfield stack. (a) Segmentation obtained with the local method of Chap. 3; (b) Segmentation obtained with our context-aware method; (c-e) Top row: Detail of the red top rectangle in (a,b). Thanks to the radial context used by our method, we obtain more accurate radii estimation. (c-e) Bottom row: Detail of the blue bottom rectangle in (a,b). Thanks to spatial context our method has less background detections and more continuous segmentations of the linear structures. The images have been inverted for visualization purposes.

5.5.2 Tracing Evaluation

The evaluation measures used to compare the results in the previous section are only local measures. For the problem of linear structure reconstruction it is also interesting to have a global measure able to evaluate how well a tubularity score can be used to trace linear structures.

To this end, we use in this section the tubularity scores obtained with our method and the baselines with the Fast Marching algorithm to generate paths between two points on a connected linear structure. We then evaluate how well these paths match the ground truth path using the approach proposed in [174], which we adapted to take into account also the estimation of the radius. In particular we use the overlapping measure OV and the accuracy measure AD. OV represents the ability to track the complete ground truth path, and is defined as

$$OV = \frac{TP_M + TP_R}{TP_M + TP_R + FN + FP} , \tag{5.6}$$

where TP_R and TP_M are the numbers of true positives in the reference path and the reconstructed path respectively, and FN and FP are the numbers of false negatives and false positives. In addition of the condition defined in [174] for true positives, we also take into account the radial estimation and impose that $\min(r_{\text{est}}, r_{\text{gt}})/\max(r_{\text{est}}, r_{\text{gt}}) > th$,

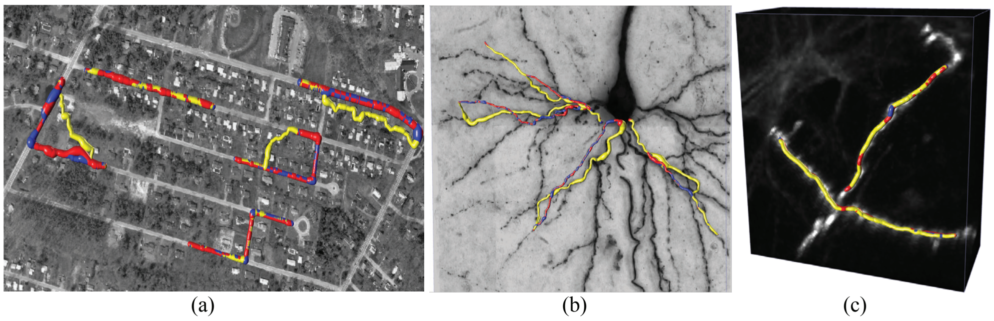


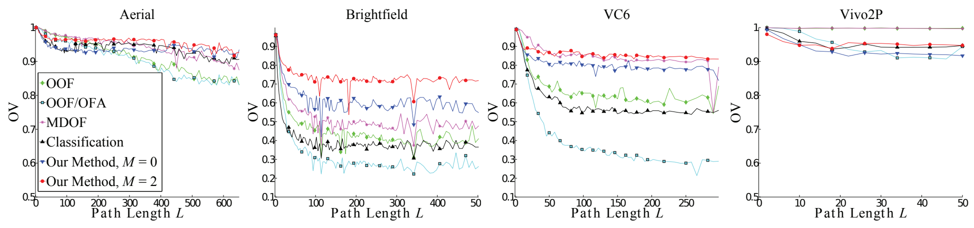
Figure 5.8: Example of random paths used in the Tracing Evaluation of Section 5.5.2. The ground truth paths are represented in blue. The paths obtained from the MDOF tubularity score are represented in yellow, while those obtained using the tubularity score returned by our method are in red. The paths obtained with our method are most of the time much closer to the ground truth paths. In particular our method is able to follow the linear structure on a longer distance even in case of complex tree topology, such as the Brightfield stack (b) or in images with many background objects, such as the Aerial image (a). In such cases, the paths returned using MDOF are partially on the background or on adjacent structures. In simpler situations, such as for the Vivo2P dataset (c), the two methods are both able to provide the correct path. Best viewed in color.

where r_{est} is the radial estimation at one point, r_{gt} the ground truth radius and $th \in [0, 1]$ is a threshold value. Setting $th = 0$, the radial estimation is not considered and the evaluation is equivalent to [174]. For $th = 1$ perfect match of the radius is required. In our experiments we set $th = 0.75$. AD is the average distance between ground truth path and the path extracted automatically. It was computed in the scale-space to again take into account the radial estimation.

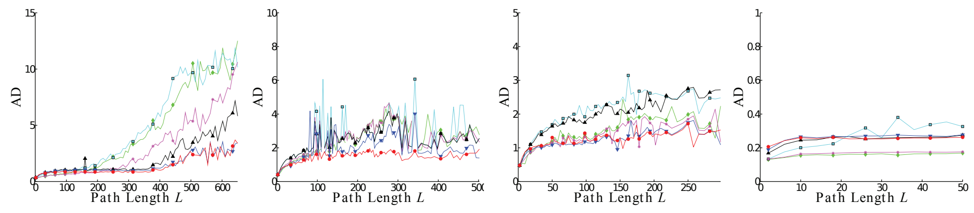
For each dataset, we randomly sampled 1000 paths of fixed length L from the ground truth, generated the paths joining the starting and ending points of these patches using the different tubularity measures and the Fast Marching algorithm and finally computed the corresponding OV and AD values.

Fig. 5.9 shows the OV and AD values as a function of the path length L. Note that as the length of the path increases our method remains more robust. Unlike the others, it also retains its accuracy. As shown in Fig. 5.8, the resulting paths follow the true linear structures over longer distances, without being disturbed by adjacent structures or background objects.

For smaller values of L, all methods perform well. The OOF-based measures even appear slightly better than the learning-based ones. This is an artifact due to the fact that the ground-truth paths have been generated using a semi-automated tracing tool that itself relies on OOF [25]. This is particularly true for the Vivo2P dataset, that only features short dentritic trees with simple topology.



(a) OV measure as a function of the path length L.



(b) AD measure as a function of the path length L.

Figure 5.9: Tracing Evaluation. Our method is more robust when used to trace linear structures. The accuracy of our method remains constant for large values of the path length L, while the performance of the other methods decreases. OV is the fraction of points on the ground truth path marked as true positives, the larger the better. AD is the average distance between ground truth path and centerlines extracted automatically, the smaller the better. On the simpler Vivo2P dataset all methods perform well; OOF-based measures appear slightly better than the learning-based ones because the ground-truth paths have been generated using a semi-automated tracing tool that relies on OOF [25].

5.5.3 Automated Reconstruction

We evaluated our approach in combination with a state-of-the-art tracing algorithm [196]. A tubularity measure is used in the first step of the algorithm to build a graph. This graph is then processed to extract the subgraph describing the linear structures. In the original implementation of [196] the tubularity measure was OOF [114]. Here we used instead the more accurate MDOF measure [194].

We used the DIADEM score [13] as an evaluation metric. It computes a similarity measure between two tree graphs and it takes into consideration the topology of the reconstruction. The results are reported in Table 5.1. Using our method yields a substantial improvement on the challenging Aerial and Brighfield datasets. On the Vivo2P dataset the difference is smaller but our method still performs slightly better. This is due to the comparative simplicity of that dataset, which contains only structures with a simple topology.

The reconstruction obtained with our method are shown in Fig. 5.10 for two Aerial images and in Fig. 5.11 for a Brightfield and a Vivo2P image stacks. From Fig. 5.10(d) we see that we are able to correct errors present in the ground truth.

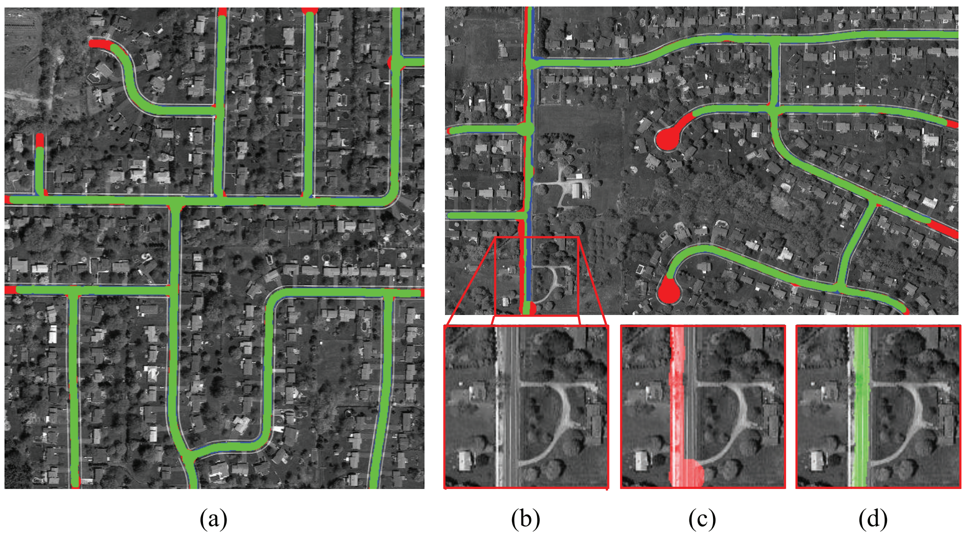


Figure 5.10: Road tracing. We used our method as a preprocessing step for a state-of-the-art tracing algorithm [196] to reconstruct road networks. The green color corresponds to true positives, red to false negatives, and blue to false positives. Most of the mistakes of our method are made at the ends of the roads. (a) An image where the roads were almost perfectly reconstructed. (b)-(d) Another image for which our method correctly recovers the centerlines and the radii (d) while the ground truth was incorrect (c) for the vertical road on the bottom-left part of the image. Best viewed in color.

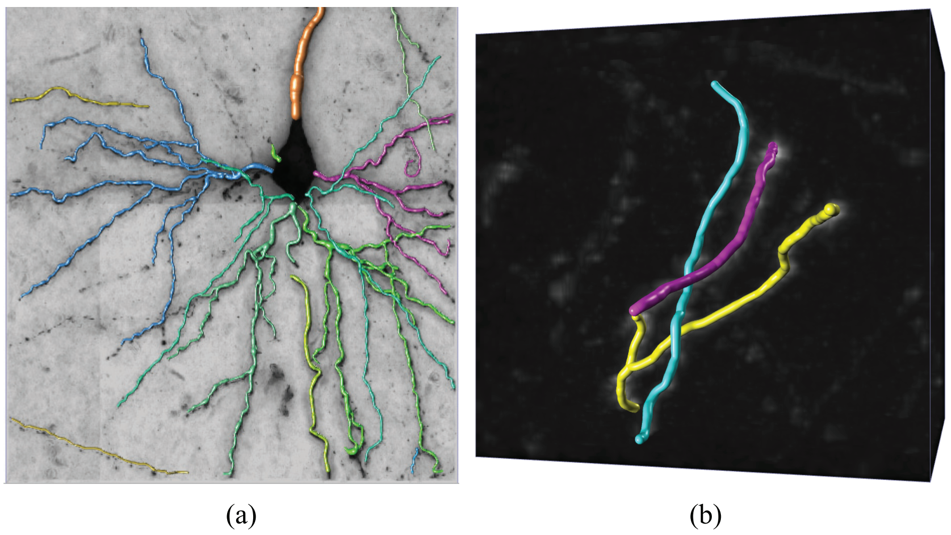


Figure 5.11: Automated neuron delineations obtained by feeding the output of our approach to the algorithm of [196]. Different colors indicate different dentritic trees. (a) Reconstruction of neurons in a Brightfield image stack; (b) Reconstruction in a Vivo2P volume. Best viewed in color.

Table 5.1: Automated reconstruction results. DIADEM scores computed on the reconstruction returned by [196] using either our method or MDOF [194] as initial tubularity measure. For the Aerial dataset, we considered only the images containing road networks having a tree topology, since the DIADEM score is not defined for generic graphs.

Method	**Aerial**	**Brightfield**	**Vivo2P**
Our Method + [196]	**0.84**	**0.60**	**0.71**
MDOF + [196]	0.75	0.56	0.70

5.6 Boundary Detection

To demonstrate how generic our approach is, we consider here the problem of boundary instead of centerline detection. As centerlines, natural image boundaries are one dimensional structures whose exact location can be uncertain, as shown in Fig. 5.12. In this example and as often the case, the local appearance of boundary pixels and of their neighbors are extremely similar. In fact, as shown in Fig. 5.12(b), different people may mark different pixels as boundary points. Moreover, large contextual information is needed to discriminate local intensity edges, such as texture or illumination changes, that do not correspond to object boundaries.

Our regression-based approach gives us a robust way to deal with this uncertainty as shown in Fig. 5.12(b) and we can enforce a single maximal response for each boundary. Moreover, by including contextual information, as described in Sec. 5.2, we are able to better discriminate local intensity changes that are not object boundaries.

In the remainder of this section, we describe how we adapt our method for boundary detection purposes and the dataset used to test its performance.

To test our algorithm we used the Berkeley BSDS500 dataset [10]: The BSDS is a standard dataset used to test boundary detection algorithms. It is composed of 500 color images. 200 are used for training, 100 for validation and the remaining 200 for testing. Ground truth is made of boundaries as drawn by several annotators.

No radial information is associated to a boundary, therefore, we train a single regressor following the single-scale approach of Sec. 5.2. Since the appearance of a boundary is often more complex than linear structures, we used 10^6 training samples and trees of depth 5 as weak learners.

Moreover, in natural images color and texture are important sources of information to detect boundaries [136]. To compute color features we converted the input RGB images to Luv color space and learned a different filter bank on each channel.

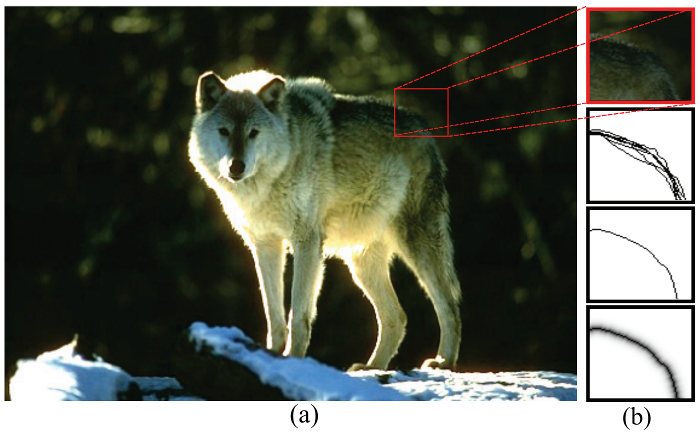


Figure 5.12: Learning to predict boundaries in natural images. (a) A training image. (b) First row: Detail of image (a). Second row: Ground truth annotated by several human subjects. All the annotations are typically used in classification based approaches. This can produce multiple responses on a boundary. Third row: Aligned ground truth obtained using [11]. Fourth row: The function d_C used in our method enforce a single detection and can model uncertainty.

To detect texture boundaries instead, we added to our feature vector a corresponding pooled version. We tried different pooling strategies and the one that worked better for us was to take the absolute value of the average over a 5×5 region. As for centerline detection, we used $M = 2$ regression iterations. Features on the score map where extracted using filters learned on the ground truth maps, as described in Sec. 5.4, in the case of the Aerial dataset.

In the BSDS500 dataset the ground truth is made of 4 to 10 different annotations. In order to compute the function d_C of Eq. (3.5), we first align the different annotations using the approach described in [11], as shown in Fig. 5.12(b). In this way, we obtain a single response for each boundary and we compute the function d_C from this binary map.

Finally, as done in [160, 53, 176], we also run our boundary detector on the test images at 3 different resolutions: half, original and double size, and then average the results [1].

For evaluation, we compare our method against state-of-the-art pixel-wise boundary detectors based on classification: Sparse Contour Gradients SCG [160] relies on gradients of sparse codes and linear SVM to classify the boundaries; Cascaded Hierarchical Model CHM [176] is an iterative method like ours, but it is based on classification. We also compare against the Structured Edge (SE) prediction approach [54], which is a patch-wise method based on structured random forests [107]. It predicts for every pixel location a binary patch corresponding to the edge profile at that point. Finally, we also consider the

[1]The code used for and the results obtained with our experiments are publicly available at: `http://cvlab.epfl.ch/software/centerline-detection`.

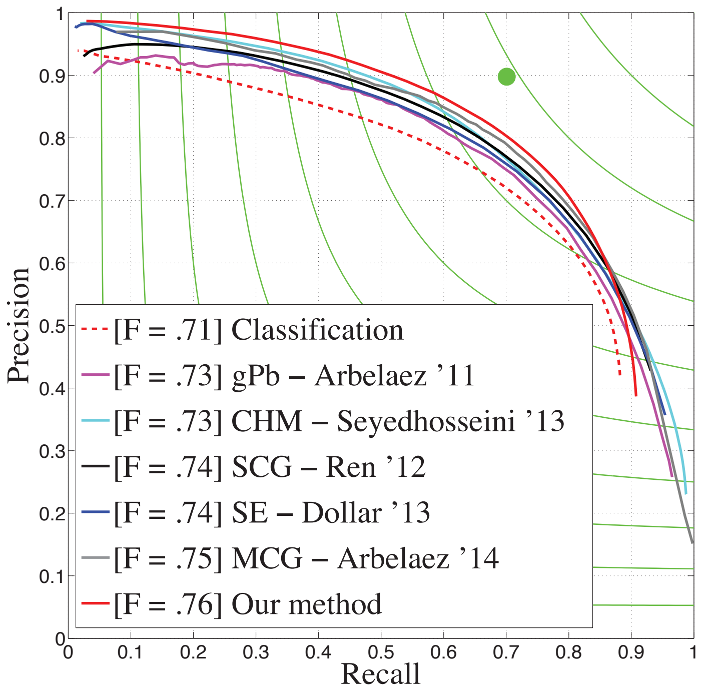


Figure 5.13: Precision-Recall curves on the BSDS500 dataset for different boundary detection methods. In brackets is shown the F-measure computed with a fixed threshold for every image (ODS [10]). Our method outperforms the baselines also on this task. The advantage of our method is mainly given by the regression formulation. Retraining our model using the binary ground truth to classify the boundaries deteriorates the performance by 5% (red dashed line in the graph).

gPb-ucm approach of [10] and MCG [11] which segment the images in different regions and also include a globalization step to make prediction more accurate.

The Precision-Recall curves obtained using the standard Berkeley benchmark [10] are shown in Fig. 5.13. Our method is more accurate than state-of-the-art pixel-wise techniques and also than the patch wise approach of [54], Moreover, unlike gPb [10], SCG [160] and MCG [11], our method does not include any globalization step.

The advantage of our method is mainly given by the regression approach. In fact, by applying our method with a classification-based formulation gives significantly worse results. This Classification results are shown in Fig. 5.13, they were obtained with the same implementation as for our regression method, using the same image features and the same number of Auto-Context iterations—the only difference was the objective function minimized during training: The classifier was trained to classify the pixels lying on boundaries versus the other pixels by minimizing the exponential loss, rather than regressing the distance transform.

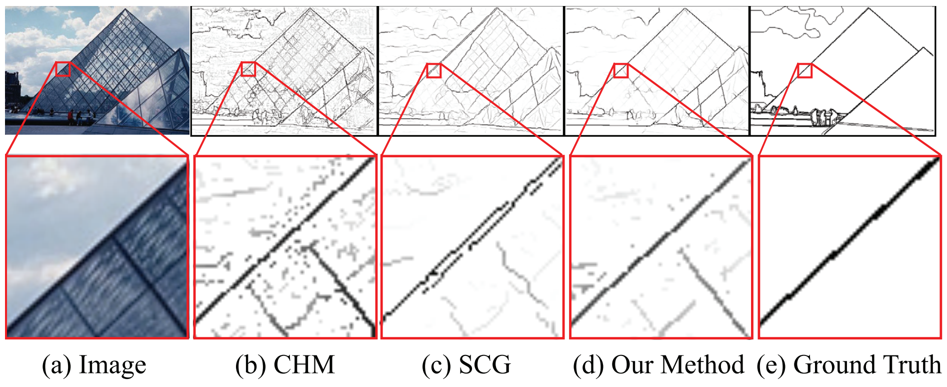


Figure 5.14: Compared to classification based approaches, the boundaries detected by our method are better localized and less noisy on the background. In particular, from the image details in the second row, we see that thanks to our regression formulation we avoid multiple detections of a boundary.

The boundaries detected by the different methods on some test images are shown in Fig. 5.15. When we compare the qualitative results obtained with our method against the two state-of-the-art pixel-wise classification-based detectors, SCG [160] and CHM [176], we see that our approach can avoid double responses and multiple detections (Fig. 5.14). The gPb-ucm [10] and SE [53] methods do not have this problem since they obtain boundaries after segmenting an image or an image patch in different regions.

However, our method is still more accurate and also gives a general framework that is not limited to the contour detection problem. Moreover, as a possible extension of our method, we can naturally incorporate information about the strength of a boundary [11, 129] by modifying the shape of the distance function we want to learn.

Despite great efforts and the use of powerful learning methods [26, 177, 91], the accuracy of local and context-aware boundary detection algorithms has remained far behind the average human performance (green dot in Fig. 5.13).

Only very recently, the method proposed in [211], could make a qualitative leap and closely approach human performance. The main difference of this approach, compared to previous ones, is its global nature. In fact, following the taxonomy of Chap. 2, this method is both global, by taking as input the entire image, and also image-wise prediction-based. Thanks to this large contextual information, the method is able to discriminate object boundaries which require high-level semantic information. This underlines once more the importance of context in solving fundamental Computer Vision problems.

We discuss in more detail the method of [211] in Chap. 7, where we consider how to integrate global information also in our algorithm.

