Learning Approaches for Remote Sensing Image Classification

Yuliya Tarabalka

UNIVERSITÉ CÔTE D'AZUR

Inria informatiques mathématiques

École Doctoral STIC
Sciences et Technologies de l'Information
et de la Communication

HABILITATION

A DIRIGER DES RECHERCHES

présentée par

Yuliya Tarabalka

Learning Approaches for Remote Sensing Image Classification

Date de soutenance : 24 Novembre 2017

Composition du Jury :

Mihai	Datcu	Rapporteur
Florence	Tupin	Rapporteur
Lorenzo	Bruzzone	Rapporteur
Paul	Scheunders	Examinateur
Gabriele	Moser	Examinateur
Xavier	Descombes	Examinateur

HDR préparée à l'Inria Sophia-Antipolis Méditerranée

Abstract

The latest generation of aerial- and satellite-based imaging sensors acquires huge volumes of Earth's images with high spatial, spectral and temporal resolution, which open the door to a large range of important applications, such as the monitoring of natural disasters, the planning of urban environments and precision agriculture. In order to fully exploit the potential offered by these sensors, there is a need to develop accurate and time-efficient mathematical models and algorithms for spectral-spatial analysis of the recorded high-resolution data.

The main goal of my research is to develop learning approaches, which would help to automatically interpret, or classify, remote sensing images. This manuscript presents several strategies I have explored for this purpose, varying from the use of strong shape priors to detect objects, regularization of classification probabilities on the image graphs, and up to the use of convolutional neural network models capable to learn deep hierarchical contextual features.

The experimental results on diverse benchmarks of images and image time series show the competitiveness of the developed methods when compared to the state-of-the-art approaches. In particular, we have recently created large-scale classification benchmark of aerial images and have demonstrated that the modern deep learning-based methods succeed in generalizing to the dissimilar urban settlements around the Earth. This opens new exciting perspectives towards designing systems which would be able to automatically update world-scale maps from remote sensing data.

Contents

Chapter 1

Introduction

1.1 Motivations and challenges

The continuous proliferation and improvement of remote (satellite and aerial) data sensors yields a huge volume of Earth's images with high spatial and temporal resolution, as well as with rich spectral information. For instance, the constellation of Pléiades satellites revisits the entire Earth every day, acquiring optical data with a spatial resolution of less than a meter. These data contain a valuable source of information, which opens the door to a large range of important applications, such as the monitoring of natural disasters, the planning of urban environments and precision agriculture. However, petabytes of these massive images are stored into binary files as unstructured raw data. As a result, a large part of them is never used. Thus, it has become crucial to develop methods to automatically analyze these data.

One of the most important problems for the automatic interpretation of remote sens-

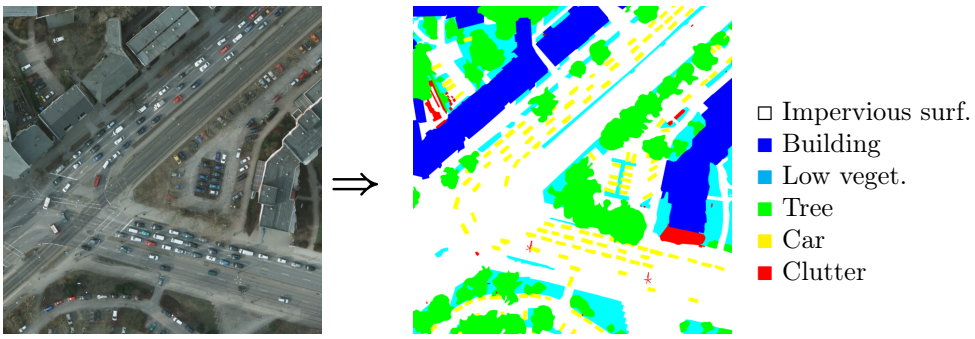


Figure 1.1: Example of pixelwise classification of an aerial image.

ing data is image *classification*, which consists in the assignment of a class, or thematic label (for example, *building*, *tree*, *car*, etc.), to each pixel in the image. Fig. 1.1 shows an example of such pixelwise classification of an aerial image into six classes of interest. In this work, I focus on *supervised* classification, in which the learning approaches are used to train the classifier from labeled reference data. In the manuscript, the terms *labeling* and *classification* are used interchangeably. I also refer to *dense classification* or *semantic classification* to denote the labeling of individual pixels in the image. The term semantic is typically used to emphasize that the classes of interest represent object types (e.g., *car*, *building* [198]) rather than low-level physical materials (e.g., *metal sheets* [13]).

While the first classification approaches have been designed more than forty years ago with the appearance of the first satellites [96], recent technological developments raise new challenges. The first challenge consists in adapting classification methods to the characteristics of new sensors, since the state-of-the-art methods may be not directly applicable or may not give satisfactory performances. For instance, first satellite sensors captured data of a low spatial resolution, and significant research was focused on discovering the proportions (or *abundances*) of physical materials within each single pixel [95]. The recently developed sensors acquire images with a high spatial resolution, where pixels become small when compared to the objects. This implies the need of object-based analysis, i.e., the development of the algorithms which learn to analyze the consistency and relationship between groups of pixels in the image, aiming at identifying the whole objects.