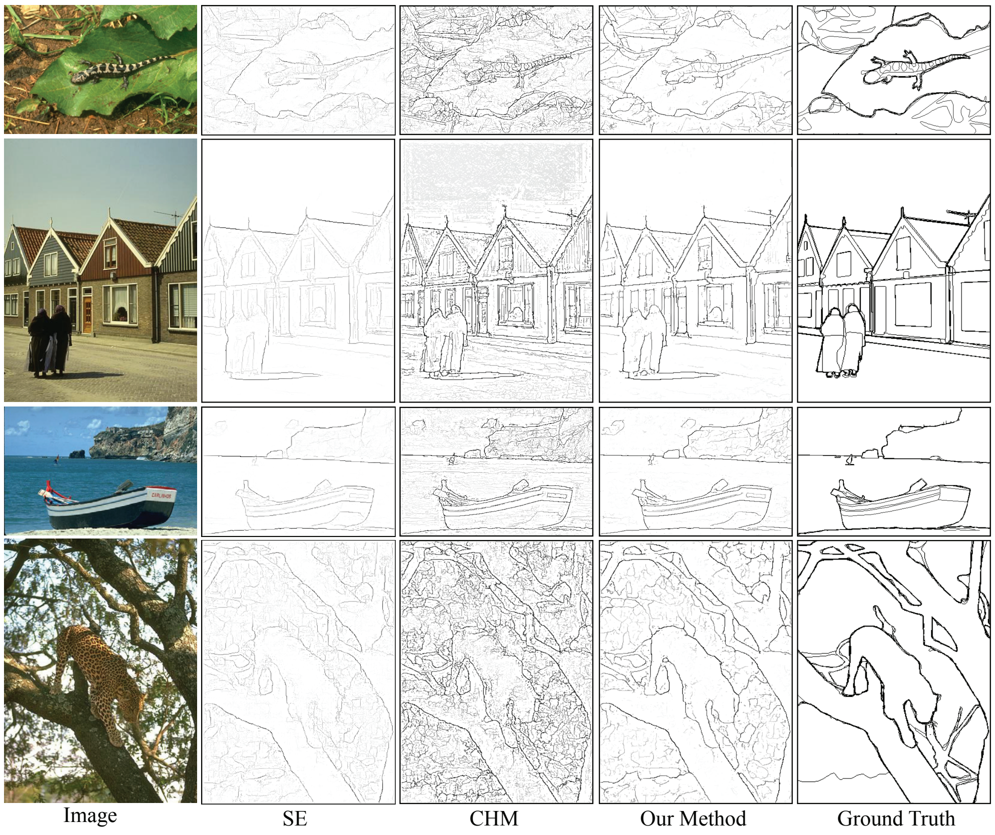


Figure 5.15: Boundary detection results. Our method is able to capture finer details, and is also more robust to false edges. See for example the lizard in the top image or the front paw of the leopard on the bottom.

Conclusion

In this chapter, we have extended our regression-based approach to take advantage of contextual information. Thanks to a scale-space iterative regression algorithm, we were able to correct the mistakes and improve the performance of the local formulation of Chap. 3. Moreover, we showed that the output of our method can be used in combination with tracing algorithms requiring a scale-space tubularity measure as input, increasing accuracy also on this task. Finally, our approach is very general and applicable to other linear structure detection tasks. For example, we obtained an improvement over previous works when training it to detect boundaries on natural images.

However, one of the main limitation of our approach is that it is based on pixel-wise predictions. Since the response of every pixel in the image is computed independently, there is no guarantee that the output of our method will be smooth. Small displacement in the input image features can abruptly change the output of the regressor. As a consequence, discontinuities and topological inconsistent results are still present in the regressor output, and this is independent on the amount of contextual information we consider in input.

In the next chapter, we consider a patch-wise techniques which allows us to induce smoothness and topological consistency on the regressor output. The method efficiently approximates the projection of the score map onto the set of admissible ground truth images. Moreover, we provide sufficient conditions under which the projection is exact.

CHAPTER 6

Efficient Projection onto the Set of Elongated Structures for Accurate Extraction

In the previous chapter, we have shown how to include contextual information in our regression-based method. Thanks to a scale-space iterative method, we could efficiently take advantage of both spatial and radial context. This led to an improvement of centerline localization accuracy and radii estimation. Moreover, we have also proven the generality of our approach by using it in conjunction with a tracing algorithm or by applying it to boundary detection, showing an improvement also in these cases.

However, the method of Chap. 5 essentially classifies individual locations and does not explicitly model the strong relationship that exists between neighboring ones. As a result, isolated erroneous responses, discontinuities, and topological errors are still present in the resulting score maps. This problem does not depend on the amount of contextual information considered as input to the regressor, but on its pixel-wise nature. In fact, since no spatial smoothness constraints are explicitly imposed on the regressor output, small changes in the input feature map can produce arbitrary different output values.

Up to a point, these problems can be mitigated by relying on structured learning to model correlations between neighbors, as in [54]. In this chapter, we show that a better way is to first compute the score map using an appropriately trained pixel-wise regressor, as in Chap. 5, and then systematically replace pixel neighborhoods by their nearest neighbors in a set of ground truth training patches.

This is in the spirit of algorithms for image denoising and inpainting that search for nearest neighbors within the image itself [44, 43, 128]. It is also closely related to the approach of [74] that improves boundary images by finding nearest neighbors using a distance defined in terms of descriptors extracted by a Convolutional Neural Network. By contrast, in our method, we compute distances in terms of the patches themselves and we will show that it improves both performance, especially near junctions, and generality.

Following the taxonomy of Chap. 2, the algorithm of this chapter falls in the category of patch-wise prediction methods. However, we will show that, under certain assumptions, our patch-wise formulation induces global consistency on the whole image. In effect, by assuming that the structure of all admissible ground truth images is well represented by the set of training patches, it can be formally shown that our method is equivalent to projecting the score map onto the set of all admissible ground truth maps.

In short, our algorithm induces global spatial consistency on the regressor score map and improves its performance. Our method is general, in the sense that it is not limited to the centerline detection problem or to the regression formulation of the Chap. 3. To prove its accuracy and generality, we apply it to challenging datasets in four different domains and show that it compares favorably to state-of-the-art methods.

The rest of the chapter is organized as follows. In Sec. 6.1, we discuss the limitations of pixel-wise prediction algorithms and show in particular some failure cases of our context-aware regression approach of Chap. 5. In Sec. 6.2, we improve our method by introducing our patch-wise projection algorithm and in Sec. 6.3 we give conditions under which our algorithm is equivalent to projecting the score map onto the set of ground truth images. We also show the validity of these conditions by applying our method to a synthetic dataset. In Sec. 6.4 we discuss some implementation details and finally, in Sec. 6.5, we demonstrate the versatility of our approach by applying it on four very different problems.

Part of the content of this chapter has been previously published in [180].

6.1 Topological Inconsistency of Pixel-Wise Approaches

As discussed in Chap. 2, methods that rely on statistical classification techniques currently deliver the best results for boundary, centerline, and membrane detection. As shown in the previous chapters, reformulating the problem in terms of regression, performs best for centerline and boundary detection, and we will demonstrate here that it performs equally well for membrane detection.

More specifically, the algorithm of Chap. 5 involves training regressors to return distances

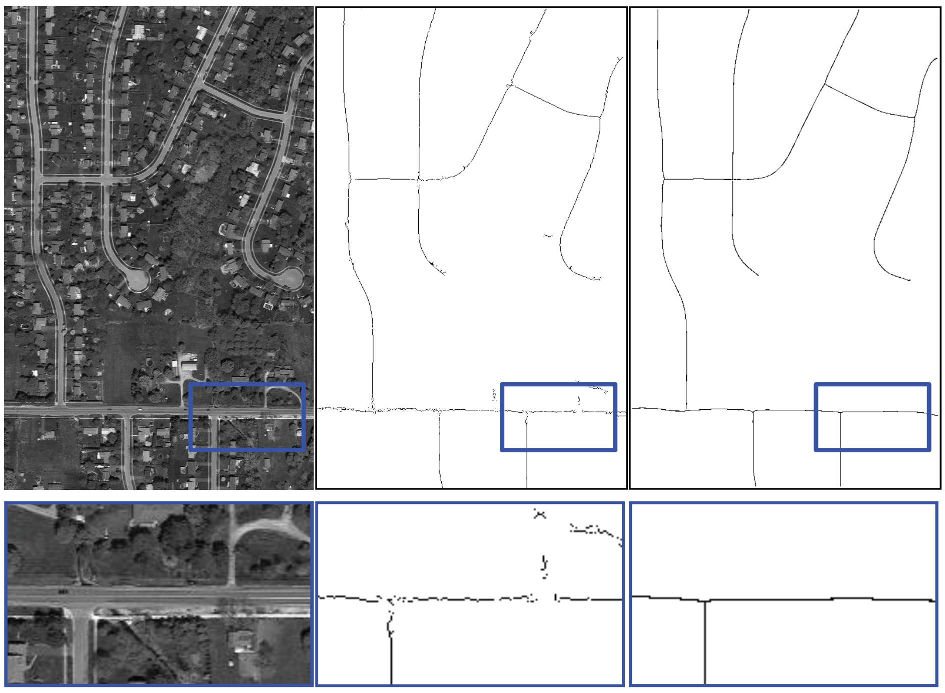


Figure 6.1: Limitation of pixel-wise approaches. Pixel-wise classifiers and regressors reach state-of-the-art performance in several Computer Vision tasks. However, the response of such methods does not take into account the very particular structure and the spatial relation present in the ground truth images. (a) An aerial road image. For this problem the ground truth is composed by a continuous 1-D curve. (b) Output of our method of Chap. 5. Since this method is based on pixel-wise regression, its output has discontinuities on the centerlines and isolated responses on the background. (c) The output of our method is obtained by projecting the patches of the score image of (b) into the closest ground truth patches from the training images. In this way the structure of the ground truth patches is transferred to the score image, resulting in a provably correct global spatial structure.

to the closest centerline in scale-space. In this way, performing Non-Maximum Suppression on their output yields both centerline locations and corresponding scales. Although this has proved very effective, like for all other pixel-wise techniques that do not incorporate any *a priori* geometric knowledge that may be available, this approach can easily result in topological mistakes, as shown in Fig 6.1.

In fact, even though the feature vector extracted from neighboring pixels have overlapping domains, small changes in the input vector, can produce a completely different output of the regressors. This phenomenon is even more pronounced for classification-based approaches, which are trained to produce a discontinuous function, returning value one for pixels on the centerline and zero for pixels immediately next to it. This problem is mitigated by our regression formulation. However, since no explicit constraint is given during training, some discontinuities in the output score map are still present.

As a consequence, the output score map of pixel-wise methods is often topologically inconsistent, meaning that it contains errors such as isolated responses, gaps in the linear structure and missing junctions, which never occurs in the ground truth images (Fig. 6.2). As discussed in Sec. 1.1.4 and Sec. 5.5.2, these kind of errors, have small effect when computing pixel-wise evaluation metrics, such as those of Sec. 3.6. However, they can become problematic when trying to reconstruct the connectivity of the network.

In this chapter, we demonstrate that we can correct the errors made by pixel-wise methods by projecting their score map onto the set of distance transforms corresponding to the kind of structures we are trying to reconstruct. This results in a technique that is more accurate and more widely applicable than the method of the previous chapter. Furthermore, it is generic in the sense that it is applicable to other methods returning a score map, such as [54, 38].

6.2 Efficient Projection using Patch-Wise Nearest Neighbors

In this section we describe how to improve the results obtained with a pixel-wise regressor by projecting patches extracted from the score map of the regressor into a set of ground truth patches.

The central element of our approach is to project the distance transform produced by pixel-wise regression, as described in the previous chapters, onto the set of all possible ones for the structures of interest. Since this set is too large to be computed in practice, we first propose a practical computational scheme. Then, in next section, we introduce formal conditions under which the projection is exact.

Before describing our method, we start by introducing the notation used in this chapter and by briefly recalling the regression formulation of Chap. 3.

Centerline Detection

Let $I \in \mathbb{R}^N$ be an image containing linear structures, where N is the number of pixels in the image, and let Y be the corresponding binary ground truth image, such that $Y(p) = 1$ if pixel p is on a centerline and $Y(p) = 0$ otherwise.

Let $f(p, I)$, the feature vector extracted from a local neighborhood around pixel p in image I. As explained in Chap. 3, learning a classifier mapping $f(p, I)$ to $Y(p)$ can be

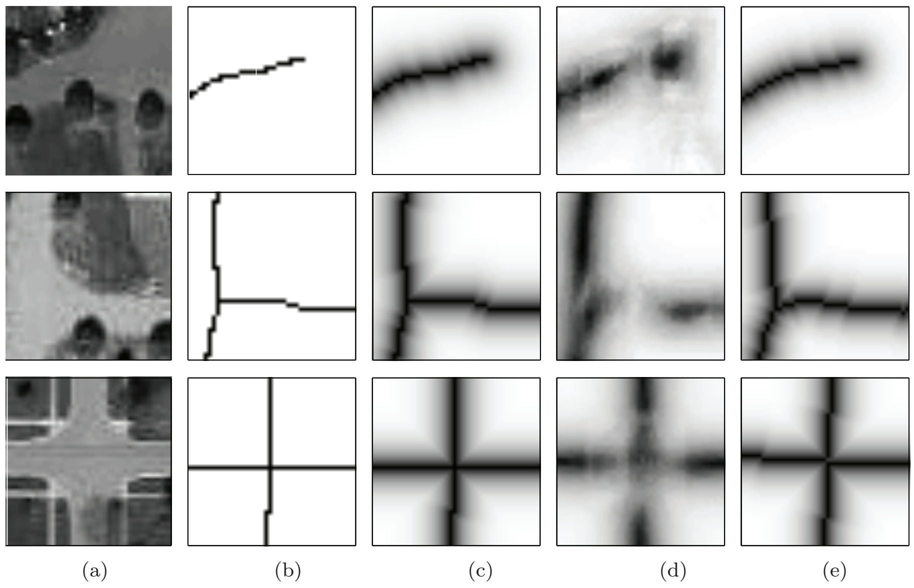


Figure 6.2: Centerline detection as a patch-wise regression problem. (a) Original image patch; (b) Centerline ground truth; (c) Distance function of Eq. (6.1) proposed in Chap. 3; (d) The response of a pixel-wise regressor trained to predict the function in (c) is discontinuous and returns topologically incorrect results, also when iterative regression is applied, as in Chap 5. (e) Nearest Neighbors of the score patches in (d), found in the training set. In our method we apply Nearest Neighbors search to the regressor output and take advantage of the particular structure of ground truth patches to correct its mistakes.

difficult in practice. To address this difficulty, we have replaced the binary ground truth Y by the modified distance transform of Y

$$d_C(p) = \begin{cases} e^{a(1-\frac{\mathcal{D}_Y(p)}{d_M})} - 1 & \text{if } \mathcal{D}_Y(p) < d_M \\ 0 & \text{otherwise} \end{cases} , \tag{6.1}$$

where $\mathcal{D}_Y$ is the Euclidean distance transform of Y.

Fig. 6.2(c) shows examples of function d_C computed on small patches. Learning a regressor to associate the feature vector $f(p, I)$ to $d_C(p)$ induces a unique local maximum in the neighborhood of the centerlines. This approach is more robust to small displacements and returns centerlines that are better localized compared to classification-based methods.

To learn the regressor we apply the GradientBoost algorithm [82], which approximates $d_C(\cdot)$ with a function $\varphi(\cdot)$ of the form $\varphi(q) = \sum_{t=1}^{T} \alpha_t h_t(q)$, where $q = f(p, I)$ denotes the input feature vector. In Chap. 3, we described in more detail how $\varphi(\cdot)$ is learned.

In addition, to include contextual information, inspired by the Auto-Context algorithm [193] for image segmentation, in Chap. 5 we adopted an iterative technique which improved the accuracy of the prediction. This was achieved by using the score map $\varphi(\cdot)$ to extract a new set of features that are added to the original ones to train a new regressor.

Boundary and Membrane Detection

The method described above, designed for centerline detection, extends naturally to boundary and membrane detection, as also described in Sec. 5.6. As centerlines, boundary in 2D images and membranes in 3D image stacks are elongated structures of codimension 1 and there are substantial ambiguities in their exact location.

Therefore, and as before, we can replace the binary ground truth, provided for such problems, by the distance transform of Eq. (6.1). By training a regressor to associate feature vectors to the distances to the boundaries, we can obtain the boundaries from the score map returned by the regressor by Non-Maximum Suppression. The distance function is computed 2D for boundaries and 3D for membranes.

6.2.1 Improving the Distance Function by Patch-Wise Projection

In this section, we show how we can improve the score map returned by a pixel-wise regressor or classifier, by projecting patches of the score map onto the set of ground truth patches.

Given an image I and corresponding binary ground truth Y, let dY be the image obtained by applying function d_C of Eq. (6.1) to every pixel of Y. Since it corresponds to pixels belonging to specific structures, Y is constrained to have well defined geometric properties. For example, in the case of centerlines or boundaries in images, Y is composed of 1-dimensional curves, while for boundaries in 3D volumes, Y is a 2D surface. This means that the set of all admissible ground truths forms a low-dimensional set in the set of all binary images. Similarly, the set of images dY is low dimensional in the set of real valued images. We will denote the set of images dY by $\mathcal{M}_N$.

Let X be the score map obtained by applying a regressor φ to each pixel of an input image I. Ideally we would like X to be an element of $\mathcal{M}_N$, so that it is guaranteed to be geometrically correct. However, this is not true in general. Fig. 6.2(d) shows typical errors committed at critical points, such as T-junctions. This is a standard problem with many edge detectors, such as the Canny detector.

In theory, one way to avoid this problem is to project X into $\mathcal{M}_N$, which is equivalent to finding the element of $\mathcal{M}_N$ closest to X,

$$\Pi_N(X) = \operatorname*{arg\,min}_{dY \in \mathcal{M}_N} \|dY - X\|^2. \tag{6.2}$$

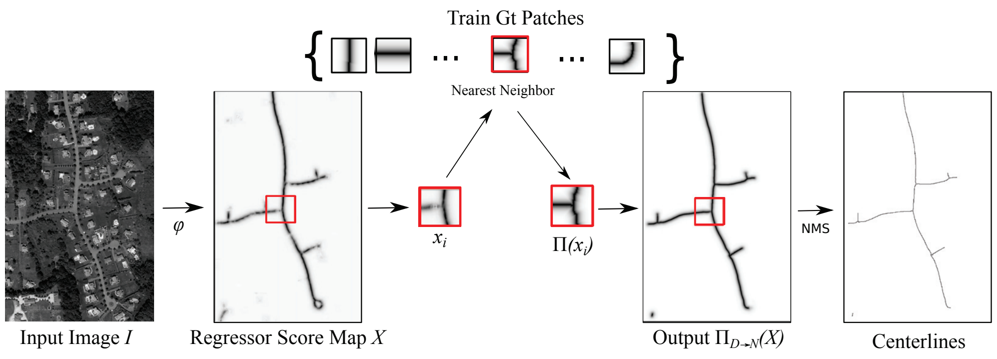


Figure 6.3: Method overview. A score map X is obtained from image I by applying a regressor φ trained to return distances to the centerlines. Every patch x_i of size D in X is projected onto the set of ground truth training patches, by nearest neigbor search. The projected patches $\Pi_D(x_i)$ are averaged to form the output score map $\Pi_{D\to N}(X)$. Centerlines are obtained by Non-Maximum Suppression.

In practice, however, $\mathcal{M}_N$ is not known or too large to be sampled exhaustively. Therefore, $\Pi_N(X)$ can not be computed directly.

As shown in Fig. 6.3, our solution is to approximate it by projecting small patches of X onto the set of ground truth train patches.

Formally, let $\mathcal{M}_D = \{y_k\}_{k=1}^{K}$ be the set of training patches of size D, extracted from local neighborhoods $\mathcal{N}_D$ in the ground truth training images. For each pixel p_i, $i = 1, \ldots, N$ in the score image X, let $x_i = X(\mathcal{N}_D(p_i))$ be the squared neighborhood of size D around pixel p_i in image X.

For every i, we consider the projection of x_i onto $\mathcal{M}_D$, given by

$$\Pi_D(x_i) = \operatorname*{arg\,min}_{y\in\mathcal{M}_D} \|y - x_i\|^2. \tag{6.3}$$

Fig. 6.2(d) shows examples of nearest neighbors for three score patches. We then average all these projections to obtain a new score image $\Pi_{D\to N}(X)$.

More precisely, given the set of projected patches $\{\Pi_D(x_i)\}_{i=1}^{N}$, we take the pixel values of the new image $\Pi_{D\to N}(X)$ to be

$$\Pi_{D\to N}(X)(p) = \frac{1}{R} \sum_{i:p-p_i\in\mathcal{N}_R(p)} \Pi_D(x_i)(p - p_i), \tag{6.4}$$

where $R \leq D$ is the size of the neighborhood used for averaging and where we take $\Pi_D(x_i)$ to be centered at zero, with $\Pi_D(x_i)(p - p_i)$ the value of $\Pi_D(x_i)$ at $p - p_i$.

The image $\Pi_{D\to N}(X)$ obtained in this way is an approximation of $\Pi_N(X)$. In the next section, we introduce sufficient conditions under which $\Pi_N(X) = \Pi_{D\to N}(X)$ and we provide a formal proof in the Appendix A.

6.3 Projection onto the Set of Elongated Structures

In the previous section, we have seen how to use the projection of the score map patches to produce a new score map, approximating the projection of the original one onto the set of ground truth images $\mathcal{M}_N$. In this section, we ask ourselves under which conditions this approximated projection is exact, *i.e.* if it belongs to the set $\mathcal{M}_N$. We give sufficient conditions in Sec. 6.3.1, while, in Sec. 6.3.2, we illustrate the validity of these conditions by applying our method to a synthetic dataset.

6.3.1 Equivalence of $\Pi_{D\to N}(X)$ and $\Pi_N(X)$

In this section we state under which conditions the output $\Pi_{D\to N}(X)$ of our method is equivalent to the projection $\Pi_N(X)$ of the score image X into the set of all admissible ground truth images $\mathcal{M}_N$. For this, we introduce two fundamental properties, which we name *completeness* and *consistency*. The first one is a characterization of the set $\mathcal{M}_N$ in terms of the patches in $\mathcal{M}_D$; while the second one is a condition of smoothness of the projection Π_D on the score map X.

More precisely, the two properties are:

(i) *Completeness* of $\mathcal{M}_D$ in $\mathcal{M}_N$: The training set of patches $\mathcal{M}_D$ is composed of all admissible ground truth patches that can be extracted from images in $\mathcal{M}_N$. Moreover, averaging patches of $\mathcal{M}_D$ that coincide for overlapping pixels, gives an image of $\mathcal{M}_N$;

(ii) *Consistency* of Π_D on X: For two patches x_i and x_j, extracted from overlapping neighborhoods $\mathcal{N}_D(p_i)$ and $\mathcal{N}_D(p_j)$ in image X, their projections $\Pi_D(x_i)$ and $\Pi_D(x_j)$ coincide for all pixels in averaging intersection area of size R of $\mathcal{N}_D(p_i) \cap \mathcal{N}_D(p_j)$.

We formalize these concepts in the Appendix A, where we also prove that under these conditions our method amounts to project the score map X into the ground truth set $\mathcal{M}_N$.

More precisely we prove the following

Theorem 1. *If $\mathcal{M}_D$ is complete in $\mathcal{M}_N$ and if $\{\Pi_D(x_i)\}_{i=1}^N$ is consistent, then*

$$\|X - \Pi_{D\to N}(X)\|^2 = \|X - \Pi_N(X)\|^2. \tag{6.5}$$

Moreover, if the function $F_X(Y) = \|Y - X\|$ has a unique minimum in $\mathcal{M}_N$, we have

$$\Pi_N(X) = \Pi_{D\to N}(X). \tag{6.6}$$

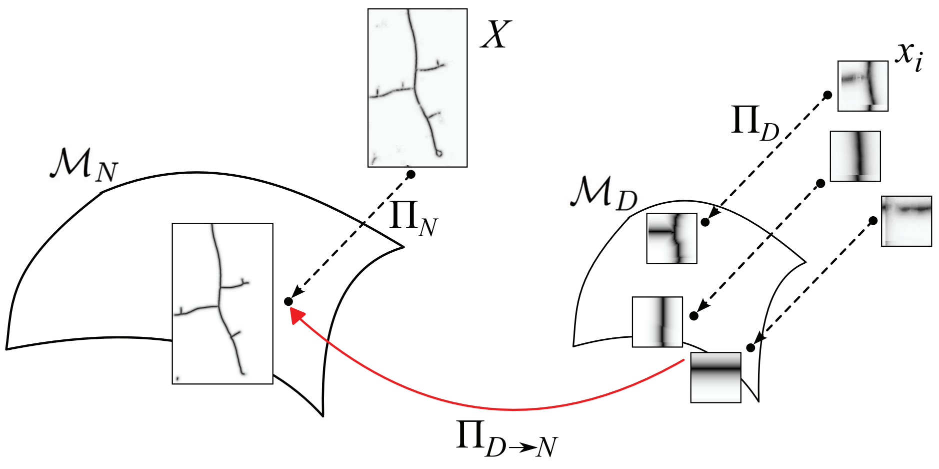


Figure 6.4: The output $\Pi_{D\rightarrow N}(X)$ of our method can be seen as a projection of the score map X into the set of admissible ground truth images $\mathcal{M}_N$. This is achieved by projecting small patches x_i of X into the set of ground truth patches $\mathcal{M}_D$ and then averaging the results to obtain $\Pi_{D\rightarrow N}(X)$.

Fig. 6.4 illustrates this equivalence. Intuitively, this means that the output of our method $\Pi_{D\rightarrow N}(X)$ is the best approximation of X in the space of ground truth images $\mathcal{M}_N$. As a consequence, $\Pi_{D\rightarrow N}(X)$ has the same geometrical properties of the images in $\mathcal{M}_N$.

Intuitively, property (i) states that the set $\mathcal{M}_N$ can be generated by "gluing" together small patches extracted from images in $\mathcal{M}_N$, and that $\mathcal{M}_D$ contains all of these patches. For example, we can suppose this to be true in the case of simple linear structures. In fact, in this case, the corresponding ground truth images can be formed by gluing together elementary line-like structures, such as straight segments, junction patches, corners, etc., provided that the glued patches are equal when overlapping.

The second property (ii), intuitively means that for small displacements of the score map patches x_i, their projection $\Pi_D(x_i)$ does not change abruptly. We can assume this to be true if X is regular enough and if the size D is large enough to capture characteristic shapes of the ground truth images.

In practice, these conditions will rarely be strictly satisfied. However, we also show in the Appendix A that, by relaxing them and assuming only approximated projections, the error made by our algorithm is within a given bound to the optimal solution. This bound can be estimated from the error committed by the projections on the patches $\Pi_D(x_i)$ and the size of our training set compared to the set all admissible training patches.

From a practical point of view, from properties (i) and (ii), we can immediately observe that there is a trade-off in the choice of the size D of the patches. In fact, smaller values of D, makes it easier to gather the complete set of admissible patches $\mathcal{M}_D$ and then to satisfy condition (i). However, with larger patches, the intersection $\mathcal{N}_D(p_i) \cap \mathcal{N}_D(p_j)$ in (ii) for two neighboring pixels p_i and p_j is larger and therefore it is more likely that the corresponding projections $\Pi_D(x_i)$ and $\Pi_D(x_j)$ are equal in the averaging region of size R.

We also remark that computing the projections in terms of the regression function d_C of Eq. (6.1), makes our algorithm more robust to small variations of the shape of the patch, compared to computing the projection using the binary ground truth. In fact, nearest neighbor search in a binary space presents many equivalent local minima and suffers from the thick boundary problem [191]. This problem is avoided when using a smooth function, like d_C, to compute the nearest neighbors. This is an additional advantage of our regression formulation.

In the next section, we apply our method to a synthetic dataset and show empirically the validity of the properties stated above and the equivalence of the projections $\Pi_{D \to N}(X)$ and $\Pi_N(X)$.

6.3.2 Synthetic Example

In this section, we apply the method described in Sec. 6.2.1 to a set of synthetic images. We show that, when the conditions of Sec. 6.3.1 are satisfied, our method can reconstruct a test image exactly. We study the behavior of the method when adding different types of noise to the test image. In particular, we investigate the influence of parameter D on the reconstruction error and empirically observe the trade-off involved in the choice of D, which was already derived from the theoretical analysis of Sec. 6.3.1.

The synthetic dataset we consider is shown in Fig. 6.5. It is composed of three train images of size 200×100 pixels and a test image of size 150×200 pixels. The images are binary and contain simple linear structures at 4 different orientations. The lines cross in different ways, forming corners, X- and T-junctions.

In order to apply our method, we compute the function d_C of Eq. (6.1), with parameters $a = 6$ and $d_M = 7$. Then, we normalize them in the range $[0, 1]$. We denote with Y_{te} the binary test image and with dY_{te} the image obtained by applying d_C to every pixel in Y_{te}. Similarly, Y_{tr_i} and dY_{tr_i}, for $i = 1, 2, 3$ indicate the ground truth images of the train set. In order to avoid boundary effects, we pad all the images with zeros before applying the distance transform. For a given value of D, the set $\mathcal{M}_D$ is created by sampling all the pixel locations in the train images. We consider squared patches of odd size, so that $D = (2\delta+1) \times (2\delta+1)$ and we take the averaging window size R to be $R = (2\rho+1) \times (2\rho+1)$,

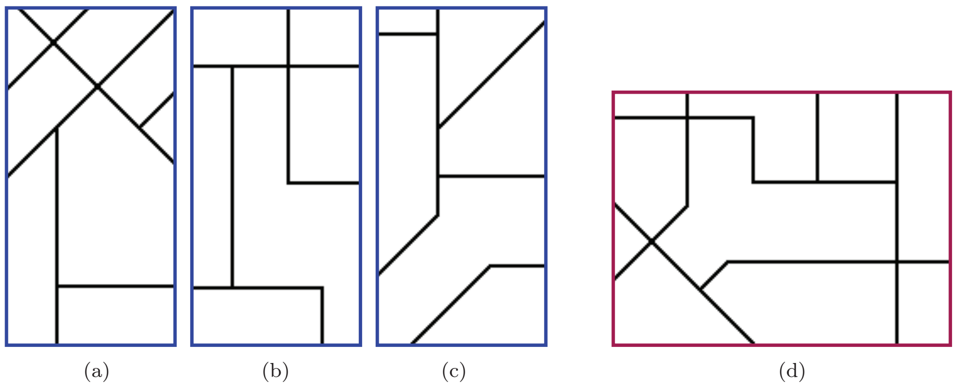


Figure 6.5: Synthetic dataset used in the experiments of Sec. 6.3.2. (a-c) Training images; (d) Test image. The centerlines have double thickness for visualization purposes.

with $\rho = \mathrm{round}(\delta/3)$.

We will create a score map X by corrupting dY_{te} with different levels and different types of noise. X emulates the output of a method which aims to approximate dY_{te}. Then, we show that, when the conditions of Sec. 6.3.1 are satisfied, we are able to recover dY_{te} exactly by applying the projection method of Sec. 6.2.1 to image X. For brevity, we will indicate the output of our method $\Pi_{D\to N}$, instead of $\Pi_{D\to N}(X)$.

In order to compute the error of the method, we consider the Mean Square Error (MSE) between the ground truth image dY_{te} and the reconstruction returned by our method $\Pi_{D\to N}$, defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\Big(dY_{\mathrm{te}}(p_i) - \Pi_{D\to N}(p_i)\Big)^2, \tag{6.7}$$

where N is the number of pixels in the test image[1].

MSE as a Function of the Patch Size

In the first set of experiments, we study the influence of the patch size on the algorithm. We start by adding Gaussian noise to the test image dY_{te}. We use Gaussian noise with mean equal to zero and standard deviation equal to 0.1. The resulting image X is shown in the top left corner of Fig. 6.6.

We apply our method for integer values of δ between 0 and 17. We repeat the experiments 5 times and average the errors obtained at the different repetitions. The results are plotted in Fig. 6.8(a). We observe that for δ in the range 8 to 12, the error is, up to numerical errors, equal to zero, showing that our method perfectly recovers the ground truth dY_{te}.

[1]Notice that if $\Pi_N(X) = dY_{\mathrm{te}}$, we have $MSE = 0$ if and only if $\|\Pi_{D\to N}(X) - \Pi_N(X)\|^2 = 0$, which is the condition of equivalence of Sec. 6.3.1.

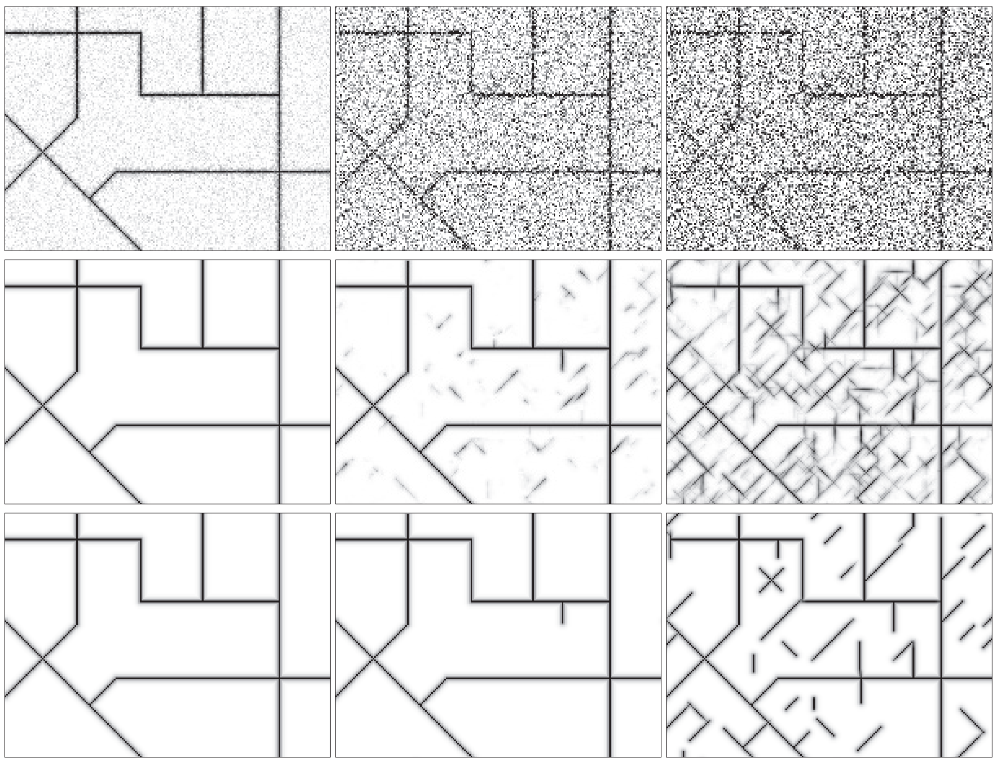


Figure 6.6: First row: synthetic score maps used in our experiments for different level of Gaussian noise, with size $\delta = 9$. Second row: results obtained when applying our algorithm to the score maps of the first row. Third row: Results obtained by iteratively applying our algorithm. (a) $\sigma = 0.1$; (b) $\sigma = 0.5$; (c) $\sigma = 0.75$.

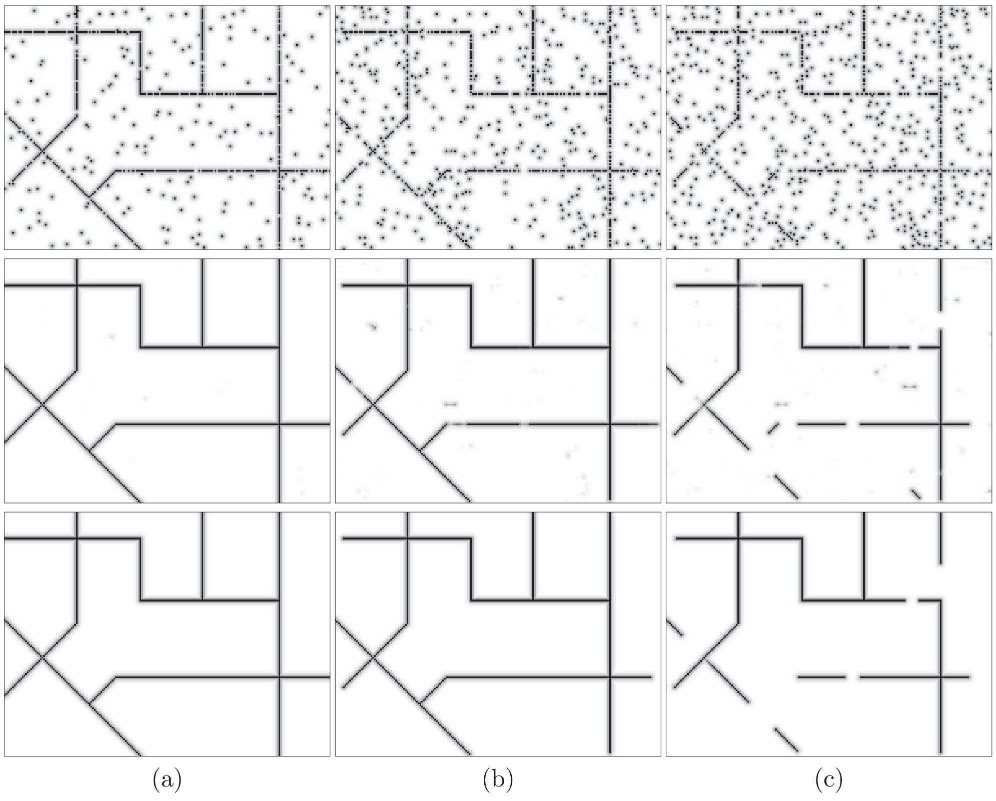


Figure 6.7: First row: synthetic score maps used in our experiments for different level of structured noise. Second row: results obtained when applying our algorithm to the score maps of the first row, with size $\delta = 9$. Third row: results obtained by iteratively applying our algorithm. (a) Noise density 0.012; (b) Noise density 0.025; (c) Noise density 0.033.

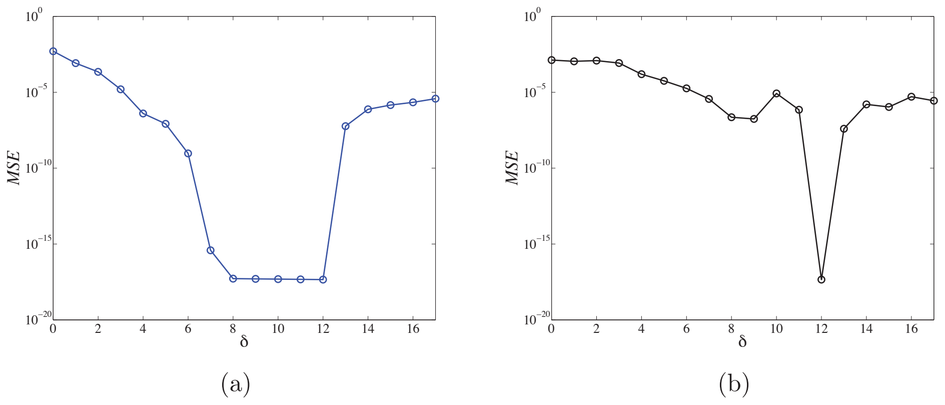


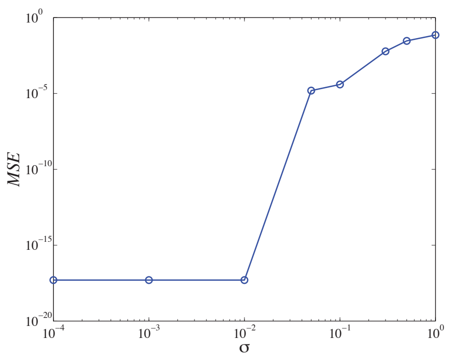
Figure 6.8: MSE as a function of δ, where the size of the patch is $D = (2\delta + 1) \times (2\delta + 1)$, when adding (a) Gaussian noise and (b) Structured noise to the test image. The results demonstrate the trade-off involved in the choice od D. In the case of Gaussian noise, the reconstruction of the method is exact for an optimal range of the size between 8 and 12. For structured noise there is a unique optimal value of δ, equal to 12. Notice the logarithmic scale of the $RMSE$ axis and that the minimum error is about 10^{-18}, thus, lower than machine precision. This means that, for the optimal values of δ, the reconstruction of our method is exact, up to numerical approximations.

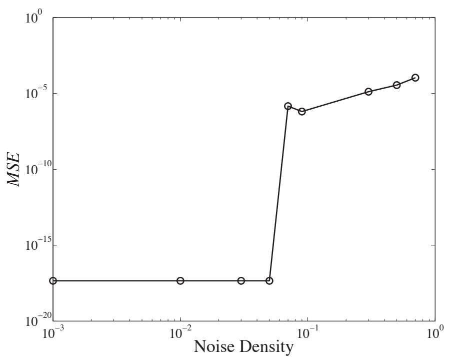
For values smaller than 8 or bigger that 12, the reconstruction is not perfect. In the case of a too small patch, we do not have enough contextual information in order to discriminate the real underline signal and the projection can vary on neighboring pixel positions. As a consequence, condition (ii) of Sec. 6.3.1 is not satisfied. While for a too large patch, the training set does not contain all the possible configurations of the training patches and therefore it is condition (i) which is not satisfied.

We repeat the same experiment, but this time we generate X by adding to Y_{te} structured noise. More precisely, we add salt-and-pepper noise, by randomly turning off pixels on the linear structure and turning on pixels in the binary ground truth. Then, we apply function d_C. Examples of the resulting score images X are shown in the first row of Fig. 6.7. They emulate the output of a pixel-wise regressor with false detections on the background and missing detections on the linear structure. The error as a function of δ is shown in Fig. 6.8(b). We see that now perfect reconstruction is achieved only for a larger value of $\delta = 12$, compared to the Gaussian noise case, this is because now the structured errors in the score map X locally resemble to the structures in the train images. Therefore, more contextual information is needed in order to correctly discriminate them.

MSE as a Function of the Noise Intensity

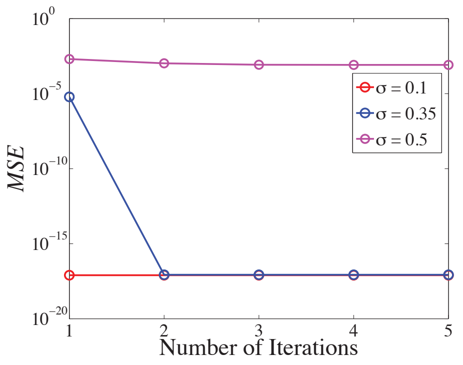
We then performed a second series of experiments. Now we fix the value of the patch size, as the optimal one found in the previous experiments and vary the intensity of noise.