The second challenge is the scalability of the systems. With the increasing amount and openness of the data covering a large geographic extent, there is a growing interest in developing *scalable* methods able to produce classification maps of large areas, in the scale of cities, countries, continents and up to the entire Earth [135]. While there is an urgent need to design computationally efficient algorithms to process large-scale data, it is also a challenge to design methods that would produce accurate maps for dissimilar areas of the Earth, given the large variability of the appearance of landscapes across the planet, i.e. large *intra-class variability* (see Fig. 1.2). While the first classification methods in remote sensing tried to distinguish different classes of low-level materials and substances (e.g., *water*, *soil*, *gravel*), current research studies are in a big part focused on semantic segmentation, where high-level contextual reasoning (e.g., shape and context analysis) must be performed to discriminate more abstract semantic classes.

In this manuscript, I present my research studies on the development of learning methods for remote sensing image classification. In the following, I first summarize the state-of-the-art approaches, and then detail my recent contributions.

Figure 1.2: Examples of aerial images around the Earth, where large intra-class variability can be appreciated for 'building' class.

1.2 Approaches, trends and contributions

The pioneering classification approaches for remote sensing imagery consider each pixel as a pattern and all the data collected by sensors for this pixel are considered as the initial set of features (for instance, spectrum values in the case of optical imagery). Then, the standard pattern recognition scheme [45] is applied to classify each pixel:

1. The classification model and its specifications are selected (e.g., neural network, support vector machines (SVM), etc.).

2. Since the initial set of features may be redundant, feature extraction/selection is often performed, aiming at obtaining the features which discriminate at best classes of interest.

3. The next step, called "training of the classifier," consists in partitioning the entire feature space into K exhaustive, non-overlapping regions, so that every point in the feature space is uniquely associated with one of the K classes. This is done by feeding into the classification model examples of samples for each of the classes (so called *training samples*).

Once the training step is accomplished, every new pixel can be classified according to its feature set. Such classification approach is often referred to as *pixelwise*, where the term pixelwise has the different meaning than in the previous section: here, by pixelwise we mean that to classify any pixel, no information from its neighborhood is taken into account. Different models have been applied for pixelwise classification in the

remote sensing community; among them, probably the most widely used ones are neural networks [7, 8, 137], SVM [193, 21] and random forests [76, 77].

In a real image, neighboring pixels are related or correlated, both because imaging sensors acquire significant amount of energy from adjacent pixels [208] and because objects occurred in the image scene are in most cases large when compared to the size of a pixel. For instance, if a given pixel in the agricultural image scene represents a corn, its adjacent pixels belong with the high probability to the corn field. This spatial contextual information significantly helps to interpret image scene. Therefore, most recent classification approaches are *spectral-spatial*, i.e. they classify each image pixel based on: 1) its own spectral values (the spectral information) and 2) information extracted from its neighborhood (the spatial information). The use of spectral-spatial classifiers is especially rewarding when processing images with high spatial resolution; that is why these techniques are especially relevant to analyze data captured by modern and near-future sensors.

I refer the reader to my PhD thesis [178] for a detailed overview of pixelwise and spectral-spatial classification approaches. There are different ways to group these techniques in the state-of-the-art. In the following, I will present three important families/strategies of spectral-spatial classification, which are successfully used until nowadays and can also be used in combination with each other, and will position my contributions with respect to these strategies:

1. **Graph-based methods.** In these approaches, an image is represented as a graph, where the nodes typically represent image pixels or regions, while edges either connect adjacent regions [180, 10], or keep track of an hierarchy in a multi-scale image representation [191, 127]. By applying different graph construction and pruning strategies, these techniques typically output a disconnected graph, where each connected subgraph corresponds to the classified object in the resulting classification map. A popular graph-cut technique applies a min-cut algorithm on the image graph to regularize outputs of the pixelwise classifier, by minimizing the Markov Random Field-based energy function associated with the image and the initial classification result [181]. Another approach consists in building first a hierarchy of image regions at different levels of details, based on standard homogeneity/dissimilarity criteria, and then selecting from this hierarchy the regions at different scales that correspond to the objects in the final classification map [191, 129]. Such hierarchical data structure can be very flexible, for instance, it can be easily augmented to include textural or shape information for more accurate image anal-

ysis. I will present our graph-based methods in Chapters 3 and 4. Chapter 3 will detail a method for shape-aware multi-object multi-class classification, based on a hierarchical data structure known as binary partition trees. Chapter 4 will describe how a graph-cut-based method can be segmentation for the analysis of image time series.