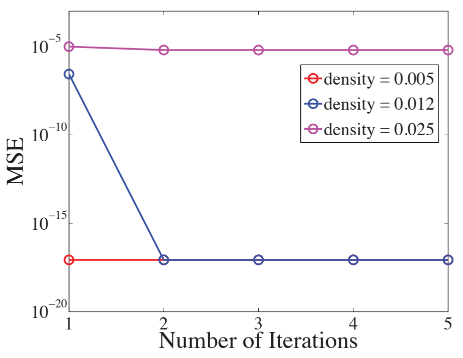




(a) (b)

Figure 6.9: MSE as a function of the noise intensity added to the test image. (a) Gaussian noise of variance σ^2 and $\delta = 9$; (b) Structured noise and $\delta = 12$. When the level of corruption is under a certain threshold, the error is lower than machine precision and therefore the reconstruction of the method is exact . When the noise is too strong, our algorithm can only compute an approximation of the real projection. Notice the logarithmic scale on both axis.





(a) (b)

Figure 6.10: MSE as a function of the number of iterations of our projection algorithm, for different levels of noise. (a) Gaussian noise and $\delta = 9$; (b) Structured noise and $\delta = 12$. The MSE decreases for the first few iterations and then remains constant. If the noise is below a certain level, our method converges to the exact solution after one or two iterations. If the noise is too strong the projection converges to a different image than the ground truth.

The results are shown in Fig. 6.9(a) and Fig. 6.9(b) for Gaussian and structured noise respectively. From these plots we see that after a certain level of noise, we can not recover the exact image dY_{te} anymore. The reason is that with such level of noise, either the real underline structure is not visible anymore or we would need too large a window in order to discriminate them. In the first case, the score map X is too far from the set $\mathcal{M}_N$ and its projection on it can be different than dY_{te}. In the second case, we are again in the situation where the training set $\mathcal{M}_D$ does not contain all admissible patches. However, notice that in all the experiments the MSE remains low. In particular, except for really strong level of noise or really small values of the patch size, the MSE is always lower than 10^{-4}. Example of images reconstructed with our method for different levels of noise are shown in Fig. 6.6 for Gaussian noise and Fig. 6.7 for structured noise. We notice that when the noise is too strong our algorithms add local line-like structures on the background or introduces gaps in the real linear structure. We notice however that even when the level of noise is extreme, our method is still able to partially recover it.

Iterative Projection

Finally, we perform a last series of experiments by iteratively apply our method to the score map. As before, we fix the value of δ as the optimal one found in the first set of experiments and we compute the MSE for different levels of noise. The reconstruction errors as a function of the number of iterations are shown in Fig. 6.10. We can see that the error decreases for the first few iterations and then it converges, meaning that the score map does not change anymore. For moderate level of noise, iterating the projection helps to further clean some spurious response that could not be eliminated at the first iteration (*e.g.* third row of Fig. 6.7(a)). For stronger levels of noise, instead, the method converges to a different image than the ground truth dY_{te}. In this situation the initial score map is too corrupted and it is not possible to recover dY_{te} with our method, as illustrated also by Fig. 6.6(c) and Fig. 6.7(c).

In summary, in this section we have empirically shown that our method is able to correctly recover an image in $\mathcal{M}_N$, when the conditions of Sec. 6.3.1 are satisfied. By studying the approximation error as a function of the size of the patch used to compute the projection we have also confirmed observations that we already deduced from the theoretical analysis.

In Sec. 6.5, we will apply our method to real datasets, showing that it helps improving the results score maps of a pixel-wise regressor. Before this, in the next section, we discuss some implementation details.

6.4 Implementation Details

In this section, we describe some implementation details of our method. In Sec. 6.4.1, we define an extension of our method by considering a multi-resolution framework. This extension will be used in the experiments section to handle situations in which very large values of the patch size D are needed in order to correct the errors of the regressor. Then, in Sec. 6.4.2, we describe how we can avoid the computation of the nearest neighbors for many image patches. In this way, we can decrease the computational complexity of our method.

6.4.1 Multi-Resolution Approach

Given the score image X, the only parameter of our method is the size D of the patches x_i on which the projection is computed.

Ideally, we would like D to be large enough to capture enough contextual information. At the same time, using a too large value for D makes it difficult to gather a representative training set of patches. As a consequence, a large value of D can provoke loss of details. To handle this trade off, we can adopt a multi-resolution approach. For clarity's sake, we describe below the case of 2 resolutions, but the generalization to an arbitrary number of them is straightforward.

Given two patch sizes $D_1 > D_2 > 0$, for every pixel p_i we consider the patch $x_i = X(\mathcal{N}_{D_1}(p_i))$ of size D_1 and its central part of size D_2, $x_i^{(\text{cent})} = X(\mathcal{N}_{D_2}(p_i))$. Then, we consider the downsampled version of x_i to size D_2, $x_i^{(\text{down})}$.

We do this for both the score images and the training ground truth patches $y \in \mathcal{M}_{D_1}$. We then perform Nearest Neighbors search in terms of the distance

$$k(x_i, y) = \|x_i^{(\text{cent})} - y^{(\text{cent})}\|^2 + \|x_i^{(\text{down})} - y^{(\text{down})}\|^2. \tag{6.8}$$

We then take the multiscale projection $\Pi_{D_1/D_2}(x_i)$ to be $y_{i_*}^{(\text{cent})}$, where $y_{i_*} = \arg\min k(x_i, y)$ in $\mathcal{M}_{D_1}$. This replaces the projection $\Pi_D(x_i)$ in the definition of $\Pi_{D \to N}(X)$.

Thanks to the downsampling term in Eq. (6.8) we can include more contextual information in the method. At the same time, by considering only the smaller central part $y_i^{(\text{cent})}$ for the final projection, we can preserve the details.

6.4.2 Efficient Implementation for Sparse Score Maps

Given the score map X, the main computational cost of our method comes from the computation of the nearest neighbors $\{\Pi_D(x_i)\}_{i=1}^N$, for every patch of size D in image X.

Many algorithms for approximated nearest neighbor search have been proposed [16, 17, 147] and can be applied in combination with our method. Since the size D used in our experiments can be very large, especially for 3D data, we use in our implementation the FLANN library [147], which is optimized for nearest neighbors search of high dimensional vectors.

Moreover, we take advantage of the specific properties of the ground truth set $\mathcal{M}_N$, and in particular of the sparsity of the ground truth images, to further reduce the computational cost. In fact, it is easy to show that if the maximum of a score patch x_i is smaller than a given threshold, its nearest neighbor is necessary the uniform patch of zeros.

More precisely, suppose that the patch of zeros $\mathbf{0}$ of size D belongs to $\mathcal{M}_D$. Then, given an image patch x_i, we have

$$\text{if} \quad \max_{p \in \mathcal{N}_D(p_i)} x_i(p) < \min_{y \in \mathcal{M}_D \setminus \{\mathbf{0}\}} \frac{\|y\|_2^2}{2\|y\|_1}, \quad \text{then} \quad \Pi_D(x_i) = \mathbf{0}. \tag{6.9}$$

This means that we do not need to explicitly compute the nearest neighbors of those patches whose maximum is smaller than a given threshold, where the threshold can be computed in closed form from the training set $\mathcal{M}_D$.

To prove the statement above, we start by observing that $\Pi_D(x_i) = \mathbf{0}$ if and only if $\|x_i - \mathbf{0}\|^2 < \|x_i - y\|^2$, for all $y \in \mathcal{M}_D \setminus \{\mathbf{0}\}$. Writing explicitely the distances, we have $\Pi_D(x_i) = \mathbf{0}$ if and only if

$$\sum_p x_i(p)^2 < \sum_p (x_i(p) - y(p))^2 = \sum_p \Big(x_i(p)^2 - 2x_i(p)y(p) + y(p)^2 \Big). \tag{6.10}$$

Subtracting the left hand side from both sides of (6.10), we have

$$0 < -2\sum_p x_i(p)y(p) + \sum_p y(p)^2 \quad \Leftrightarrow \quad 2\sum_p x_i(p)y(p) < \sum_p (y(p))^2. \tag{6.11}$$

This gives us the condition

$$\Pi_D(x_i) = \mathbf{0} \Leftrightarrow 2\sum_{p \in \mathcal{N}_D(p_i)} x_i(p)y(p) < \sum_{p \in \mathcal{N}_D(p_i)} (y(p))^2, \ \forall y \in \mathcal{M}_D \setminus \{\mathbf{0}\}. \tag{6.12}$$

Now let $x_{\max} = \max_p x_i(p)$. Since $y(p) \geq 0$ for every p, we have[2]

$$\sum_p x_i(p)y(p) \leq \sum_p x_{\max} y(p) = x_{\max} \sum_p y(p). \tag{6.13}$$

Thus, from (6.13) and (6.12)

$$\sum_p y(p)^2 > 2x_{\max} \sum_p y(p) \Rightarrow \sum_p y(p)^2 > 2 \sum_p x_i(p)y(p) \Rightarrow \Pi_D(x_i) = \mathbf{0}, \tag{6.14}$$

where the last implication follows by (6.12). Since $y(p) \geq 0$ and $y \neq \mathbf{0}$, we have $\sum_p y(p) = \|y\|_1 > 0$ and we can write (6.14) as

$$\text{if} \quad x_{\max} < \frac{\sum_p y(p)^2}{2\sum_p y(p)} \quad \forall y \in \mathcal{M}_D \setminus \{\mathbf{0}\}, \quad \text{then} \quad \Pi_D(x_i) = \mathbf{0}, \tag{6.15}$$

that is condition (6.9) we wanted to prove.

Thanks to this observation, depending on the sparsity of the ground truth images and of the score map, in the experiments of next section, we can avoid calculating the nearest neighbor for up to 60% of pixels.

6.5 Experimental Evaluation

To demonstrate the versatility of our method, we evaluate it on four very different problems, road centerline detection in aerial images, blood vessels delineation in retinal scans, membrane detection in 3D Electron Microscopy (EM) stacks and boundary detection in natural images. The code used in our experiments is available online at `http://cvlab.epfl.ch/software/centerline-detection`.

6.5.1 Centerline Detection

In this section, we consider again the problem of road centerline detection in aerial images.

We use the same Aerial dataset as in Chap. 5, a sample image of the dataset is shown in Fig. 6.1. This dataset comprises 13 training- and 13 test-images. For each one, manually annotated road centerlines and widths are available. In this section, we will focus only on the task of centerline localization, and therefore we will not consider the radial information.

[2]Form the definition of d_C in Eq. (6.1), we have $y(p) \geq 0$. For generic $y(p)$ we should consider $x_{\max} = \max_p |x_i(p)|$

We use the training data to learn the regressor used to compute the score map, using the method of Chap. 5. As can be seen in Fig. 6.1(b), the result while state-of-the-art can still be improved, especially near junctions, which is illustrated in Fig. 6.1(c).

To this end, we used the approach of Sec. 6.2.1 with the projection patch size equal to $D = 81 \times 81$ and the size of the patch used to compute the averaging equal it $R = 21 \times 21$.

To build the training set of patches used in the nearest neighbor search, we randomly sampled $3 \cdot 10^5$ patches from locations within a distance of 16 pixels to the ground truth centerlines to which we added a uniform patch of zeros, corresponding to the background. We also randomly rotated the training patches to obtain a more general dataset. In our MATLAB implementation, processing a small 620×505 image on a multi-core machine took a few seconds and a larger 1185×898 one about 40.

For this dataset, we found that the use of a large patch size D is required to correct the mistakes of the regressor. However, using a too large value for D makes it difficult to gather a representative training set of patches. As a consequence, a large value for D can result in loss of details. To handle this trade off, we adopted the multi-resolution approach of Sec. 6.4.1. We use $D_2 = 81 \times 81$ and $D_1 = 41 \times 41$.

In order to select the optimal parameters, thanks to the efficiency of our method, we could extensively search in the space of the parameters by using a cross-validation procedure.

We will refer to our approach as **Ours-SingleRes** and **Ours-MultiRes** depending on whether we use this multiscale approach or not.

Baselines In order to evaluate the accuracy of our method, we consider for comparison the algorithm we used to produce our input score maps (that is the method of Chap. 5) and the Structured Edge detector of [54], which we will refer to as **Reg-AC** and **SE** respectively. For the latter, we used the code provided by [54] and trained a structured Random Forest to predict the centerline locations.

To highlight the importance of using ground truth images for nearest neighbor search, we also applied Non-Local Means denoising [168], which relies on nearest neighbor search of patches in image itself, to the score images we used as input for our algorithm. We also applied to them K-SVD denoising [3], where the required dictionary was built from the ground truth images. We used the code provided by the authors of [168, 3]. We experimented with different parameters and found consistently similar results. We will refer to these approaches as **Reg-AC-NLM** and **Reg-AC-KSVD**, respectively.

Table 6.1: Aerial dataset results. The values correspond to the F-measure computed on whole image (first column) and on pixels close to junctions only (second column). Our method is more accurate than the state-of-the-art, in particular close to junctions.

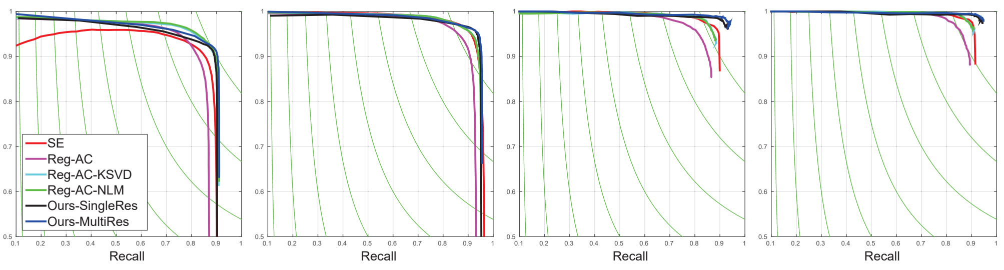
	Whole image	Junctions only
SE [54]	0.93	0.89
Reg-AC (Chap. 5)	0.91	0.82
Reg-AC-NLM	0.93	0.87
Reg-AC-KSVD	0.93	0.87
Ours-SingleRes	0.94	0.92
Ours-MultiRes	**0.95**	**0.94**

Image Evaluation We applied Non-Maximum Suppression to the output of all methods to find the actual centerlines and used the evaluation procedure of the Berkeley benchmark [10]. We computed Precision-Recall curves that include a tolerance factor for centerline localization. In Table 6.1, we give the results for a 2 pixel tolerance. The corresponding Precision-Recall curves are shown in Fig. 6.11. The rankings are mostly independent of the choice of this factor and our approach comes out consistently ahead.

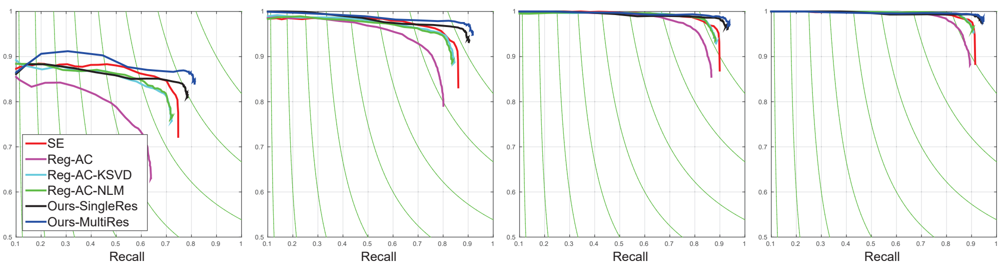
Junctions Evaluation The evaluation above does not account for the topological properties of the centerlines. Therefore, since junctions are present only at sparse image locations, errors close to junctions have only a small influence on the final performance. This is a weakness of this evaluation scheme because accurate delineation near junctions is particularly important for subsequent processing steps.

To remedy this, we recomputed the Berkeley metrics only near junctions. More precisely, we automatically identified from the test ground truth images the junction locations by using morphological operations, and then considered 21×21 regions centered around them. We then computed the Precision-Recall curves only there. The Precision-Recall curves are shown in Fig. 6.11 and the corresponding F-measures for a tolerance of 2 pixels are given in Table 6.1.

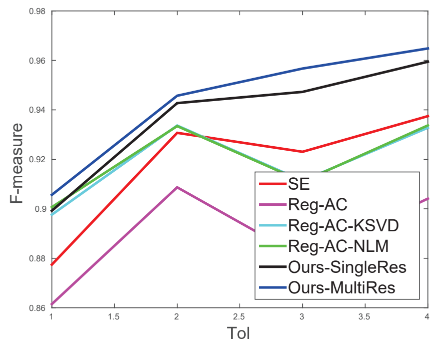
Note that the advantages of our method and in particular the multi-resolution approach are even more marked near junctions, where ambiguities are strongest. The F-measure as a function of the tolerance factor is shown in Fig. 6.11.



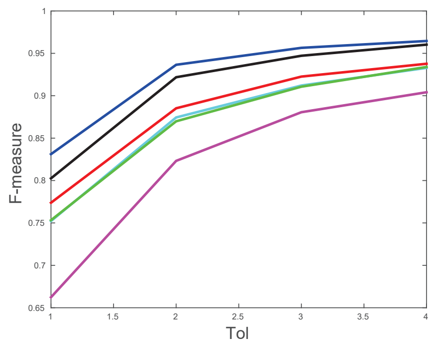
(a) tol = 1 (b) tol = 2 (c) tol = 3 (d) tol = 4



(a) tol = 1 (b) tol = 2 (c) tol = 3 (d) tol = 4



(a)



(b)

Figure 6.11: Aerial Dataset evaluation results. Top row: Precision-Recall curves for centerline detection for different tolerance values (in pixels), computed on the whole image. Second Row: Precision-Recall curves for the junctions evaluation. Third row: F-measure as a function of the tolerance factor (in pixels), (a) Whole image; (b) Junctions only.

6.5.2 Vessel Segmentation

We consider the problem of segmenting blood vessels in retinal scans. To test our approach we consider the DRIVE dataset [185], which comprises 20 training images and 20 test images of size 565×584.

We train a regressor to return distances from the blood vessels. For this task, the structure of interest, while still elongated, is not limited to centerlines and has a visible width. Therefore, we trained the regressor to return its maximal response over the whole width of the blood vessels, instead of only at centerlines as for Sec. 6.5.1. We then applied nearest neighbor projection with patch sizes of Sec. 6.2.1 equal to $D = 13 \times 13$ and $R = 7 \times 7$. We sampled all training patches within a distance of 6 pixels to a vessel and randomly rotating them.

We compare our approach with **SE** [54], **N^4-Fields** [74] and **KernelBoost** [23]. Table 6.2 shows the F-measure obtained with the different methods. Our approach is comparable or better than the state-of-the-art [23] and [74].

To study the behavior of the methods close to junctions, which are of great importance to get the topology of the vessels right, but have little influence on the performance computed on the whole image, we repeated the junctions evaluation, similarly to Sec. 6.5.1. As shown in Table 6.2 our method outperforms the baselines by a large margin. Fig. 6.12 shows the results on a test image. Fig. 6.13 shows the results on a particularly complex region of a test image, with several thin junctions and low contrast. Our method can correctly reconstruct the topology of the blood network, which [74] fails to achieve. Moreover, as can be seen from Fig. 6.13, our method is more accurate than the ground truth in some part of the images. This actually penalizes our method when evaluating on the whole image.

Table 6.2: DRIVE results. The table shows the F-measure for the different methods computed on the whole image and only on regions around a junction. Our method reaches state-of-the art performance on the dataset and outperforms the other methods on the junction evaluation.

	Whole image	Junctions only
SE [54]	0.67	0.52
Reg-AC	0.79	0.71
KernelBoost [23]	0.80	0.76
N^4-Fields [74]	**0.81**	0.74
Ours	**0.81**	**0.80**

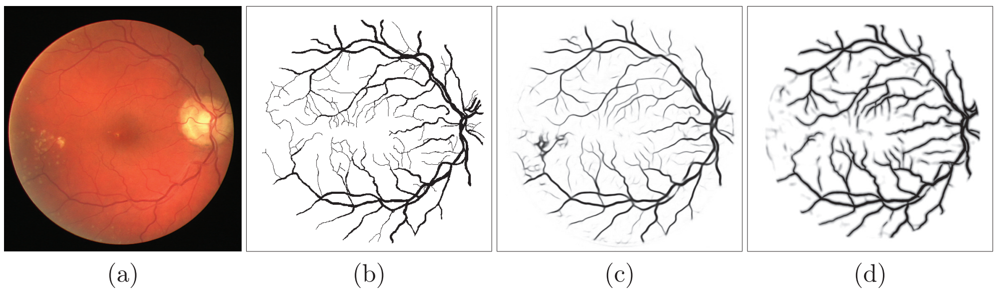


Figure 6.12: Drive results. (a) Image; (b) Ground truth; (c) **N^4-Fields** [74]; (d) **Our approach**. Our method responds strongly on thin vessels and is less sensitive to the bright structured noise on the left part of the image.

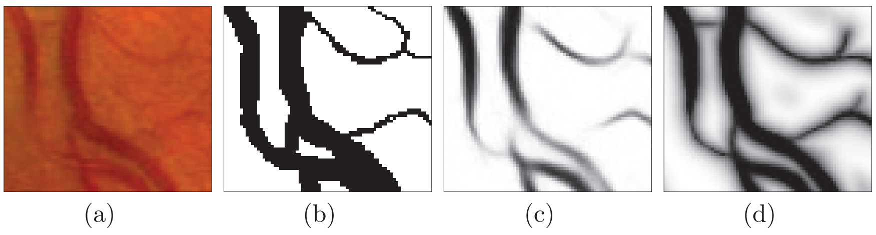


Figure 6.13: Vessel segmentation. Raw image and score maps on a region with complex topology. (a) Image; (b) Ground truth; (c) **N^4-Fields**; (d) Our approach. Our method recovers jucntions in the image even in regions with very low contrast and for very thin structures.