2. **Feature engineering.** The spatial information can be included into a classifier by analyzing the similarity/correlations between the data in the neighborhood of an image pixel. The initial approaches would typically use a well-defined algorithm, which can be often described by a (set of) pre-defined convolutional mask(s), for this purpose. For instance, Tsai et al. [187] and Huang and Zhang [82] have investigated the use of texture measures derived from the gray level co-occurrence matrix, such as angular second moment, entropy and homogeneity, for spectral-spatial classification. A popular approach in remote sensing explores the principles of mathematical morphology, which by definition aims at analyzing spatial relations between sets of pixels, i.e., extracting information about the size, shape and orientation of structures. For this purpose, morphological profiles [9], extended morphological profiles [48], attribute profiles [142] and extinction profiles [61] have been successfully applied. In all cases, the core idea is to progressively simplify image based on a pre-defined rule/attribute, which allows to analyze a particular characteristic (for instant, area or standard deviation of regions) within this image. The extracted morphological features typically lay in a very-high-dimensional space, and additional feature extraction must be applied to reduce the dimensionality of the feature space. Another group of techniques introduces a strong shape priors into classification: for example, in remote sensing images buildings can be modelled as rectangles [145, 199] and roads can be modelled by line segments [106]. Chapter 2 will describe our studies using such approach, where we model curvilinear structures by a set of line segments with variable length and orientations.

3. **Deep learned features.** Even though feature engineering gives satisfactory results for many applications, when the complexity of the data and the number of classes increases, it is not straightforward to design hand-crafted features that would exhibit high discriminative power for all classes. The recent trend in the state-of-the-art consists in designing classification systems that would be able to automatically learn hundreds of expressive high-level contextual features. In this context, the use of deep learning, in particular *convolutional neural networks*, is intensively investigated by both image analysis and remote sensing communities.

While some works use deep learning-based features as the input to a conventional classifier, such as SVM or logistic regression classifier [213], most recent studies design an end-to-end system which jointly learns to extract relevant contextual features and conduct the classification [140, 131, 132]. I will detail our recent contributions on the development of deep learning architectures for remote sensing image classification in Chapter 5. In particular, I will present how we face the challenge to obtain fine-grained high-resolution classification maps by designing appropriate deep learning architectures.

Even though the main focus of my research is remote sensing data classification, I will also show experimental results on other datasets, such as medical or computer vision benchmarks, to illustrate the genericity of the proposed methods.

1.3 Selected publications

This section lists my publications after the PhD thesis defense, i.e. after 2010.

Peer-reviewed international journals

1. E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "High-resolution aerial image labeling with convolutional neural networks," *IEEE Trans. Geosc. Remote Sens.*, 2017.

2. E. Maggiori, G. Charpiat, Y. Tarabalka, and P. Alliez, "Recurrent neural networks to correct satellite image classification maps," *IEEE Trans. Geosc. Remote Sens.*, 2017.

3. E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Convolutional neural networks for large-scale remote sensing image classification," *IEEE Trans. Geosc. Remote Sens.*, Volume 55(2), pages 645–657, February 2017.

4. H. Aghighi, J. Trinder, S. Lim and Y. Tarabalka, "Fully spatially adaptive smoothing parameter estimation for Markov random field super-resolution mapping of remotely sensed images," *International Journal of Remote Sensing*, vol. 36, no. 11, pp. 2851-2879, 2015.

5. B. B. Damodaran, R. R. Nidamanuri and Y. Tarabalka, "Dynamic ensemble selection approach for hyperspectral image classification with joint spectral and spatial

information," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2405-2417, 2015.

6. Y. Tarabalka, G. Charpiat, L. Brucker and B. H. Menze, "Spatio-temporal video segmentation with shape growth or shrinkage constraint," *IEEE Transactions on Image Processing*, vol. 23, no. 9, pp. 3829-3840, 2014.

7. H. Aghighi, J. Trinder, Y. Tarabalka, and S. Lim, "Dynamic block-based parameter estimation for MRF classification of high-resolution images," *IEEE GRSL*, vol. 11, no. 10, pp. 1687-1691, 2014.

8. M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Advances in spectral-spatial classification of hyperspectral images," *Proceedings of the IEEE*, vol. 101, no. 3, pp. 652-675, March 2013.

9. J. C. Tilton, Y. Tarabalka, P. M. Montesano, and E. Gofman, "Best merge region growing with integrated region object classification," *IEEE Trans. Geosc. Remote Sens.*, vol. 50, no. 11, pp. 4454-4467, Nov. 2012.

10. K. Bernard, Y. Tarabalka, J. Angulo, J. Chanussot, and J. A. Benediktsson, "Spectral-spatial classification of hyperspectral data based on a stochastic minimum spanning forest approach," *IEEE Trans. on Image Processing*, vol. 21, no. 4, pp. 2008-2021, April 2012.

11. Y. Tarabalka, J. C. Tilton, J. A. Benediktsson, and J. Chanussot, "A marker-based approach for the automated selection of a single segmentation from a hierarchical set of image segmentations," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 1, pp. 262-272, Feb. 2012.

Book chapters

1. E. Maggiori, A. Plaza and Y. Tarabalka, "Models for hyperspectral image analysis: from unmixing to object-based classification," *Mathematical Models for Remote Sensing Image Processing* (edit. by G. Moser and J. Zerubia), Springer, 2017.

2. J. Tilton, S. Aksoy and Y. Tarabalka, "Image segmentation algorithms for land categorization," *Remote Sensing Handbook* (edited by P. S. Thenkabail), Taylor and Francis, 2015.

International conference papers

1. E. Maggiori, Y. Tarabalka, G. Charpiat and P. Alliez. "Polygonization of Remote Sensing Classification Maps by Mesh Approximation", *IEEE ICIP'17*. Beijing, China, 2017.

2. E. Maggiori, Y. Tarabalka, G. Charpiat and P. Alliez, "Can semantic labeling methods generalize to any city? The Inria aerial image labeling benchmark," *IGARSS'17*, Fort Worth, Texas, USA, 2017.

3. E. Maggiori, Y. Tarabalka, G. Charpiat and P. Alliez, "High-resolution image classification with convolutional networks," *IGARSS'17*, Fort Worth, Texas, USA, 2017.

4. E. Maggiori, Y. Tarabalka, G. Charpiat and P. Alliez, "Fully convolutional neural networks for remote sensing image classification," *IGARSS'16*, Beijing, China, 2016.

5. E. Maggiori, Y. Tarabalka and G. Charpiat, "Optimizing partition trees for multi-object segmentation with shape prior," *BMVC'2015*, Swansea, UK, 2015.

6. S-G. Jeong, Y. Tarabalka and J. Zerubia, "Stochastic model for curvilinear structure reconstruction using morphological profiles," *IEEE ICIP'15*, Quebec City, Canada, 2015.

7. E. Maggiori, Y. Tarabalka and G. Charpiat, "Improved partition trees for multi-class segmentation of remote sensing images," *IGARSS'15*, Milan, Italy, 2015.

8. S-G. Jeong, Y. Tarabalka and J. Zerubia, "Marked point process model for curvilinear structures extraction," *EMMCVPR'15*, Hong Kong, 2015.

9. S-G. Jeong, Y. Tarabalka and J. Zerubia, "Marked point process model for facial wrinkle detection," *IEEE ICIP'14*, Paris, France, 2014.

10. Y. Tarabalka and A. Rana, "Graph-cut-based model for spectral-spatial classification of hyperspectral images," *IGARSS'14*, Quebec City, Canada, 2014.

11. H. Aghighi, J. Trinder, K. Wang, Y. Tarabalka and S. Lim, "Smoothing parameter estimation for Markov random field classification of non-Gaussian distribution image," *ISPRS TC VII Symposium*, Istanbul, Turkey, 2014.

12. Y. Tarabalka, G. Charpiat, L. Brucker and B. H. Menze, "Enforcing monotonous shape growth or shrinkage in video segmentation," *BMVC'2013*, Bristol, UK, 2013.

13. C. Price, Y. Tarabalka and L. Brucker, "Graph-based method for multitemporal segmentation of sea ice floes from satellite data," *LARS'13*, Santiago, Chile, 2013.

14. Y. Tarabalka and G. Charpiat, "A graph-cut-based method for spatio-temporal segmentation of fire from satellite observations," *IGARSS'13*, Melbourne, Australia, 2013.

15. Y. Tarabalka, "Overview of hierarchical models for hyperspectral image classification," *UkrOBRAZ* - Signal/Image Processing and Pattern Recognition Conference, Kyiv, Ukraine, 2012.

16. Y. Tarabalka, L. Brucker, A. Ivanoff and J. C. Tilton, "Shape-constrained segmentation approach for Arctic multiyear sea ice floe analysis," *IGARSS'12*, Munich, Germany, 2012.

17. Y. Tarabalka and J. Tilton, "Improved hierarchical optimization-based classification of hyperspectral images using shape analysis," *IGARSS'12*, Munich, Germany, 2012.

18. Y. Tarabalka and J. C. Tilton, "Spectral-spatial classification of hyperspectral images using hierarchical optimization," *WHISPERS'11*, Lisbon, Portugal, 2011.

19. Y. Tarabalka, J. C. Tilton, J. A. Benediktsson and J. Chanussot, "Marker-based hierarchical segmentation and classification approach for hyperspectral imagery", *ICASSP'11*, Czech Republic, 2011.

20. K. Bernard, Y. Tarabalka, J. Angulo, J. Chanussot, and J. A. Benediktsson, "A stochastic minimum spanning forest approach for spectral-spatial classification of hyperspectral images," *ICIP'11*, Brussels, Belgium, 2011.