6.5.3 Membrane Segmentation

In this section we consider the problem of membrane detection in 3D EM stacks. Our dataset is made of four stacks of size $250 \times 250 \times 309$. The first stack is used for training, the second for validation, and the last two for testing. An expert annotated all voxels belonging to dendrites in these volumes. From these, we automatically extracted the dendritic boundary voxels that form the membranes. Since other cells such as axons and ganglions are also present but not annotated, we only considered the voxels within a distance of 11 voxels from the dendrites for both training and evaluation.

The Context Cue Features of [22] have proved very effective for EM supervoxel classification and we use them here as input to the regressor of Chap. 5.

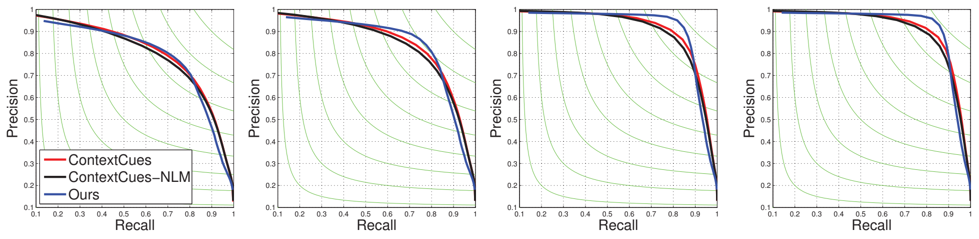
We applied our method to the output score returned by the regressor after 0, 1, and 2 iterative regression iterations. For this task smaller patches gave better results, so we used patch sizes $D = 9 \times 9$ and averaging on a $R = 5 \times 5$ window. We sampled $2 \cdot 10^6$ truth patches within a distance of 6 pixels to a vessel plus a uniform patch of zeros, for nearest neighbor search. For comparison purposes, we also applied Non-Local Means denoising to the score maps. All the parameters were optimized using the validation volume. Processing a test stack took about 6 minutes.

As for the centerlines, to account for potential inaccuracies in the annotations, we compute a 3D version of the Berkeley benchmark with different tolerance factors. The F-measures for a tolerance of 3 voxels are shown in Table 6.3. The the Precision-Recall curves and the F-measure as a function of the tolerance are given in Fig. 6.14 and the rankings are similar.

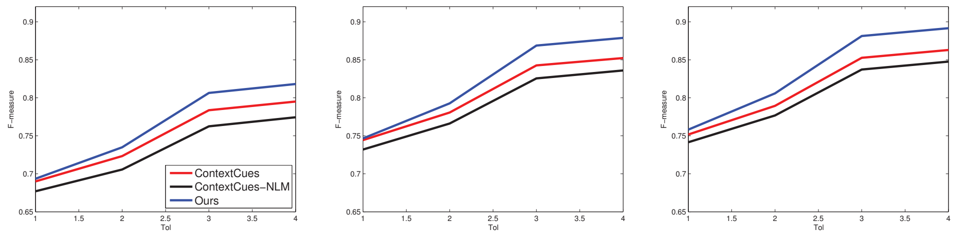
Our approach always brings an improvement compared to the baseline. Applying Non-Local Means to the score map made the performance slightly worse in this case. This is probably because applying Non-Local Means smoothes the regressor's response while our approach keeps it sharp, especially close to junctions, as shown in Fig. 6.15.

Table 6.3: F-measure for the Membrane Detection dataset. We applied our approach to the output of [22], trained to predict the regression function of Sec. 6.2, at different iterative regression iterations. Notice that our method applied at the first iteration performs better ($F = 0.87$) than the other methods at the second iteration ($F = 0.85$ and $F = 0.84$).

	No Iter	Iter 1	Iter 2
ContextCues [22]	0.78	0.84	0.85
ContextCues + NLM	0.76	0.83	0.84
Ours	**0.81**	**0.87**	**0.88**



(a) tol = 1 (b) tol = 2 (c) tol = 3 (d) tol = 4



(a) No Iterations (b) Iteration 1 (c) Iteration 2

Figure 6.14: Membrane Segmentation Results. Top: Precision-Recall curves at the second iterations for different tolerance values (in voxels). Bottom: F-measure as a function of the tolerance factor (in voxels) for different iterations.

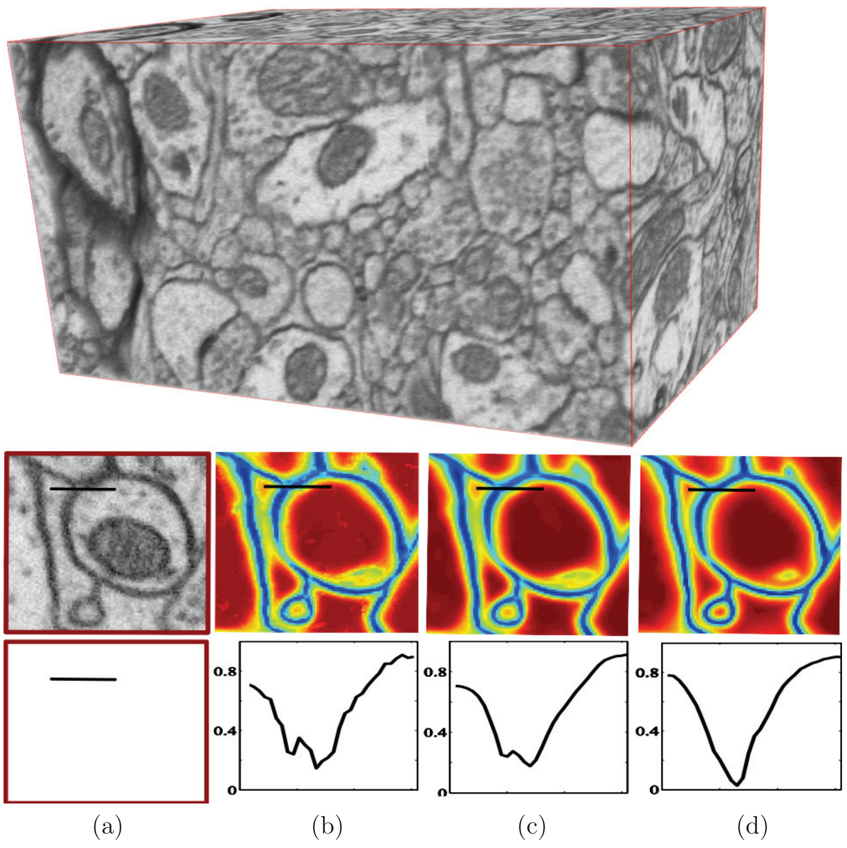


Figure 6.15: Top row: A test stack used in our experiments. Middle row: Detail of a slice of a test stack and responses of different methods. (a) Image; (b) Initial score map; (c) Non-Local Means applied to (b); (d) Our approach applied to (b). Bottom row: intensity values along the orizontal black lines in the images. Our method removes background spurious responses while sharpening the response on the membranes. The smoothing effect of the Non-Local Means approach instead decreases accuracy on the junctions. Best viewed in color.

6.5.4 Boundary Detection

To test our method on the boundary detection task, we consider the BSDS dataset [10], as in Chap. 5. We trained our regressor with the same parameters as in Sec. 5.6. We ran our approach with patch size $D = 21 \times 21$ and $R = 11 \times 11$ sampling $3 \cdot 10^6$ train patches. As is done [160, 54, 74], we use a multiscale approach to detect the boundaries. More precisely, we apply our projection method to the score maps returned by the regressor at 3 different resolutions—half, original, and double size—and then average the results. Finally, Non-Maximum Suppression was applied to the score map for evaluation purposes. Processing one image at the three different scales took less than one minute.

Table 6.4 shows the ODS and OIS scores of ours and previous methods. ODS is the best F-measure computed using the same threshold for all images in the test set. OIS is the F-measure computed by selecting the best threshold for each image independently. AP is the area under the Precision-Recall curve [10]. The baselines are given by a pixel-wise classifier (**SCG** [160]) and two patch wise classifiers (**SE** [54] and **N^4-Fields** [74]). We also considered an extension of **SE**, where we train the structured forests to return patches corresponding to our regression-based ground truth, rather than binary ones. We will refer to this approach as **Reg-SE**. Below, we describe in detail how its output is computed.

Structured Edge Detection with Regression In order to process structured output patches with binary decision trees, in **SE**, at every node, a low-dimensional embedding of the ground truth patches is considered. In this way, node splitting is reduced to a classical muli-label classification problem and standard information gain criteria can be used.

For our regression formulation, **Reg-SE**, we tried different low-dimensional embedding and splitting criteria. The best that worked for us was to first compute sums along random lines through the ground truth patches and then to compute a PCA decomposition of the resulting vector. Then, the Gini information gain criterion was applied to the first 8 PCA components. At the leaf nodes, in order to store only one patch corresponding to a boundary profile, we experiments both by averaging all the patches arriving at that node, and also by considering the nearest neighbor of this average in the set of training patches. The latter gave the best performance on the validation set and therefore we used it at test time. The performance are given in Tab. 6.4, notice that for **Reg-SE**, they are computed without applying our projection method to the score map.

From Tab. 6.4, we see that our approach is the most accurate in terms of ODS and OIS. We also observe that using regression in combination with [54] (**Reg-SE**) gives slightly higher ODS and OIS compared to the original binary formulation **Reg-SE**. However,

Reg-SE has lower performance than our approach. This is probably do to the fact that it is more difficult for the trees to separate the regression-based ground truth patches, compared to a pixel-wise formulation. This underlines the advantage of our approach of splitting the learning phase in two steps, by first learning an approximated pixel-wise regressor and then matching local patches, rather than directly predicting the patches.

We notice that in this case our approach only slightly improves the performance of the input pixel-wise score map **Reg-AC**. This is because to the Berkeley benchmark considers a large tolerance factor (of about 4 pixels) in the localization of the centerlines. Moreover, the benchmark does not take into account the topology of the solution. However, when we compare qualitatively the output of our method, we can see that our method is able to eliminate background noisy detection and that it returns more continuous boundaries, as shown in Fig. 6.17. A comparison of the boundaries detected with the other patch-wise methods on two test images are depicted in Fig. 6.16.

Finally, we observe that applying our method slightly decreases the AP of the score map. This is because the projection, by removing background noise and weak responses, can also eliminate some weak detections corresponding to actual boundaries. However, the advantage brought by the smoothing and linking effect and the projection increase the overall performance, as indicated by the ODS and OIS scores.

As already discussed in Sec. 5.6, the work of [211] recently improved the performance of local pixel-wise and patch-wise methods for boundary detection, giving an ODS measure of 0.79. A similar idea has been used in [165] for segmenting membranes in 2D EM slices, showing also in this case a large accuracy improvement. This was achieved thanks to the image-wise formulation of these two algorithms, which rely on a Fully Convolutional Neural Network architecture [125]. In the next chapter, we discuss how the principles behind these approaches can be used to improve our centerline detection method.

Table 6.4: Boundary dataset results. Our method is more accurate than state-of-the-art pixel-wise and patch-wise prediction methods. Direcly learning the patches from the images, even by using our regression-based ground truth (**Reg-SE**), is less accurate than our solution of first learning an approximate score map using a pixel-wise regressor and then matching local patches to improve its consistency.

	ODS	OIS	AP
SCG [160]	0.739	0.757	0.768
SE [54]	0.743	0.764	**0.800**
N^4-Fields [74]	0.747	0.770	0.777
Reg-SE	0.746	0.768	0.738
Reg-AC	0.755	0.775	0.787
Ours	**0.756**	**0.777**	0.763

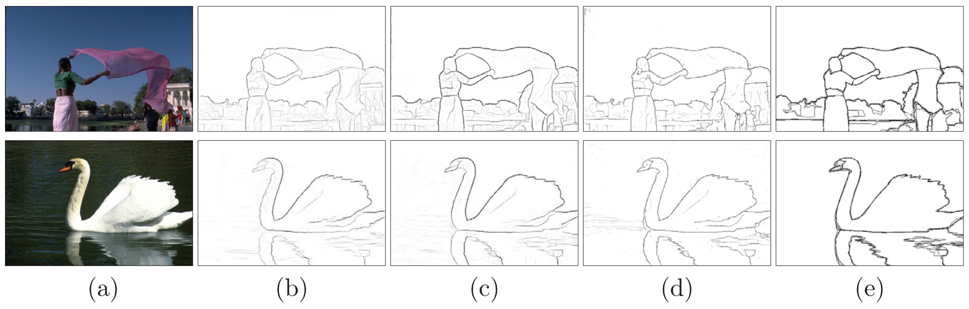


Figure 6.16: Boundary detection results. Boundaries obtained by Non-Maximum Suppression on the score image returned by different methods. (a) Image; (b) **SE** [54]; (c) **N^4-Fields** [74] (d) **Ours**; (e) Human annotations. Our approach returns more continuous boundaries and preserves important details, like for example the right hand of the lady on the top image and the beak of the swan in the bottom image.

Conclusion

In this chapter, we have proposed an effective method for detecting centerlines, segmenting linear structures, and finding boundaries and membranes.

We have shown that it compares favorably to the state-of-the-art and can be understood as an efficient projection onto the set of feasible solutions. This means that domain knowledge, such as engineering constraints on roads and biological ones on blood vessels and membranes, could be introduced as a preprocessing step by refining this set. In practice, this could mean cleaning-up the patches we sample from it.

In the next chapter we discuss this and other possible extensions of our method. We consider the limitations of the current approach and possible future work for improving it.

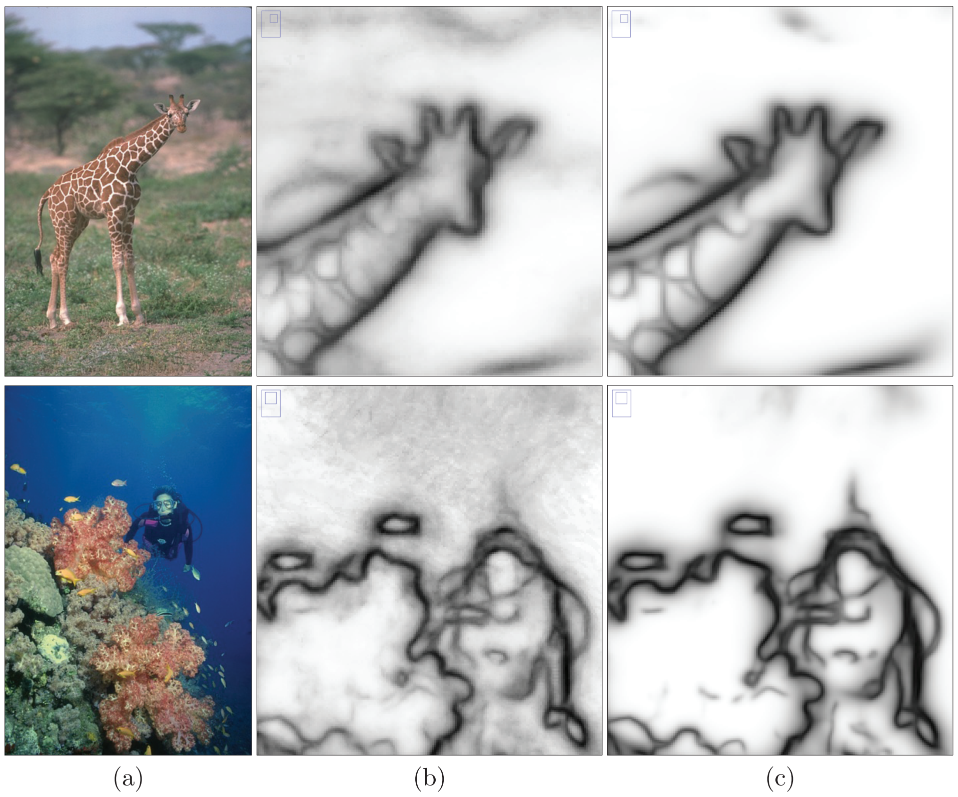


Figure 6.17: Details of the score map obtained for boundary detection. (a) Original Image; (b) Detail of the boundary score map of **Reg-AC**; (c) Detail of the score map after applying our patch-wise method. Our method sharpen the boundary map, suppress background responses and recover details, like for example the giraffe's horn in the first row and the scuba diver head in the second row.

CHAPTER 7

Conclusion

In this thesis, we have considered the problem of detecting centerline and estimating the radius of linear structures in n-dimensional images.

We began in Chap. 1 by defining the centerline detection problem and the interest it represents for the field of Computer Vision. We also presented the main challenges and some relevant applications associated with it.

In Chap. 2, we discussed previous work. We described the main characteristics of linear structures detection methods, and we underlined their advantages and their limitations. In particular we presented a general taxonomy for centerine and boundary detection methods, which was useful to categorize the different approaches presented in the thesis.

In Chap. 3, we reformulated the centerline detection problem using a regression-based approach. We showed that regression is a better formalism compared to both classification-based and hand-crafted methods. To solve the associated regression problem, we used a pixel-wise method, based on local convolutional features. We demonstrated the advantage of our method by applying it to several challenging 2D and 3D datasets and showing a large improvement over the state-of-the-art.

The features used by our method were described in Chap. 4. In this chapter, we also introduced a novel and generic approximation scheme for convolutional filter banks. More precisely, we have shown how tensor decomposition techniques can be used to approximate a large non-separable filter bank as linear combination of a smaller set of separable filters. This decomposition allowed us to efficiently process large datasets and compute large feature vectors, used for different Computer Vision tasks.

In Chap. 5, we introduced a scale-space context-aware method for estimating the location and the radius of the linear structures. Thanks to radial and spatial contextual information, our algorithm is able to correctly discriminate the linear structures when local information is not enough, thus, improving the performance of the local method of Chap. 3. Moreover, we also used the score map returned by our method as input to a state-of-the-art tracing algorithm, showing an improvement also in this case. Finally, we demonstrated the generality of our approach by applying it to the boundary detection problem, showing better localization accuracy compared to a classification formulation also for this problem.

In Chap. 6, we considered the problem of the geometrical consistency of the solution, which is of critical importance for correctly reconstruct the topology of the linear structures. Pixel-wise methods trat each pixel independently, therefore, they generally return inconsistent results. However, in this chapter we have shown that by applying a patch-wise projection method to the score map of a pixel-wise regressor, we can mitigate this problem. We also introduced a geometric interpretation of our method and proved that it approximates the projection of the score map onto the set of admissible ground truth images. Our method is more accurate than other patch-wise methods, in particular close to topological relevant points, such as junctions.

We believe that our work represents an important step forward in the understanding and for the solution of the linear structures detection problem. However, many challenges and questions still need to be tackled in order to obtain a "computer system that can perform the delineation task at close to human levels" [65]. In the following we discuss future directions of research that, in the light of the work of this thesis, can help to further approach to this goal.

Future Work

In this section, we discuss the limitations of the methods described in this thesis and propose possible improvements.

(a) Improving the regression formulation In Sec. 3.2, we introduced our regression formulation as an efficient way to learn a function whose local maxima correspond to the centerline points of the structure of interest. We designed a function satisfying this criterion and tried to predict it by minimizing the squared error of our prediction. However, the shape of our regression ground truth is arbitrary and, in fact, there exist an infinity of functions satisfying the same criterion. Moreover, predicting exactly the values of our function is not the main interest of our formulation, since we are just interested in its local maxima. As discussed in Sec. 3.1, a ranking formulation could help overcome these limitations. However, solving the optimization problem associated with it would be computationally impractical. A way to solve this limitation can be to combine in an efficient way our regression formulation and the ranking one. For example, we could design a specific loss function penalizing the errors close to the centerline in a different way than the error for points far from it and we could add the ranking constraints only for a relevant subsets of pixels. Another solution would be to automatically learn the loss function, as it is done in adversarial training approaches [78, 48]. This case is discussed in point **(e)** below.

(b) Image-wise prediction Recent advances in Convolutional Neural Networks architectures [125] have shown that deep learning algorithms can be trained to return entire images as output. These architectures have been applied to the problem of pixel labeling [125], boundary detection [211] and membrane segmentation [165]. They have the double advantage of considering global information, by taking as input the whole image and also of producing smooth score maps, by directly returning the prediction for the whole image. Because of these advantages, they outperform pixel-wise and patch-wise prediction methods. However, these architectures usually show low localization accuracy. In the case of image segmentation, the results are too smooth and further processing is needed for accurate results, for example by using Conditional Random Fields [215]. In the case of boundary detection, the score maps returned by these methods have thick contours. Combining this type of architectures with our regression formulation we could solve this issue when applying them to the centerline detection problem.

(c) Separable filters for Convolutional Networks Although deep learning based methods, as the ones described in the previous paragraph, reach state of the art performance, they require huge computational power and very large databases in order to be trained effectively. In particular, their use is strongly limited in the case of 3D volumes, such as those used in biomedical settings and in this thesis. Typical 3D architecture only takes as input small volumes and therefore they can not take advantage of large contextual information as in the 2D case. The use of separable filters could help to mitigate this issue. In Chap. 4, we have shown how our separable filters scheme could be used to increase efficiency of Convolutional Networks at test time. In order to apply separable convolutions effectively also during training, our method should be modified. For example, we could start by approximating the filters learned on a dataset where we have enough training data with our separable filter approach. Then, supposing that the separable filters learned in this way are generic enough, we could train a new network, by only optimizing the weights w_k^j of Eq. (4.2), to recombine the separable filters. In this way, we would have a double computational advantaged: First, we have the expressive power of non-separable convolutions, but computed efficiently by separable ones and their linear combinations; Second, we do not need to compute the gradient of the filters, but, we would only optimize on the linear weights w_k^j used to recombine them.