21. Y. Tarabalka and J. C. Tilton, "Best merge region growing with integrated probabilistic classification for hyperspectral imagery," *IGARSS'11*, Vancouver, Canada, 2011.

Chapter 2

Reconstruction of Curvilinear Structures

One of the pioneering methods aiming at incorporating information from the spatial neighbourhood to properly classify images used a template-based concept. This implies introducing a strong shape prior into classification. For instance, in moderate-spatial-resolution aerial images roads can be modelled by line segments [106], and buildings can be seen as rectangles [145].

This chapter covers my research on reconstruction of curvilinear structures based on stochastic modeling and ranking learning system. We assumed that the entire line network can be decomposed into a set of line segments with variable lengths and orientations. This assumption enables us to reconstruct arbitrary shapes of curvilinear structure for different types of datasets.

In the following, I describe two models we have developed. The first model works in an unsupervised way, and is based on Marked point process (MPP) and Monte Carlo sampling. We reconstruct the latent curvilinear structure by sampling disjoint line segments which are associated with with the given image data. Therefore, we aim to maximize a posterior probability density with respect to a set of line segments for given image data. We constrain local interaction of the line segments to obtain smoothly connected line configuration. The optimization technique consists of two steps to reduce the significance of the parameter selection in our stochastic model. We simulate several Markov chains with different parameter settings to collect line hypotheses on the same configuration space. Then, we maximize the consensus among line hypotheses to reduce the sampling space and to improve the reliability of the curvilinear structure reconstruction.

The second model is based on supervised learning and graph theory. To extract local

curvilinear features, we compute oriented gradient information using steerable filters. We then employ structured support vector machine (SVM) for ordinal regression of the input image patches, where the ordering is determined by shape similarity to latent curvilinear structure. Finally, we progressively reconstruct the curvilinear structure by looking for geodesic paths connecting remote vertices in the graph built on the structured output rankings.

2.1 Related work

Curvilinear structure extraction

In early vision, researchers design convolution filters to separate curvilinear structure from the background texture ([54, 150, 86]). The main idea behind such filter design is to create simple line shape templates in order to extract features showing high gradient magnitudes with a locally consistent orientation. However, these image filtering responses are unable to discern linear structures from undesirable high-frequency components, *e.g.*, noise and edges. Enhancement filtering (EF) algorithm proposed by [52] analyzes the eigenvalues of the Hessian matrix of the image to evaluate local tubularity score. Optimally Oriented Flux (OOF) measures the amount of gradient flow at the boundaries to find the continuous linear structure ([110]). Morphological operator also highlights the curvilinear structure by collecting pixels according to the recursive structural similarity ([176]).

On the other hand, the graphical models exploited image-based evidence with geometric shape constraints to improve the detection performances. For instance, tree structure can be used to reinterpret complex line networks. The authors of [66] looked for a path between singular points corresponding to intersections of the latent curvilinear structure. Türetken *et al.* [189] formulated a large linear programming problem to constrain the diverse cases of the local interaction on the line networks.

Furthermore, curves can be approximated by multiple straight line segments. Stochastic models specify the distribution of line objects with pairwise interaction terms ([107]). Reversible Jump Markov chain Monte Carlo sampler proposed by [70] has been involved to optimize the probability density. However, due to the sensitivity of the parameter setup, the stochastic models are less practical for a large amount of varied datasets.

In recent years, machine learning algorithms have been favoured for designing optimal filter banks to extract curvilinear features ([152, 5]). While a threshold value should be set to reconstruct the centerline of the curvilinear structure from the filtering scores, it

is difficult to find the value which would be suitable for different types of applications. To address this issue, [169] proposed a regression model of distance transform to predict the scale of the curvilinear structure and localize centerlines.

Learning structured information

We want to extract the structured information of the line networks that appear within homogeneous background textures. Since the information embedded in a single pixel is limited to infer the latent structure, Markov Random Fields ([59, 12]) or Conditional Random Fields (CRF, [108, 103]) based models have been developed to enforce the label consistency on the pre-organized output space. *e.g.*, neighborhood pixels are assumed to have the same label with the probability when minimizing the cost function. However, the topology of the curvilinear structure composed of line objects is too intricate to be applicable with such design approaches. It is also impossible to specify the topology of the complex line networks with a few parameters.

Recently, machine learning algorithms have been involved to detect curvilinear structures latent in various types of images. Becker *et al.* [5] applied a boosting algorithm to obtain an optimal set of convolution filter banks. Sironi *et al.* proposed a regression model to detect centerlines by learning the scale (width) of the tubular structures with the non-maximum suppression technique [169].

Structured learning systems have been employed in image segmentation models based on random fields [11, 126, 97]. More specifically, Structured Support Vector Machine (SSVM) [188] has been used to predict model parameters for inference of the structured information between input image space and output label space. Exploring all possible combinations of the labels in the output space is computationally intractable. Thus, the random field models define the pairwise relationship in a neighborhood system to enforce the labeling consistency. While such prior models based on random fields are successful to describe convex shaped objects, they should be adapted to detect thin and elongated shaped curvilinear objects.

2.2 Stochastic modeling of curvilinear structure reconstruction

In the following, I describe the proposed stochastic model based on MPP framework for curvilinear feature extraction in a fully automatic way, where the performance is not biased by the hyperparameter selection.

2.2.1 Marked point process modeling

We briefly review the definition of MPP [121, 175] to provide a mathematical description of the proposed model.

Definition 1 (**Spatial point process**). *A realization of point process consists of an unordered set of points in a compact set* $\mathcal{F} \subset \mathbb{R}^d$. *A point process on* $\mathcal{F}$ *maps from a measurable probability space* $(\mathcal{F}, \mathcal{B}, \mu)$ *onto the configuration space* $\Omega = \cup_{n=0}^{\infty} \Omega_n$, *where* $\mathcal{B}$ *denotes* σ*-algebra of subset of* $\mathcal{F}$, *and* μ *is the Lebesgue measure. In other words, for all bounded Borel sets* $B \subseteq \mathcal{B}$, *the number of points falling in* B *is a finite random variable.*

Definition 2 (**Marked point process**). *In the MPP framework, each point is associated with additional information which describe a shape of the object. Specifically, we reconstruct curvilinear structures as smoothly connected line segments. Let* $s_i = (\mathbf{x}_i, \mathbf{m}_i)$ *be a line segment specifying its center point* $\mathbf{x}_i = (x_i, y_i)$ *in the image sites* $\mathcal{F}$ *with a label of the length and the orientation* $\mathbf{m}_i = (\ell_i, \theta_i)$, *where the label is sampled from the mark space* M *with a probability measure* μ_M. *We now define a marked point process on* $\mathcal{F} \times M$ *as a finite random configuration* $\boldsymbol{s} = \{s_1, \ldots, s_n\} \in \Psi$.

The probability distribution of the MPP is defined based on an image I and spatial interactions between line segments. Given an image, we look for an optimal configuration $\hat{\mathbf{s}}$ which maximizes the unnormalized probability density $f(\mathbf{s})$ as follows:

$$\hat{\mathbf{s}} = \underset{\mathbf{s} \in \Psi}{\operatorname{argmax}} f(\mathbf{s}) = \underset{\mathbf{s} \in \Psi}{\operatorname{argmin}} \sum_{i=1}^{\#(\mathbf{s})} U_d(s_i) + \sum_{i \sim j} U_p(s_i, s_j), \tag{2.1}$$

where $\#(\mathbf{s})$ is the cardinality of the configuration, and $i \sim j$ represents the symmetry relationship between interacting line segments s_i and s_j. U_d and U_p denote the *data energy* (which we also call *data likelihood*) and the *prior energy*, respectively. In general, Monte Carlo samplers [57, 71, 72, 195] are employed in MPP models to maximize the proposed density function $f(\mathbf{s})$. Each state of a discrete Markov chain $(X_t)_{t \in \mathbb{N}}$ corresponds to a random configuration on the Ψ. The chain is locally perturbed by transition kernels, and is evolved to converge to the stationary distribution which is identical to the proposed probability density.

2.2.1.1 Data energy

We define the data energy of the line segment s_i as a weighted sum of the rotated gradient magnitudes U_d^m and the intensity variance U_d^v along the line:

$$U_d(s_i) = \omega_d^m U_d^m(s_i) + \omega_d^v U_d^v(s_i), \tag{2.2}$$

where ω_d^m and ω_d^v are weighting coefficients corresponding to U_d^m and U_d^v, respectively.

We obtain the rotated gradient information by convolving the input image with steerable filters [55, 87]. Steerable filters are generated from a linear combination of basis filters. In this work, we use second-order derivatives of an isotropic Gaussian function as the basis filters. Let $g_{\theta_i}\left(\mathbf{x};\sigma^2\right)$ be a steerable filter associated with an orientation θ_i and $\nabla I_{\theta_i} = g_{\theta_i} * I$ be its filtering response, which adaptively accentuates gradient magnitudes corresponding to the angle θ_i. Then, the gradient magnitude energy U_d^m is defined as

$$U_d^m(s_i) = \int_0^1 \left|\nabla I_{\theta_i}\left(\mathbf{p}_i(t)\right)\right| dt, \tag{2.3}$$

where $\mathbf{p}_i(t)$ represents points on the line segment s_i. Note that $\mathbf{p}_i(t) = (1-t)\mathbf{u}_i + t\mathbf{v}_i$ is a function of the endpoints $\mathbf{u}_i$ and $\mathbf{v}_i$ with parameter $t \in [0, 1[$.