(d) Improving the patch-wise projections Once a score map is obtained, for example by using the image-wise prediction as in **(b)**, it is still possible to apply the projection-based method of Chap. 6. However, the main limitations of this method are its dependency on a representative training set of patches and on the accuracy of the score map, which needs to be close enough to the real underline ground truth for accurate results. In order to overcome the first limitation, we could explicitly design the training set by modifying the patches sampled from the training images to have the desired properties. We could also weight the average of Eq. (6.4) depending on the distance of the score and the ground truth patches. To overcome the second limitation, instead, we could modify the distance function used to compute the nearest neighbors. For example, we could include a term which takes into account the original image information of the patches. Moreover, since it could be difficult to define an appropriate distance function for the image patches, we could automatically learn to match corresponding patches by using a Siamese Neural Network architecture [31].

(e) Adversarial training for centerline detection As discussed above, defining the right set of constraints and the most appropriate loss function for our problem might be problematic in practice. Difficulties might come from the computational point of view or by the impossibility to explicitly define the actual criterion we want our system to be able to emulate. Recently, adversarial training techniques [78, 48] have emerges as a way to automatically learning the loss function. They consist in alternatively training two models: the first one trying to discriminate the output of the second one from the ground truth images; and the second one trying to deceive the first one. In this way, after convergence, the second model will be able to generate predictions which are not distinguishable from the ground truth. This is achieved without defining an explicit criterion on the output shape. This method can therefore be adapted to the centerline detection problem to obtain geometrically consistent results.

(f) Other applications Finally, the ideas introduced in this thesis could be useful to solve other Computer Vision problems. Some works have already adapted our regression formulation to the 1-dimensional case, for the detection of feature points [202] or for accurately detecting cells in biomedical data [102, 212]. Our method could also be extended and used to track points in spatio-temporal sequences. In fact, trajectories of moving points appear as curvilinear structure in a 3D or 4D space. Using our regression-based approach could be useful to accurately localize and track these objects. For example, a relevant application would be the tracking of people joints in videos. The algorithm would take a video sequence as input and would output a score map whose local maxima correspond to joints locations in space and time. Then, applying Non-Maximum Suppression to the score map would then return joints locations over time.

To conclude, the problem of linear delineation remains a challenging and prolific Computer Vision problem. Since the beginning of the field, it interested a large number of scientists and brought to the design of general techniques, whose scope proved to go beyond the single linear delineation task. The study of this problem influenced and was influenced by advances in artificial and biological intelligence and led to important practical applications. We hope that our contribution represents a step forward, even if small, in the same directions and that it can be useful for the pursuit of the same goals.

APPENDIX A

Proof of Theorems

In this appendix, we consider the patch-wise projection method of Chap. 6. This method was introduced to improve the score map returned by a pixel-wise regressor. It amounts to projecting small patches of the score map onto the set of training ground truth patches. In Sec. 6.3.1, we stated that our approach can be interpreted as projecting the whole score map onto the set of admissible ground truth images. In this appendix, we formalize this concepts and we provide a proof of the Theorem 1 enunciated in Sec. 6.3.1. Moreover, we also consider the case of approximated projections and, by relaxing the hypothesis of Theorem 1, we provide bounds for the error made by our method, in terms of the errors made on the projected patches.

A.1 Equivalence of $\Pi_{D\to N}(X)$ and $\Pi_N(X)$

We start this section by introducing the notation used in the chapter, in Sec. A.1.1. Then, in Sec. A.1.2, we define precisely the hypothesis of Theorem 1 and prove its validity. In Sec. A.1.3, instead, we consider the case of approximated projection.

A.1.1 Notation

Let $I \in \mathbb{R}^N$ be an image containing elongated structures we are interested in extracting. Let $\{p_i\}_{i=1}^N$ be the grid of pixels on which the images are defined, where for simplicity we assume the image to be squared: $N = \sqrt{N} \times \sqrt{N}$. For a pixel p_i, we denote by $\mathcal{N}_D(p_i)$ the squared neighborhood of pixels centered at p_i of size $D = \sqrt{D} \times \sqrt{D}$.

Let $Y \in \{0,1\}^N$ be the binary ground truth corresponding to the centerlines in image I and let $dY \in \mathbb{R}^N$ be the image obtained by applying function d_C to every pixel of Y, where d_C is defined by

$$d_C(p) = \begin{cases} e^{a(1-\frac{\mathcal{D}_Y(p)}{d_M})} - 1 & \text{if } \mathcal{D}_Y(p) < d_M \\ 0 & \text{otherwise} \end{cases}, \tag{A.1}$$

with $\mathcal{D}_Y$ the Euclidean distance transform of Y, $a > 0$ a constant that controls the exponential decrease rate of d_C close to the centerline and $d_M > 0$ a threshold value determining how far from a centerline d_C is set to zero.

We denote by $\mathcal{M}_N$ the set of images dY, obtained from all ground truth images Y, corresponding to admissible solutions for a given problem.

We define X the score image obtained by applying the regressor φ of Chap. 3 or Chap. 5 to every pixel of image I. The projection of X onto $\mathcal{M}_N$ is denoted by $\Pi_N(X)$ and it is given by

$$\Pi_N(X) = \underset{dY \in \mathcal{M}_N}{\arg\min} \|dY - X\|^2. \tag{A.2}$$

This projection represents the best approximation of X in the set of admissible ground truth images.

As described in Chap. 6, $\Pi_N(X)$ can not be computed in practice, therefore we approximate it by averaging the projections of small patches of X. In order to formalize this, we define for every pixel p_i of image X the corresponding patch x_i of size D centered at p_i, $x_i = X(\mathcal{N}_D(p_i))$. We also denote by $\mathcal{M}_D$ the set of all patches of size D extracted from images in $\mathcal{M}_N$.

The projection of x_i onto $\mathcal{M}_D$ is given by

$$\Pi_D(x_i) = \underset{y \in \mathcal{M}_D}{\arg\min} \|y - x_i\|^2. \tag{A.3}$$

Then, the approximated projection $\Pi_{D\to N}(X)$, returned by our method is defined as

$$\Pi_{D\to N}(X)(p) = \frac{1}{R} \sum_{i:p-p_i \in \mathcal{N}_R(p)} \Pi_D(x_i)(p - p_i), \tag{A.4}$$

where $R \leq D$ is the size of the neighborhood used for averaging and where we take $\Pi_D(x_i)$ to be centered at zero, with $\Pi_D(x_i)(p-p_i)$ the value of $\Pi_D(x_i)$ at $p-p_i$. In the following, to simplify notations, we will consider the case $R = D$. The generalization to a generic value of R is straightforward and it is discussed at the end Sec. A.1.2.

Let $\{q_d\}_{d=1}^D$ be the $\sqrt{D}\times\sqrt{D}$ grid of pixels on which the patches of size D are defined. For a pixel $p \in \{p_i\}_{i=1}^N$ in the image grid of pixels, we denote with $p^{(l)}$, for $l = 1,\ldots,D$ the elements of $\mathcal{N}_D(p)$.

For every patch x of size D, we can associate a neighborhood $\mathcal{N}_D(p)$ of size D in the image grid of pixels thanks to a one to one correspondence ν, given by $q_l = \nu(p^{(l)})$, for every l. In this case, we will say that the *support* of x is $\mathcal{N}_D(p)$. In other words, we say that the support of x is $\mathcal{N}_D(p)$ if x is considered to be centered at image location p. In such case, to simplify our notation, we will write $x(p^{(l)})$ instead of $x(\nu(p^{(l)}))$.

For example, given the score image X, the image patch $x_i = X(\mathcal{N}_D(p_i))$, centered at p_i, has support $\mathcal{N}_D(p_i)$. In the same way, we will consider the support of the projections $\Pi_D(x_i)$ to be $\mathcal{N}_D(p_i)$. In this way, we can rewrite the definition of $\Pi_{D\to N}(X)$ Eq. (A.4) as

$$\Pi_{D\to N}(X)(p) = \frac{1}{D}\sum_{i:p\in\mathcal{N}_D(p_i)} \Pi_D(x_i)(p). \tag{A.5}$$

We say that two patches x_i and x_j *overlap* if the intersection of their supports, $\mathcal{N}_D(p_i)\cap\mathcal{N}_D(p_j)$, is not empty.

We indicate with $X|_{\mathcal{N}}$ the restriction of an image X to a subset of pixels $\mathcal{N} \subseteq \{p_i\}_i$. For example, using this notation, we can write $x_i = X|_{\mathcal{N}_D(p_i)}$.

Let $\{y_j\}_{j=1}^J$ be a set of patches of size D such that for every j, y_j has support $\mathcal{N}_D(p_j)$. We say that $\{y_j\}_{j=1}^J$ *covers* the pixel grid $\{p_i\}_{i=1}^N$ if

$$\bigcup_j \mathcal{N}_D(p_j) = \{p_i\}_{i=1}^N. \tag{A.6}$$

This means that for every pixel p_i there exist at least one j such that p_i is in the support of y_j.

Then, for a set of patches $\{y_j\}_{j=1}^J$, such that $\{y_j\}_{j=1}^J$ covers $\{p_i\}_{i=1}^N$, we can define a new image of size N, which we call the *average image of patches* $\{y_j\}_{j=1}^J$ and we denote by $\bigsqcup_{j=1}^J y_j$. This new image is obtained by averaging the patches y_j on their supports. More precisely, for every pixel $p \in \{p_i\}_i$, the value of the average image at pixel p is given by

$$\left(\bigsqcup_{j=1}^{J} y_j \right)(p) = \frac{1}{\Gamma_p} \sum_{j:\ p \in \mathcal{N}_D(p_j)} y_j(p) \tag{A.7}$$

Where the normalization constant Γ_p is equal to the number of elements in the sum and in general it depends on the pixel location p. Notice that (A.7) is well defined for every p because of (A.6).

In other words, for a pixel p, the value of $\bigsqcup_{j=1}^{J} y_j$ in p is given by the average of the values $y_j(p)$, for those y_j with support containing p. For example, in the case of image patches $x_i = X(\mathcal{N}_D(p_i))$, since $x_i(p) = x_j(p)$ for all $p \in \mathcal{N}_D(p_i) \cap \mathcal{N}_D(p_j)$, we have

$$X = \bigsqcup_{i=1}^{N} x_i. \tag{A.8}$$

Notice that in this case we have $\Gamma_p = D$ for every p.

With this notations, using the fact that the support of $\Pi_D(x_i)$ is $\mathcal{N}_D(p_i)$, we can write the approximated projection $\Pi_{D\to N}(X)$ as

$$\Pi_{D\to N}(X) = \bigsqcup_{i=1}^{N} \Pi_D(x_i). \tag{A.9}$$

Also in this case the normalization constant in Eq. (A.7) is $\Gamma_p = D$ for every p.

A.1.2 Exact Projection

In this section, we prove the equivalence $\Pi_{D\to N}(X) = \Pi_N(X)$. We first give two definitions that will be used as hypothesis of the following Theorems.

Definition 1. *We say that $\mathcal{M}_D$ is* complete *in $\mathcal{M}_N$ if the following holds:*

$$Y \in \mathcal{M}_N \iff \exists \{y_i\}_{i=1}^{N} \subseteq \mathcal{M}_D \text{ such that } Y = \bigsqcup_{i=1}^{N} y_i, \tag{A.10}$$

where, $\forall i$, y_i has support $\mathcal{N}_D(p_i)$ and $\forall i, j$, $y_i(p) = y_j(p)$ for all $p \in \mathcal{N}_D(p_i) \cap \mathcal{N}_D(p_j)$.

This is property (i) given in Sec. 6.3.1 of the thesis. It means that the training set of patches $\mathcal{M}_D$ is composed of all admissible ground truth patches and that averaging patches that coincide in the intersection of their supports, gives an image of $\mathcal{M}_N$.

The following definition formalizes instead hypothesis (ii) of Sec. 6.3.1.

Definition 2. *Let us consider the set of all the projections $\{\Pi_D(x_i)\}_{i=1}^N$ of all patches of size D of an image X. We say that $\{\Pi_D(x_i)\}_{i=1}^N$ is* consistent *if, for all x_i and x_j such that $\mathcal{N}_D(p_i)\cap\mathcal{N}_D(p_j)\neq\emptyset$, we have $\Pi_D(x_i)(p)=\Pi_D(x_j)(p)$ for all $p\in\mathcal{N}_D(p_i)\cap\mathcal{N}_D(p_j)$.*

This means that the projection of two overlapping patches is the same for every pixel in the intersection of their support.

We can now state the following

Theorem 2. *If $\mathcal{M}_D$ is complete in $\mathcal{M}_N$ and if $\{\Pi_D(x_i)\}_{i=1}^N$ is consistent, then*

$$\|X-\Pi_{D\to N}(X)\|^2=\|X-\Pi_N(X)\|^2. \tag{A.11}$$

Proof. Since $\Pi_N(X)\in\mathcal{M}_N$ and $\mathcal{M}_D$ is complete, for every patch x_i the restriction of $\Pi_N(X)$ to x_i belongs to $\mathcal{M}_D$, $\Pi_N(X)|_{\mathcal{N}_D(p_i)}\in\mathcal{M}_D$. Then, by (A.3), we have for all x_i

$$\|\Pi_D(x_i)-x_i\|^2\le\|\Pi_N(X)|_{\mathcal{N}(p_i)}-x_i\|^2. \tag{A.12}$$

Let $\{x_{i_j}\}_{j=1}^K$ be a subset of $\{x_i\}_{i=1}^N$ such that $\mathcal{N}_D(p_{i_{j_1}})\cap\mathcal{N}_D(p_{i_{j_2}})=\emptyset$ for all $j_1\neq j_2$ and $X=\bigsqcup_j x_{i_j}$. The subset of patches $\{x_{i_j}\}_{j=1}^K$ is given by a grid of non-overlapping image patches covering the whole image [1].

Since $\{\Pi_D(x_i)\}_{i=1}^N$ is consistent, $\Pi_{D\to N}(X)=\bigsqcup_j\Pi_D(x_{i_j})$. In fact, for the hypothesis of consistency, the patches $\Pi_D(x_i)$ coincide in their intersection. Then,

$$\begin{aligned}\|\Pi_{D\to N}(X)-X\|^2&=\|\bigsqcup_j\Pi_D(x_{i_j})-\bigsqcup_j x_{i_j}\|^2=\\&=\sum_j\|\Pi_D(x_{i_j})-x_{i_j}\|^2.\end{aligned} \tag{A.13}$$

Where the second equality holds since patches x_{i_j} do not overlap. Eq.(A.13) tells that the distance between images X and $\Pi_{D\to N}(X)$ can be computed by summing the distances between non-overlapping patches $\Pi_D(x_{i_j})$ and x_{i_j}.

Then, from (A.12) and (A.13)

$$\begin{aligned}\|\Pi_{D\to N}(X)-X\|^2&\le\sum_j\|\Pi_N(X)|_{\mathcal{N}(p_{i_j})}-x_{i_j}\|^2=\\&=\|\Pi_N(X)-X\|^2,\end{aligned} \tag{A.14}$$

where again the second equality follows by the fact that patches x_{i_j} do not overlap.

[1] Such a decomposition always exists assuming that we can pad images with zeros.

However, since $\mathcal{M}_D$ is complete and $\{\Pi_D(x_i)\}_i$ is consistent, $\Pi_{D\to N}(X) \in \mathcal{M}_N$. In fact $\Pi_{D\to N}(X)$ is given by the average of the projections $\Pi_D(x_i)$ satisfying definition (A.10). Therefore, since $\Pi_N(X)$ is defined in (A.2) as the point of minimum distance to X in $\mathcal{M}_N$, we have

$$\|\Pi_N(X) - X\|^2 \leq \|\Pi_{D\to N}(X) - X\|^2. \tag{A.15}$$

From (A.14) and (A.15) we have the thesis. $\square$

Notice that the thesis of Theorem 2 tells us that the distance between our approximation $\Pi_{D\to N}(X)$ and X is as good as the distance between X and $\Pi_N(X)$. The equivalence of the two projections is given assuming that there is a unique minimum of the distance to X in $\mathcal{M}_N$. This is stated in the following

Corollary 1. *In the hypothesis of Theorem 2, if the function $F(Y) = \|Y - X\|$ has a unique minimum in $\mathcal{M}_N$, we have*

$$\Pi_N(X) = \Pi_{D\to N}(X). \tag{A.16}$$

Proof. The thesis follows by (A.11) and the hypothesis. $\square$

In the proof of Theorem 2, we have considered the averaging size R to be equal to the projection size D. The generalization to the case $R \leq D$ is straightforward. In fact, in this case we only have to consider the intersections in Definition 1 to be $y_i(p) = y_j(p)$ for all $p \in \mathcal{N}_R(p_i) \cap \mathcal{N}_R(p_j)$, and in Definition 2 to be $\Pi_D(x_i)(p) = \Pi_D(x_j)(p)$ for all $p \in \mathcal{N}_R(p_i) \cap \mathcal{N}_R(p_j)$. Moreover, in the proof of the theorem, the disjoint partition $\{x_{i_j}\}_{j=1}^K$ should be such that such that $\mathcal{N}_R(p_{i_{j_1}}) \cap \mathcal{N}_R(p_{i_{j_2}}) = \emptyset$. Notice that these conditions are weaker than the case $R = D$, making our theorem more general.

However, also in the case $R \leq D$, in real applications only limited training patches are available and the hypothesis of completeness might not be satisfied. Also consistency will not hold in general and projections in the intersection of overlapping patches will not be exactly the same.

Nevertheless, in the next section we show that by relaxing the hypothesis of Theorem 2 and assuming only approximated projections, we can prove that the error committed by our method is within a certain bound to the optimal solution. This bound is directly related to the error committed by the projections on the patches $\Pi_D(x_i)$ and the size of our training set.

A.1.3 Approximated Projection

In this section, we consider the case of approximated projection. We start by relaxing the hypothesis of completeness and supposing that only a smaller subset $\mathcal{M}_D$ of all the possible training patches is available.

More precisely, let $\bar{\mathcal{M}}_D$ the set of all patches y_i of size D of images dY in $\mathcal{M}_N$, assume that $\bar{\mathcal{M}}_D$ is complete in $\mathcal{M}_N$[2] and that the set of available training patches $\mathcal{M}_D$ is included in $\bar{\mathcal{M}}_D$, $\mathcal{M}_D \subseteq \bar{\mathcal{M}}_D$. For a patch x, we denote $\Pi_{\bar{D}}(x)$ the projection of x onto $\bar{\mathcal{M}}_D$.

For $\epsilon \geq 0$, we say that $\mathcal{M}_D$ is $\epsilon-$*complete* in $\mathcal{M}_N$, if for all $y \in \mathcal{M}_D$ there exists $\bar{y} \in \bar{\mathcal{M}}_D$ such that $\|y - \bar{y}\|^2 \leq \epsilon$. This means that every ground truth patch $\bar{y}$ is close, up to an error of ε, to an available training patch y. This hypothesis will replace the hypothesis of completeness of Theorem 2.

The hypothesis of consistency will be replaced by the following:

$$\exists \epsilon_1 \geq 0 \text{ s.t. } \forall x_i \; \|\Pi_{D\to N}(X)|_{\mathcal{N}(p_i)} - \Pi_D(x_i)\|^2 \leq \epsilon_1, \tag{A.17}$$

$$\exists \epsilon_2 \geq 0 \text{ s.t. } \forall x_i \; \|\Pi_N(X)|_{\mathcal{N}(p_i)} - \Pi_{\bar{D}}(x_i)\|^2 \leq \epsilon_2. \tag{A.18}$$

This means that the restriction to $\mathcal{N}(p_i)$ of the images $\Pi_{D\to N}(X)$ and $\Pi_N(X)$ are close to the projections of patch x_i onto $\mathcal{M}_D$ and $\bar{\mathcal{M}}_D$ respectively. Note that if $\{\Pi_D(x_i)\}_i$ is consistent, then $\epsilon_1 = 0$ and if $\{\Pi_{\bar{D}}(x_i)\}_i$ is consistent, then $\epsilon_2 = 0$.

We now have the following

Theorem 3. *If $\mathcal{M}_D$ is $\epsilon-$complete in $\mathcal{M}_N$ and if* (A.17) *and* (A.18) *hold. Then,*

$$-\frac{N}{D}(\epsilon_1 + \epsilon) \leq \|\Pi_N(X) - X\|^2 - \|\Pi_{D\to N}(X) - X\|^2 \leq \frac{N}{D}(\epsilon_1 + \epsilon_2). \tag{A.19}$$

Proof. Let $\{x_{i_j}\}_j$ be a disjoint partition of the image, as in the proof of Theorem 2. Then,

$$\begin{aligned}\|\Pi_N(X) - X\|^2 &= \sum_j \|\Pi_N(X)|_{\mathcal{N}(p_{i_j})} - x_{i_j}\|^2 \leq \\ &\leq \sum_j \left(\|\Pi_N(X)|_{\mathcal{N}(p_{i_j})} - \Pi_{\bar{D}}(x_{i_j})\|^2 + \|\Pi_{\bar{D}}(x_{i_j}) - x_{i_j}\|^2\right) \leq \\ &\leq \sum_j \left(\epsilon_2 + \|\Pi_{\bar{D}}(x_{i_j}) - x_{i_j}\|^2\right) = \\ &= \frac{N}{D}\epsilon_2 + \sum_j \|\Pi_{\bar{D}}(x_{i_j}) - x_{i_j}\|^2.\end{aligned} \tag{A.20}$$

[2]If $\bar{\mathcal{M}}_D$ is not complete, we can extend $\mathcal{M}_N$ by adding to it all images that can be obtained with Eq. (A.7) from patches in $\bar{\mathcal{M}}_D$ that coincide in their supports.