When the input image is heavily corrupted by noise or composed of uneven textures, observing gradient distribution often fails to detect linear structures. To ease this problem, we also measure the intensity variance along the line segment. This is because intensities are likely to be homogeneous, if pixels are laid on the same line. We can write:

$$U_d^v(s_i) = \frac{1}{\ell_i}\int_0^1 \left(I\left(\mathbf{p}_i(t)\right) - \mathbb{E}[I(s_i)]\right)^2 dt, \tag{2.4}$$

where $\mathbb{E}[I(s_i)]$ denotes the intensity mean of the line segment s_i, and ℓ_i is the line length.

2.2.1.2 Prior energy

In this section, we propose the prior energy to define spatial interactions on a local configuration. We want to obtain smoothly connected lines with a small curvature as a final solution. We compute the overlapping area $\Upsilon(s_i, s_j)$ to reject congestion of lines and the coupling energy states $\mathbf{c}_{ij}$ to evaluate attraction between line segments (see Fig. 2.1). The prior energy $U_p(s_i, s_j)$ is defined as

$$U_p(s_i, s_j) = \Upsilon(s_i, s_j) + \mathbf{w}_p^\top \mathbf{c}_{ij}, \quad \forall\ i \sim j, \tag{2.5}$$

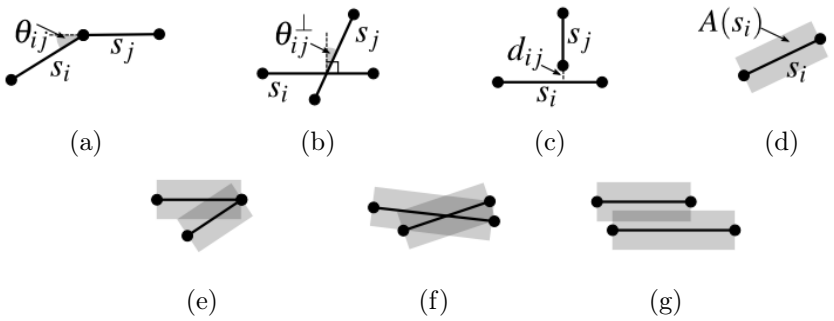


Figure 2.1: Examples of the line configurations with different prior energies: (a)–(c) show preferable line configurations composed of aligned lines (a), almost perpendicular lines (b), and adjacent lines (c). (d)–(g) depict unfavourable line configurations which are penalized because of a singular segment (d), acute corner (e), overlap (f), and parallel (g), respectively.

where $\mathbf{w}_p$ denotes a vector of weighting factors which control relative importance of each element in $\mathbf{c}_{ij}$. We assume that a line segment only correlates with the other ones within a certain distance. Thus, a neighborhood system consists of pairs of line segments, such that their center distance is smaller than half the sum of their lengths. In other words,

$$i \sim j = \left\{ (s_i, s_j) \in \Psi^2 : 0 < \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \frac{\ell_i + \ell_j}{2} + \epsilon \right\}, \tag{2.6}$$

where ϵ denotes the marginal distance to be connected with each other.

In order to evaluate an overlapping area between line segments, we dilate the line segments with a three pixel-radius disk, and then count up the number of pixels falling in the same image site. Suppose that we have a set of points $A(s_i)$ which is a dilated version of the line segment s_i, and $n(A(s_i))$ denotes the number of pixels in $A(s_i)$. As shown in Fig. 2.1 (e)–(g), we penalize a configuration $\{s_i,\ s_j\}$, when a portion of the overlapping area is greater than 10% of $\min\{n(A(s_i)), n(A(s_j))\}$. However, almost perpendicular line segments are excluded from this penalty. The criteria for rejection are then given as

$$\Upsilon(s_i, s_j) = \begin{cases} 0 & \text{if } \theta_{ij}^{\perp} < \tau, \\ 0 & \text{if } \frac{n(A(s_i) \cap A(s_j))}{\min\{n(A(s_i)), n(A(s_j))\}} < 0.1, \\ \infty & \text{otherwise,} \end{cases} \tag{2.7}$$

where $\theta_{ij}^{\perp} = \frac{\pi}{2} - \theta_{ij}$ represents an angle difference between s_i and the perpendicular line of s_j, τ is the maximum angle difference for segments to be aligned.

The coupling energy states $\mathbf{c}_{ij}$ of the lines are composed of the singularity, connec-

tivity, curvature, and perpendicularity:

$$\mathbf{c}_{ij} = [1, \varphi(d_{ij}, \epsilon), \varphi(\theta_{ij}, \tau), \varphi(\theta_{ij}^{\perp}, \tau)]^{\top}, \quad \varphi(u, v) = \min\{0, (u/v)^2 - 1\}, \tag{2.8}$$

where d_{ij} denotes the minimum distance from endpoints of s_i to a point on the line s_j, and θ_{ij} is the angle difference between line segments. The function $\varphi(u, v)$ tests a firmness of the coupling state u by comparing with the given tolerance value v.