Where the first inequality is the triangle inequality and the second inequality is given by hypothesis (A.18).

Since $\mathcal{M}_D \subseteq \bar{\mathcal{M}}_D$, for every x_i, $\|\Pi_{\bar{D}}(x_i) - x_i\|^2 \leq \|\Pi_D(x_i) - x_i\|^2$. In fact, $\Pi_{\bar{D}}(x_i)$ is the point of minimum distance to x_i, computed on a larger set $\bar{\mathcal{M}}_D$ compared to $\Pi_D(x_i)$.

Then, Eq.(A.20) becomes

$$\|\Pi_N(X) - X\|^2 \leq \frac{N}{D}\epsilon_2 + \sum_j \|\Pi_D(x_{i_j}) - x_{i_j}\|^2. \tag{A.21}$$

By using the triangle inequality and hypothesis (A.17), Eq. (A.21) becomes

$$\begin{aligned}
\|\Pi_N(X) - X\|^2 &\leq \frac{N}{D}\epsilon_2 + \sum_j \Big(\|\Pi_D(x_{i_j}) - \Pi_{D\to N}(X)|_{\mathcal{N}_D(p_{i_j})}\|^2 \\
&\quad + \|\Pi_{D\to N}(X)|_{\mathcal{N}_D(p_{i_j})} - x_{i_j}\|^2\Big) \leq \\
&\leq \frac{N}{D}\epsilon_2 + \sum_j \Big(\epsilon_1 + \|\Pi_{D\to N}(X)|_{\mathcal{N}_D(p_{i_j})} - x_{i_j}\|^2\Big) = \\
&= \frac{N}{D}(\epsilon_2 + \epsilon_1) + \sum_j \|\Pi_{D\to N}(X)|_{\mathcal{N}_D(p_{i_j})} - x_{i_j}\|^2 = \\
&= \frac{N}{D}(\epsilon_2 + \epsilon_1) + \|\Pi_{D\to N}(X) - X\|^2.
\end{aligned} \tag{A.22}$$

The inequality in (A.22) proves the right hand side in (A.19). For the left hand side we proceed analogously:

$$\begin{aligned}
\|\Pi_{D\to N}(X) - X\|^2 &= \sum_j \|\Pi_{D\to N}(X)|_{\mathcal{N}_D(p_{i_j})} - x_{i_j}\|^2 \leq \\
&\leq \sum_j \Big(\|\Pi_{D\to N}(X)|_{\mathcal{N}_D(p_{i_j})} - \Pi_D(x_{i_j})\|^2 + \|\Pi_D(x_{i_j}) - x_{i_j}\|^2\Big) \leq \\
&\leq \frac{N}{D}\epsilon_1 + \sum_j \|\Pi_D(x_{i_j}) - x_{i_j}\|^2.
\end{aligned} \tag{A.23}$$

Where we first used triangle inequality and then hypothesis (A.17).

Since $\Pi_N(X)|_{\mathcal{N}_D(p_i)} \in \bar{\mathcal{M}}_D$ for all p_i, and since $\mathcal{M}_D$ is $\epsilon-$complete, $\forall p_i$ there exists $y_i \in \mathcal{M}_D$ such that $\|y_i - \Pi_N(X)|_{\mathcal{N}(p_i)}\|^2 \leq \epsilon$. Moreover, for all $y_i \in \mathcal{M}_D$, $\|\Pi_D(x_i) - x_i\|^2 \leq \|y_i - x_i\|^2$. In fact, $\Pi_D(x_i)$ is the point of minimum distance to x_i in $\mathcal{M}_D$. Hence, substituting in (A.23) we have

$$\begin{aligned}
\|\Pi_{D\to N}(X) - X\|^2 &\leq \frac{N}{D}\epsilon_1 + \sum_j \|\Pi_D(x_{i_j}) - x_{i_j}\|^2 \leq \\
&\leq \frac{N}{D}\epsilon_1 + \sum_j \|y_{i_j} - x_{i_j}\|^2 \leq \\
&\leq \frac{N}{D}\epsilon_1 + \sum_j \Big(\|y_{i_j} - \Pi_N(X)|_{\mathcal{N}_D(p_{i_j})}\|^2 + \\
&\quad + \|\Pi_N(X)|_{\mathcal{N}_D(p_{i_j})} - x_{i_j}\|^2\Big) \leq \\
&\leq \frac{N}{D}\epsilon_1 + \sum_j \Big(\epsilon + \|\Pi_N(X)|_{\mathcal{N}_D(p_{i_j})} - x_{i_j}\|^2\Big) = \\
&= \frac{N}{D}(\epsilon_1 + \epsilon) + \|\Pi_N(X) - X\|^2.
\end{aligned} \tag{A.24}$$

Equation (A.24) proves the left hand side of (A.19) and this ends the proof. □

Bibliography

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda. A Scalable Optimization Approach for Fitting Canonical Tensor Decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011.

[2] G. Agam and C. Wu. Probabilistic Modeling-Based Vessel Enhancement in Thoracic CT Scans. In *CVPR*, volume 2, pages 649–654, 2005.

[3] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *TSP*, 54(11):4311–4322, 2006.

[4] A. Aibinu, M. Iqbal, A. Shafie, M. Salami, and M. Nilsson. Vascular Intersection Detection in Retina Fundus Images Using a New Hybrid Approach. *Computers in Biology and Medicine*, 40(1):81–89, 2010.

[5] K. Al-Kofahi, S. Lasek, D. Szarowski, C. Pace, G. Nagy, J. Turner, and B. Roysam. Rapid Automated Three-Dimensional Tracing of Neurons from Confocal Image Stacks. *TITB*, 6(2):171–187, 2002.

[6] P. Alivisatos, M. Chun, G. Church, R. Greenspan, M. Roukes, and R. Yuste. The Brain Activity Map Project and the Challenge of Functional Connectomics. *Neuron*, 74(6):970–974, 2012.

[7] M. Allan., S. Ourselin., S. Thompson., D. Hawkes, J. Kelly, and D. Stoyanov. Toward Detection and Localization of Instruments in Minimally Invasive Surgery. *TBE*, 60(4):1050–1058, 2013.

[8] B. Andres, U. Koethe, M. Helmstaedter, W. Denk, and F. Hamprecht. Segmentation of SBFSEM Volume Data of Neural Tissue by Hierarchical Classification. In *DAGM*, pages 142–152, 2008.

[9] B. Andres, T. Kröger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Köthe, and F. A. Hamprecht. Globally Optimal Closed-Surface Segmentation for Connectomics. *ECCV*, pages 778–791, 2012.

[10] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *PAMI*, 33(5):898–916, 2011.

[11] P. Arbelaez, J. Pont-tuset, J. Barron, F. Marqués, and J. Malik. Multiscale Combinatorial Grouping. In *CVPR*, pages 328–335, 2014.

[12] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, T. Liu, M. Seyedhosseini, T. Tasdizen, L. Kamentsky, R. Burget, V. Uher, X. Tan, C. Sun, T. Pham, E. Bas, M. G. Uzunbas, A. Cardona, J. Schindelin, and H. S. Seung. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9, 2015.

[13] G. Ascoli, K. Svoboda, and Y. Liu. Digital Reconstruction of Axonal and Dendritic Morphology DIADEM Challenge, 2010.

[14] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with Sparsity-Inducing Penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[15] P. Bakker, L. Van Vliet, and P. Verbeek. Analysis and detection of 3-d curvilinear structures. In *7th Annual Conference of the Advanced School for Computing and Imaging, Heijen, The Netherlands, May 30-June 1, 2000*. ASCI, 2001.

[16] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[17] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The Generalized PatchMatch Correspondence Algorithm. In *ECCV*, pages 29–43, 2010.

[18] E. Bas and D. Erdogmus. Principal Curves as Skeletons of Tubular Objects - Locally Characterizing the Structures of Axons. *Neur. Inf.*, 9(2-3):181–191, 2011.

[19] E. Bas, D. Erdogmus, R. Draft, and J. Lichtman. Local Tracing of Curvilinear Structures in Volumetric Color Images: Application to the Brainbow Analysis. *Journal of Visual Communication and Image Representation*, 23(8):1260–1271, 2012.

[20] C. Bauckhage. Tensor-Based Filter Design Using Kernel Ridge Regression. In *ICIP*, volume 4, pages IV–45, 2007.

[21] C. Bauckhage, T. Käster, and J. K. Tsotsos. Applying Ensembles of Multilinear Classifiers in the Frequency Domain. In *CVPR*, volume 1, pages 95–102, 2006.

[22] C. Becker, K. Ali, G. Knott, and P. Fua. Learning Context Cues for Synapse Segmentation. *TMI*, 32(10):1864–1877, 2013.

[23] C. Becker, R. Rigamonti, V. Lepetit, and P. Fua. Supervised Feature Learning for Curvilinear Structure Segmentation. In *MICCAI*, September 2013.

[24] Y. Bengio. *Learning Deep Architectures for AI*, volume 2. Now Publishers, 2009.

[25] F. Benmansour and L. Cohen. Tubular Structure Segmentation Based on Minimal Path Method and Anisotropic Enhancement. *IJCV*, 92(2):192–210, 2011.

[26] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, pages 4380–4389, 2015.

[27] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[28] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning Mid-Level Features for Recognition. In *CVPR*, pages 2559–2566, 2010.

[29] D. Breitenreicher, M. Sofka, S. Britzen, and S. Zhou. Hierarchical Discriminative Framework for Detecting Tubular Structures in 3D Images. In *International Conference on Information Processing in Medical Imaging*, pages 328–339, 2013.

[30] H. Bristow, A. Eriksson, and S. Lucey. Fast Convolutional Sparse Coding. In *CVPR*, pages 391–398, 2013.

[31] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature Verification Using a Siamese Time Delay Neural Network. In *NIPS*, 1993.

[32] J. Bruna and S. Mallat. Invariant scattering convolution networks. *PAMI*, 35(8):1872–1886, 2013.

[33] D. J. Bumbarger, M. Riebesell, C. Rödelsperger, and R. J. Sommer. System-wide rewiring underlies behavioral differences in predatory and bacterial-feeding nematodes. *Cell*, 152(1):109–119, 2013.

[34] J. Canny. A Computational Approach to Edge Detection. *PAMI*, (6):679–698, 1986.

[35] A. Cardona, S. Saalfeld, S. Preibisch, B. Schmid, A. Cheng, J. Pulokas, P. Tomancak, and V. Hartenstein. An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. *PLoS Biol*, 8(10):e1000502, 2010.

[36] H. Chen, H. Xiao, T. Liu, and H. Peng. Smarttracing: Self-Learning-Based Neuron Reconstruction. *Brain Informatics*, (3):135–144, 2015.

[37] P. Chothani, V. Mehta, and A. Stepanyants. Automated Tracing of Neurites from Light Microscopy Stacks of Images. *Neur. Inf.*, 9(2-3):263–278, 2011.

[38] D. Cireşan, A. Giusti, L. Gambardella, and J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In *NIPS*, pages 2843–2851, 2012.

[39] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-Column Deep Neural Networks for Image Classification. In *CVPR*, pages 3642–3649, 2012.

[40] A. Coates and A. Ng. The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization. In *ICML*, pages 921–928, 2011.

[41] G. Conte and P. Doherty. An Integrated UAV Navigation System Based on Aerial Image Matching. In *IEEE Aerospace Conference*, pages 1–10, 2008.

[42] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[43] A. Criminisi, P. Perez, and K. Toyama. Region Filling and Object Removal by Exemplar-Based Image Inpainting. *TIP*, 13(9):1200–1212, 2004.

[44] K. Dabov, A. Foi, and V. Katkovnik. Image Denoising by Sparse 3D Transformation-Domain Collaborative Filtering. *JMLR*, 16(8):2080–2095, 2007.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *CVPR*, pages 248–255, 2009.

[46] M. Denil, B. Shakibi, L. Dinh, and N. de Freitas. Predicting Parameters in Deep Learning. In *NIPS*, 2013.

[47] W. Denk and H. Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biol*, 2(11):e329, 2004.

[48] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[49] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[50] R. Deriche. Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector. *IJCV*, 1(2):167–187, 1987.

[51] X. Descombes, G. Malandain, C. Fonta, L. Negyessy, and R. Mosko. Automatic dendrite spines detection from x-ray tomography volumes. In *IEEE 10th International Symposium on Biomedical Imaging (ISBI)*, pages 436–439, 2013.

[52] P. Dollar, Z. Tu, and S. Belongie. Supervised Learning of Edges and Object Boundaries. In *CVPR*, volume 2, pages 1964–1971, 2006.

[53] P. Dollár and C. L. Zitnick. Structured Forests for Fast Edge Detection. In *ICCV*, pages 1841–1848, 2013.

[54] P. Dollár and C. L. Zitnick. Fast Edge Detection Using Structured Forests. *PAMI*, 37(8):1558–1570, 2015.

[55] R. Duda and P. Hart. Use of the Hough Transform to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[56] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[57] O. Dzyubak and E. Ritman. Automation of Hessian-Based Tubularity Measure Response Function in 3D Biomedical Images. *IJBI*, 2011:7, 2011.

[58] M. Elad and M. Aharon. Image Denoising via Sparse and Redundant Representations over Learned Dictionaries. *TIP*, 15(12):3736–3745, 2006.

[59] D. V. Essen, K. Ugurbil, E. Auerbach, D. Barch, T. Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S. Curtiss, S. D. Penna, D. Feinberg, M. Glasser, N. Harel, A. Heath, L. Larson-Prior, D. Marcus, G. Michalareas, S. Moeller, R. Oostenveld, S. Petersen, F. Prior, B. Schlaggar, S. Smith, A. Snyder, J. Xu, and E. Yacoub. The human connectome project: A data acquisition perspective. *NeuroImage*, 62(4):2222–2231, 2012.

[60] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello. Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems. In *International Symposium on Circuits and Systems*, pages 257–260, 2010.

[61] M. Fazel, H. Hindi, and S. Boyd. A Rank Minimization Heuristic with Application to Minimum Order System Approximation. In *ACC*, volume 6, pages 4734–4739, 2001.

[62] M. Fink and P. Perona. Mutual Boosting for Contextual Inference. In *NIPS*, pages 1515–1522, 2004.

[63] M. Fischler, J. Tenenbaum, and H. Wolf. Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique. *CVGIP*, 15(3):201–223, 1981.

[64] M. Fischler and H. Wolf. Linear Delineation. In *CVPR*, 1983.

[65] M. A. Fischler and H. C. Wolf. A general approach to machine perception of linear, structure in imaged data. Technical report, AI Center, SRI International, 1983.

[66] A. H. Foruzan, R. A. Zoroofi, Y. Sato, and M. Hori. A Hessian-Based Filter for Vascular Segmentation of Noisy Hepatic CT Scans. *International Journal of Computer Assisted Radiology and Surgery*, 7(2):199–205, 2012.

[67] A. Frangi, W. Niessen, K. Vincken, and M. Viergever. Multiscale Vessel Enhancement Filtering. *Lecture Notes in Computer Science*, pages 130–137, 1998.

[68] W. Freeman and E. Adelson. The Design and Use of Steerable Filters. *PAMI*, 13(9):891–906, 1991.

[69] Y. Freund and R. Schapire. A Short Introduction to Boosting, 1999. Journal of Japanese Society for Artificial Intelligence.

[70] J. Friedman. Stochastic Gradient Boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[71] H. Fu, Y. Xu, D. W. K. Wong, and J. Liu. Retinal vessel segmentation via deep learning network and fully-connected conditional random fields. In *IEEE 13th International Symposium on Biomedical Imaging, ISBI*, 2016.

[72] J. Funke, D. Andres, F. A. Hamprecht, A. Cardona, and M. Cook. Efficient Automatic 3D-Reconstruction of Branching Neurons from EM Data. In *CVPR*, pages 1004–1011, 2012.

[73] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114(6):712–722, 2010.

[74] Y. Ganin and V. Lempitsky. n^4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms. In *ACCV*, pages 536–551, 2014.

[75] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber. Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks. In *ICIP*, 2013.

[76] G. Gonzalez, F. Aguet, F. Fleuret, M. Unser, and P. Fua. Steerable Features for Statistical 3D Dendrite Detection. In *MICCAI*, pages 625–632, 2009.

[77] G. Gonzalez, F. Fleuret, and P. Fua. Learning Rotational Features for Filament Detection. In *CVPR*, pages 1582–1589, 2009.

[78] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[79] D.-Y. Gu, C.-F. Zhu, J. Guo, S.-X. Li, and H.-X. Chang. Vision-aided uav navigation using gis data. In *IEEE International Conference on Vehicular Electronics and Safety*, pages 78–82, 2010.

[80] L. Gu and L. Cheng. Learning to boost filamentary structure segmentation. In *CVPR*, pages 639–647, 2015.

[81] S. Hallman and C. C. Fowlkes. Oriented Edge Forests for Boundary Detection. In *CVPR*, pages 1732–1740, 2015.

[82] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, 2009.

[83] S. Hawe, M. Seibert, and M. Kleinsteuber. Separable Dictionary Learning. In *CVPR*, pages 438–445, 2013.

[84] B. He, T. Coleman, et al. Grand Challenges in Mapping the Human Brain: NSF Workshop Report. *TBE*, 60(11):2983–2992, 2013.

[85] K. He, X. Zhang, R. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In *ICCV*, pages 1026–1034, 2015.

[86] G. Heitz and D. Koller. Learning Spatial Context: Using Stuff to Find Things. In *ECCV*, pages 30–43, 2008.

[87] E. M. Hillman. Optical brain imaging in vivo: techniques and applications from animal to man. *Journal of biomedical optics*, 12(5):051402–051402, 2007.

[88] G. Hinton. Learning to Represent Visual Input. *Philosophical Transactions of the Royal Society*, 365(1537):177–184, 2010.

[89] P. Hough and P. V. C. Method and Means for Recognizing Complex Patterns. 1962.

[90] X. Huang and L. Zhang. Road Centreline Extraction from High-Resolution Imagery Based on Multiscale Structural Features and Support Vector Machines. *International Journal of Remote Sensing*, 30(8):1977–1987, 2009.

[91] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. *arXiv preprint arXiv:1504.01989*, 2015.

[92] ISBI Challenge: Segmentation of neuronal structures in EM stacks. http://brainiac2.mit.edu/isbi_challenge, 2012.

[93] M. Jacob and M. Unser. Design of Steerable Filters for Feature Detection Using Canny-Like Criteria. *PAMI*, 26(8):1007–1019, 2004.

[94] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[95] V. Jain, B. Bollmann, M. Richardson, D. Berger, M. Helmstaedter, K. Briggman, W. Denk, J. Mendenhall, W. Abraham, K. harris, N. Kasthuri, K. Hayworth, R. Schalek, J. Tapia, J. Lichtman, and H. Seung. Boundary Learning by Optimization with Topological Constraints. In *CVPR*, pages 2488–2495, 2010.

[96] V. Jain, J. Murray, F. Roth, S. Turaga, V. Zhigulin, K. Briggman, M. Helmstaedter, W. Denk, and H. Seung. Supervised Learning of Image Restoration with Convolutional Networks. In *ICCV*, pages 1–8, 2007.

[97] V. Jain, H. S. Seung, and S. Turaga. Machines That Learn to Segment Images: A Crucial Technology for Connectomics. *Current Opinion in Neurobiology*, 20(5):653–666, 2010.

[98] T. A. Jarrell, Y. Wang, A. E. Bloniarz, C. A. Brittin, M. Xu, J. N. Thomson, D. G. Albertson, D. H. Hall, and S. W. Emmons. The connectome of a decision-making neural network. *Science*, 337(6093):437–444, 2012.

[99] H. F. Jelinek, M. J. Cree, J. J. Leandro, J. V. Soares, R. M. Cesar, and A. Luckie. Automated segmentation of retinal blood vessels and identification of proliferative diabetic retinopathy. *JOSA A*, 24(5):1448–1456, 2007.

[100] E. Jurrus, A. Paiva, S. Watanabe, J. Anderson, R. Whitaker, B. Jones, R. Marc, and T. Tasdizen. Detection of Neuron Membranes in Electron Microscopy Images Using a Serial Neural Network Architecture. *MIA*, 14(6):770–783, 2010.

[101] K. K. Rein, M. Zöckler, M. M. T, C. Grübel, and M. Heisenberg. The Drosophila Standard Brain. *Current Biology*, 12(3):227–231, 2002.

[102] P. Kainz, M. Urschler, S. Schulter, P. Wohlhart, and V. Lepetit. You should use regression to detect cells. In *MICCAI*, pages 276–283. Springer, 2015.

[103] H.-P. Kang and C.-R. Lee. Improving performance of convolutional neural networks by separable filters on gpu. In *European Conference on Parallel Processing*, pages 638–649. Springer, 2015.

[104] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning Convolutional Feature Hierarchies for Visual Recognition. In *NIPS*, pages 1090–1098, 2010.

[105] W. Kienzle, G. Bakir, M. Franz, and B. Schölkopf. Face Detection – Efficient and Rank Deficient. In *NIPS*, pages 673–680, 2005.

[106] T. G. Kolda and B. W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, 2009.

[107] P. Kontschieder, S. Bulo, H. Bischof, and M. Pelillo. Structured Class-Labels in Random Forests for Semantic Image Labelling. In *ICCV*, pages 2190–2197, 2011.

[108] P. Kontschieder, S. R. Bulò, M. Donoser, M. Pelillo, and H. Bischof. Semantic Image Labelling as a Label Puzzle Game. In *BMVC*, pages 1–12, 2011.

[109] L. Korbo, B. Pakkenberg, O. Ladefoged, H. Gunderson, P. Arlien-Soborg, and H. Pakkenberg. An Efficient Method for Estimating the Total Number of Neurons in the Rat Brain Cortex. *Journal of Neuroscience Methods*, 31(2):93–100, 1990.

[110] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, and Y. Trousset. Model-Based Detection of Tubular Structures in 3D Images. *CVIU*, 80(2):130–171, 2000.

[111] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, pages 1097–1105, 2012.

[112] M. Kummu, X. Lu, A. Rasphone, J. Sarkkula, and J. Koponen. Riverbank changes along the mekong river: remote sensing detection in the vientiane–nong khai area. *Quaternary International*, 186(1):100–112, 2008.

[113] M. Law and A. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *ECCV*, pages 368–382, 2008.

[114] M. Law and A. Chung. An Oriented Flux Symmetry Based Active Contour Model for Three Dimensional Vessel Segmentation. In *ECCV*, pages 720–734, 2010.

[115] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[116] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *PIEEE*, 1998.

[117] Y. LeCun and C. Cortes. MNIST Handwritten Digit Database, 2010.

[118] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *ICML*, pages 609–616, 2009.

[119] P. Lee, C. Chuang, A. Chiang, and Y. Ching. High-Throughput Computer Method for 3D Neuronal Structure Reconstruction from the Image Stack of the Drosophila Brain and Its Applications. *PLoS Comput Biol*, 8(9):e1002658, 2012.

[120] Q. Li, B. Feng, L. Xie, P. Liang, H. Zhang, and T. Wang. A cross-modality learning approach for vessel segmentation in retinal images. *TMI*, 35(1):109–118, 2016.

[121] J. Lim, C. L. Zitnick, and P. Dollár. Sketch Tokens: A Learned Mid-Level Representation for Contour and Object Detection. In *CVPR*, pages 3158–3165, 2013.

[122] E. Limonova, A. Sheshkus, and D. Nikolaev. Computational optimization of convolutional neural networks using separated filters architecture. *International Journal of Applied Engineering Research*, 11(11):7491–7494, 2016.

[123] T. Lindeberg. Scale-Space for Discrete Signals. *PAMI*, 12(3):234–254, 1990.

[124] J. Livet, T. Weissman, H. Kang, R. Draft, J. Lu, R. Bennis, J. Sanes, and J. Lichtman. Transgenic Strategies for Combinatorial Expression of Fluorescent Proteins in the Nervous System. *Nature*, 450(7166):56–62, 2007.

[125] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, pages 3431–3440, 2015.

[126] A. Lucchi, C. Becker, P. Marquez-neila, and P. Fua. Exploiting Enclosing Membranes and Contextual Cues for Mitochondria Segmentation. In *MICCAI*, pages 65–72, 2014.

[127] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative Learned Dictionaries for Local Image Analysis. In *CVPR*, pages 1–8, 2008.

[128] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-Local Sparse Models for Image Restoration. In *ICCV*, pages 2272–2279, 2009.

[129] M. Maire, S. X. Yu, and P. Perona. Hierarchical Scene Annotation. In *BMVC*, 2013.

[130] F. Mamalet and C. Garcia. Simplifying Convnets for Fast Learning. In *ICANN*, pages 58–65, 2012.

[131] D. Marin, Y. Zhong, M. Drangova, and Y. Boykov. Thin structure estimation with curvature regularization. In *ICCV*, pages 397–405, 2015.

[132] H. Markram. The Blue Brain Project. *Nature Reviews Neuroscience*, 7(2):153–160, 2006.

[133] H. Markram. The human brain project. *Scientific American*, 306(6):50–55, 2012.

[134] H. A. Marquering, J. Dijkstra, P. J. de Koning, B. C. Stoel, and J. H. Reiber. Towards quantitative analysis of coronary CTA. *The international journal of cardiovascular imaging*, 21(1):73–84, 2005.

[135] D. Marr and E. Hildreth. Theory of Edge Detection. *Proceedings of the Royal Society of London, Biological Sciences*, 207(1167):187–217, 1980.

[136] D. Martin, C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color and Texture Cues. *PAMI*, 26(5):530–549, 2004.

[137] A. Martinez-Sanchez, I. Garcia, and J. Fernandez. A Ridge-Based Framework for Segmentation of 3D Electron Microscopy Datasets. *Journal of Structural Biology*, 181(1):61–70, 2013.

[138] C. R. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *PAMI*, 25(2):265–270, 2003.

[139] E. Meijering, M. Jacob, J.-C. F. Sarria, P. Steiner, H. Hirling, and M. Unser. Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images. *Cytometry Part A*, 58(2):167–176, 2004.

[140] M. Meilă. Comparing Clusterings - An Information Based Distance. *JMVA*, 98(5):873–895, 2007.

[141] G. Michelin, L. Guignard, U.-M. Fiuza, and G. Malandain. Embryo cell membranes reconstruction by tensor voting. In *IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 1259–1262, 2014.

[142] K. D. Micheva and S. J. Smith. Array tomography: A new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 55(1):25–36, 2007.

[143] H. Mirzaalian, T. Lee, and G. Hamarneh. Hair Enhancement in Dermoscopic Images Using Dual-Channel Quaternion Tubularness Filters and MRF-Based Multi-Label Optimization. *TIP*, 23(12):5486–5496, 2014.

[144] V. Mnih and G. Hinton. Learning to Detect Roads in High-Resolution Aerial Images. In *ECCV*, pages 210–223, 2010.

[145] V. Mnih and G. Hinton. Learning to Label Aerial Images from Noisy Data. In *ICML*, pages 567–574, 2012.

[146] K. Mosaliganti, F. Janoos, A. Gelas, R. Noche, N. Obholzer, R. Machiraju, and S. Megason. Anisotropic Plate Diffusion Filtering for Detection of Cell Membranes in 3D Microscopy Images. In *ICBI*, pages 588–591, 2010.

[147] M. Muja and D. G. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *PAMI*, 36(11):2227–2240, 2014.

[148] A. Narayanaswamy, Y. Wang, and B. Roysam. 3D Image Pre-Processing Algorithms for Improved Automated Tracing of Neuronal Arbors. *Neur. Inf.*, 9(2-3):219–231, 2011.

[149] W. Neuenschwander, P. Fua, G. Székely, and O. Kubler. Initializing Snakes. In *CVPR*, pages 658–663, 1994.

[150] B. Olshausen and D. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[151] R. B. Palm. Prediction as a Candidate for Learning Deep Hierarchical Models of Data. Master's thesis, Technical University of Denmark, 2012.

[152] M. Pechaud, G. Peyré, and R. Keriven. Extraction of Tubular Structures over an Orientation Domain. In *CVPR*, pages 336–342, 2009.

[153] P. Perona. Deformable Kernels for Early Vision. *PAMI*, 17(5):488–499, 1995.

[154] P. Perona and J. Malik. Scale Space and Edge Detection Using Anisotropic Diffusion. In *IEEE Computer Society Workshop on Computer Vision*, 1987.

[155] W. Perry, A. Broers, F. El-Baz, W. Harris, B. Healy, and W. D. H. et al. Grand Challenges for Engineering. *National Academy of Engineering*, 2008.

[156] H. Pirsiavash and D. Ramanan. Steerable Part Models. In *CVPR*, pages 3226–3233, 2012.

[157] M. Pool, J. Thiemann, A. Bar-Or, and A. E. Fournier. Neuritetracer: A Novel Imagej Plugin for Automated Quantification of Neurite Outgrowth. *Journal of Neuroscience Methods*, 168(1):134–139, 2008.

[158] J. Porway, K. Wang, and S. Zhu. A Hierarchical and Contextual Model for Aerial Image Understanding. In *CVPR*, pages 1–8, 2008.

[159] L. Quam. Road Tracking and Anomaly Detection. In *DARPA IUW*, 1978.

[160] X. Ren and L. Bo. Discriminatively Trained Sparse Code Gradients for Contour Detection. In *NIPS*, pages 584–592, 2012.

[161] R. Rigamonti, M. Brown, and V. Lepetit. Are Sparse Representations Really Relevant for Image Classification? In *CVPR*, pages 1545–1552, 2011.

[162] R. Rigamonti and V. Lepetit. Accurate and Efficient Linear Structure Segmentation by Leveraging Ad Hoc Features with Learned Filters. In *MICCAI*, pages 189–197, 2012.

[163] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua. Learning Separable Filters. In *CVPR*, pages 2754–2761, 2013.

[164] R. Rigamonti, E. Türetken, G. González, P. Fua, and V. Lepetit. Filter Learning for Linear Structure Segmentation. Technical report, EPFL, 2011.

[165] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. 2015.

[166] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *J. ACM*, 13(4):471–494, 1966.

[167] R. Rubinstein, M. Zibulevsky, and M. Elad. Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation. *TSP*, 58(3):1553–1564, 2010.

[168] J. Salmon and Y. Strozecki. Patch Reprojections for Non Local Methods. *Signal Processing*, 92(2):477–489, 2012.

[169] A. Santamaría-Pang, T. Bildea, C. M. Colbert, P. Saggau, and I. Kakadiaris. Towards Segmentation of Irregular Tubular Structures in 3D Confocal Microscope Images. In *MICCAI Workshop in Microscopic Image Analysis and Applications in Biology*, pages 78–85, 2006.

[170] A. Santamaría-Pang, C. Colbert, P. Saggau, and I. Kakadiaris. Automatic Centerline Extraction of Irregular Tubular Structures Using Probability Volumes from Multiphoton Imaging. In *MICCAI*, 2007.

[171] Y. Sato, S. Nakajima, H. Atsumi, T. Koller, G. Gerig, S. Yoshida, and R. Kikinis. 3D Multi-Scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images. *MIA*, pages 213–222, 1998.

[172] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Tissue Classification Based on 3D Local Intensity Structure for Volume Rendering. *IEEE Trans. on Visualization and Computer Graphics*, 6(2):160–180, 2000.

[173] B. Savas and L. Eldén. Handwritten Digit Classification Using Higher Order Singular Value Decomposition. *PR*, 40(3):993–1003, 2007.

[174] M. Schaap, C. Metz, T. van Walsum, A. V. der Giessen, A. Weustink, N. Mollet, C. Bauer, H. Bogunovi, C. Castro, X. Deng, E. Dikici, T. O'Donnell, M. Frenay, O. Friman, M. H. Hoyos, P. Kitslaar, K. Krissian, C. Kuhnel, M. A. Luengo-Oroz, M. Orkisz, O. Smedby, M. Styner, A. Szymczak, H. Tek, C. Wang, S. K. Warfield, S. Zambal, Y. Zhang, G. Krestin, and W. Niessen. Standardized Evaluation Methodology and Reference Database for Evaluating Coronary Artery Centerline Extraction Algorithms. *MIA*, 13(5):701–714, 2009.

[175] M. Seyedhosseini, R. Kumar, E. Jurrus, R. Guily, M. Ellisman, H. Pfister, and T. Tasdizen. Detection of Neuron Membranes in Electron Microscopy Images Using Multi-Scale Context and Radon-Like Features. In *MICCAI*, pages 670–677, 2011.

[176] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen. Image Segmentation with Cascaded Hierarchical Models and Logistic Disjunctive Normal Networks. In *ICCV*, pages 2168–2175, 2013.

[177] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, pages 3982–3991, 2015.

[178] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

[179] A. Sironi, V. Lepetit, and P. Fua. Multiscale Centerline Detection by Learning a Scale-Space Distance Transform. In *CVPR*, pages 2697–2704, 2014.

[180] A. Sironi, V. Lepetit, and P. Fua. Projection onto the Manifold of Elongated Structures for Accurate Extraction. In *ICCV*, pages 316–324, 2015.

[181] A. Sironi, B. Tekin, R. Rigamonti, V. Lepetit, and P. Fua. Learning Separable Filters. *PAMI*, 37(1):94–106, 2015.

[182] A. Sironi, E. Türetken, V. Lepetit, and P. Fua. Multiscale centerline detection. *PAMI*, 38(7):1327–1341, 2016.

[183] K. Smith, A. Carleton, and V. Lepetit. Fast Ray Features for Learning Irregular Shapes. In *ICCV*, pages 397–404, 2009.

[184] M. Sofka and C. Stewart. Retinal Vessel Centerline Extraction Using Multiscale Matched Filters, Confidence and Edge Measures. *TMI*, 25(12):1531–1546, 2006.

[185] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken. Ridge Based Vessel Segmentation in Color Images of the Retina. *TMI*, 23(4):501–509, 2004.

[186] T. M. Strat and M. A. Fischler. Context-Based Vision: Recognizing Objects Using Both 2D and 3D Imagery. *PAMI*, 13(10):1050–1065, 1991.

[187] S.-y. Takemura, A. Bharioke, Z. Lu, A. Nern, S. Vitaladevuni, P. K. Rivlin, W. T. Katz, D. J. Olbris, S. M. Plaza, P. Winston, et al. A visual motion detection circuit suggested by drosophila connectomics. *Nature*, 500(7461):175–181, 2013.

[188] A. Torralba, K. Murphy, W. Freeman, and M. Rubin. Context-Based Vision System for Place and Object Recognition. In *ICCV*, pages 273–280, 2003.

[189] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection. In *CVPR*, volume 2, pages II–762, 2004.

[190] S. Treitel and J. Shanks. The Design of Multistage Separable Planar Filters. *IEEE Transactions on Geoscience Electronics*, 9(1):10–27, 1971.

[191] T. Trzcinski, V. Lepetit, and P. Fua. Thick Boundaries in Binary Space and Their Influence on Nearest-Neighbor Search. *PRL*, 33(16):2173–2180, 2012.

[192] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *ICML*, page 104, 2004.

[193] Z. Tu and X. Bai. Auto-Context and Its Applications to High-Level Vision Tasks and 3D Brain Image Segmentation. *PAMI*, 32(10):1744–1757, 2010.

[194] E. Turetken, C. Becker, P. Glowacki, F. Benmansour, and P. Fua. Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery Using Multi-Directional Oriented Flux. In *ICCV*, pages 1553–1560, 2013.

[195] E. Turetken, F. Benmansour, B. Andres, P. Glowacki, H. Pfister, and P. Fua. Reconstructing Curvilinear Networks Using Path Classifiers and Integer Programming. *PAMI*, 2016.

[196] E. Turetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua. Reconstructing Loopy Curvilinear Structures Using Integer Programming. In *CVPR*, pages 1822–1829, 2013.

[197] E. Turetken, F. Benmansour, and P. Fua. Semi-Automated Reconstruction of Curvilinear Structures in Noisy 2D Images and 3D Image Stacks. Technical report, EPFL-182839, 2013.

[198] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward Objective Evaluation of Image Segmentation Algorithms. *PAMI*, 29(6):929–944, 2007.

[199] J. D. Van Horn and A. W. Toga. Human neuroimaging as a “big data” science. *Brain imaging and behavior*, 29(6):929–944, 2014.

[200] C. van Rijsbergen. Foundation of Evaluation. *Journal of Documentation*, 30(4):365–373, 1974.

[201] A. Vazquez-reina, M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfister. Segmentation Fusion for Connectomics. In *ICCV*, pages 177–184, 2011.

[202] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. TILDE: A Temporally Invariant Learned DEtector. In *CVPR*, pages 5279–5288, 2015.

[203] A. Wahle, E. Wellnhofer, I. Mugaragu, H. Saner, H. Oswald, and E. Fleck. Assessment of diffuse coronary artery disease by quantitative analysis of coronary morphology based upon 3-d reconstruction from biplane angiograms. *TMI*, 14(2):230–241, 1995.

[204] C. Wang, Y. Li, W. Ito., K. Shimura, and K. Abke. A Machine Learning Approach to Extract Spinal Column Centerline from Three-Dimensional CT Data. In *SPIE*, pages 72594T–72594T, 2009.

[205] C. Wang, A. Stefanidis, A. Croitoru, and P. Agouris. Map registration of image sequences using linear features. *Photogrammetric Engineering & Remote Sensing*, 74(1):25–38, 2008.

[206] Y. Wang, A. Narayanaswamy, and B. Roysam. Novel 4D Open-Curve Active Contour and Curve Completion Approach for Automated Tree Structure Extraction. In *CVPR*, pages 1105–1112, 2011.

[207] J. Wegner, J. Montoya-Zegarra, and K. Schindler. A Higher-Order CRF Model for Road Network Extraction. In *CVPR*, pages 1698–1705, 2013.

[208] R. Westaway, S. Lane, and D. Hicks. Remote survey of large-scale braided, gravel-bed rivers using digital photogrammetry and image analysis. *International Journal of Remote Sensing*, 24(4):795–815, 2003.

[209] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan. Sparse Representation for Computer Vision and Pattern Recognition. *Proc. IEEE*, 98(6):1031–1044, 2010.

[210] C. Xiao, M. Staring, Y. Wang, D. Shamonin, and B. Stoel. Multiscale Bi-Gaussian Filter for Adjacent Curvilinear Structures Detection with Application to Vasculature Images. *TIP*, 22(1):174–188, 2013.

[211] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, pages 1395–1403, 2015.

[212] Y. Xie, F. Xing, X. Kong, H. Su, and L. Yang. Beyond classification: structured regression for robust cell detection using convolutional neural network. In *MICCAI*, pages 358–365. Springer, 2015.

[213] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional Networks. In *CVPR*, pages 2528–2535, 2010.

[214] T. Zhao, J. Xie, F. Amat, N. Clack, P. Ahammad, H. Peng, F. Long, and E. Myers. Automated Reconstruction of Neuronal Morphology Based on Local Geometrical and Global Structural Models. *Neur. Inf.*, 9(2-3):247–261, 2011.

[215] S. Zheng, S. Jayasumana, B. Romera-paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional Random Fields as Recurrent Neural Networks. In *ICCV*, pages 1529–1537, 2015.

[216] Y. Zheng, M. Loziczonek, B. Georgescu, S. Zhou, F. Vega-Higuera, and D. Comaniciu. Machine Learning Based Vesselness Measurement for Coronary Artery Segmentation in Cardiac CT Volumes. *SPIE*, 7962:79621K–1, 2011.

[217] C. Zhou, L. M. Hadjiiski, B. Sahiner, H.-P. Chan, S. Patel, P. Cascade, E. A. Kazerooni, and J. Wei. Computerized detection of pulmonary embolism in 3D computed tomographic (ct) images: vessel tracking and segmentation techniques. In *Medical Imaging 2003*, pages 1613–1620. International Society for Optics and Photonics, 2003.

[218] S. K. Zhou, C. Tietjen, G. Soza, A. Wimmer, C. Lu, Z. Puskas, D. Liu, and D. Wu. A Learning Based Deformable Template Matching Method for Automatic Rib Centerline Extraction and Labeling in CT Images. In *CVPR*, pages 980–987, 2012.

[219] Q. Zhu, D. Zheng, and H. Xiong. 3D Tubular Structure Extraction Using Kernel-Based Superellipsoid Model with Gaussian Process Regression. In *VCIP*, pages 1–6, 2012.

AMOS SIRONI

32, Rue Louis de Savoie
1110 Morges
Switzerland

email: amos.sironi@gmail.com
webpage: people.epfl.ch/amos.sironi/bio

Areas of Expertise: Computer Vision, Deep Learning, Machine Learning, Biomedical Imaging

---

Experience

Ecole Polytechnique Fédérale de Lausanne, Switzerland — 2012 - 2016
Research Assistant at the Computer Vision Laboratory

- Design and Implementation of Computer Vision and Machine Learning Systems
  - Acceleration of image processing operations. Obtained a speed of over 30 times for very large datasets, over previous approaches. The method has been used for accelerating Deep Learning systems
  - Biomedical image segmentation. Improved the performance up to 28% over state-of-the-art linear structure detectors
- Projects Supervision
  Definition of Computer Vision projects, selection and training of candidate students. After completing the project, 75% of the students continued their career in a Computer Vision related field

Ecole Normale Supérieure de Paris, France. — Nov. 2015 - Dec. 2015
Visiting Researcher at the Data Processing and Classification Team

- Implementation and analysis of Convolutional Neural Networks using wavelet based analysis

Education

PhD in Computer Science — 2012 - 2016
Ecole Polytechnique Fédérale de Lausanne, Switzerland
Specialization in Computer Vision

MSc in Mathematics - *110/110 Summa Cum Laude* — 2009 - 2011
Università degli Studi di Padova, Italy

BSc in Mathematics - *110/110 Summa Cum Laude* — 2006 - 2009
Università degli Studi di Milano Bicocca, Italy

Technical Skills

- *Programming Languages*: Matlab, Python, Lua, C/C++
- *Libraries and Frameworks*: Thorch, Theano, Caffe, ITK, NumPy, OpenCV
- *Operating Systems*: Unix/Linux, Mac OS X, Windows

Other Activities

Machine Learning Reading Group

- Open discussion and analysis of latest Machine Learning algorithms

Reviewer for international conferences and journals

Languages

Italian (mother tongue), English (fluent), French (fluent)

Selected Publications

A. Sironi, V. Lepetit and P. Fua. *Projection onto the Manifold of Elongated Structures for Accurate Extraction.* International Conference on Computer Vision (ICCV), Santiago, Chile, 2015.

A. Sironi, E. Tretken, V. Lepetit and P. Fua. *Multiscale Centerline Detection*, in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2015.

A. Sironi, B. Tekin, R. Rigamonti, V. Lepetit and P. Fua. *Learning Separable Filters*, in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2014.

A. Sironi, V. Lepetit and P. Fua. *Multiscale Centerline Detection by Learning a Scale-Space Distance Transform.* Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, Ohio, USA, 2014.

R. Rigamonti, A. Sironi, V. Lepetit and P. Fua. *Learning Separable Filters.* Conference on Computer Vision and Pattern Recognition (CVPR), Portland, Oregon, USA, 2013.