The weighting factors $\mathbf{w}_p = [\omega_p^s, \omega_p^c, \omega_p^a, \omega_p^r]^{\top}$ can be derived from their role in the prior energy. Specifically, ω_p^s penalizes birth of a single line segment in the final configuration; hence its value is affected by the average gradient magnitude and the noise level of the input. ω_p^c encourages adjacent segments within ϵ to become connected. ω_p^a promotes segments being aligned with a small curvature in the final configuration. ω_p^r supports perpendicularly approaching line segments. Although the selection of $\mathbf{w}_p$ values is critical for the performances of the MPP model, it is hard to estimate the coefficients because of hidden dependencies among them.

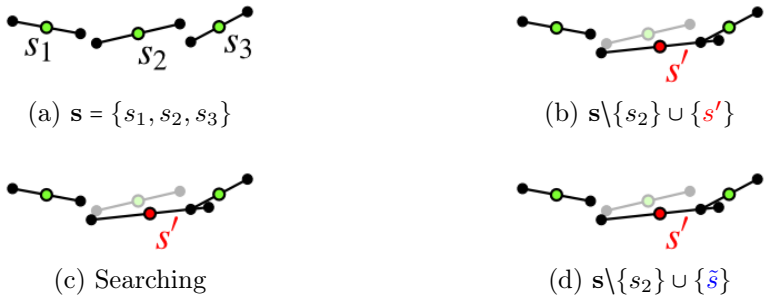
2.2.1.3 Monte Carlo sampler with delayed rejection

We employ the *Reversible jump Markov chain Monte Carlo* (RJMCMC) sampler [71] to obtain an optimal line configuration which maximizes the probability density function. The RJMCMC sampler is an iterative method that locally perturbs a current configuration $\mathbf{s}$ with a transition kernel. The transition kernel consists of multiple sub-transition kernels, namely, *birth-and-death* (BD) and *linear transform* (LT). A new configuration $\mathbf{s}'$ is proposed according to the transition kernel, given by

$$\xi\left(\mathbf{s}, \mathbf{s}'\right) = \sum_{\mathrm{m}} p_{\mathrm{m}} \xi_{\mathrm{m}}\left(\mathbf{s}, \mathbf{s}'\right), \tag{2.9}$$

where p_{m} denotes a probability to choose m-th type of sub-transition kernel $\xi_{\mathrm{m}}\left(\mathbf{s}, \mathbf{s}'\right)$. For each sub-transition kernel, the detailed balance condition [71] is required to ensure the reversibility of the Markov chain. Acceptance ratio $\alpha_{\mathrm{m}}(\mathbf{s}, \mathbf{s}')$ is compared with a stochastic value $\texttt{rand}[0, 1]$ to take a new configuration into account. The RJMCMC sampler is coupled with the *simulated annealing* (SA) algorithm [98] to secure the convergence of the Markov chain via relaxation parameter T (temperature); the temperature gradually decreases as the iteration goes on. To compute an acceptance ratio of the transition kernel, we use a density $f(\mathbf{s})^{1/T}$ instead of $f(\mathbf{s})$. The acceptance ratio is

$$\alpha_{\mathrm{m}}(\mathbf{s}, \mathbf{s}') = \min\left(1, \frac{\xi_{\mathrm{m}}\left(\mathbf{s}', \mathbf{s}\right)}{\xi_{\mathrm{m}}\left(\mathbf{s}, \mathbf{s}'\right)} \frac{f(\mathbf{s}')^{1/T}}{f(\mathbf{s})^{1/T}}\right). \tag{2.10}$$



(a) $\mathbf{s} = \{s_1, s_2, s_3\}$ (b) $\mathbf{s}\backslash\{s_2\} \cup \{s'\}$

(c) Searching (d) $\mathbf{s}\backslash\{s_2\} \cup \{\tilde{s}\}$

Figure 2.2: Given configuration (a), if a line segment s' proposed by LT kernel is rejected (b), the delayed rejection kernel searches for the nearest extremes in the rest of line segments (c). An alternative line segment $\tilde{s}$, which enforces connectivity, will be proposed by interpolation of the retrieved points (d).

The BD kernel changes the dimensionality of the current configuration $\mathbf{s}$ by adding a new line segment or removing an existing line segment. When the birth kernel proposes a new configuration $\mathbf{s}' = \mathbf{s} \cup \{s\}$, the length and the orientation of the new line segment are uniformly sampled from the mark space $M = [\ell_{\min}, \ell_{\max}] \times [\theta_{\min}, \theta_{\max}]$, where $\ell_{\min}$ and $\ell_{\max}$ are the minimum and maximum length of the line segment, respectively. $\theta_{\min}$ and $\theta_{\max}$ denote the minimum and maximum orientation of the line segment, respectively. Note that we refuse a birth of the line lying on *singular points*, which have zero gradient magnitudes. On the other hand, the death kernel removes a line segment which is randomly picked from the current configuration. Thus, a new configuration $\mathbf{s}' = \mathbf{s}\backslash\{s\}$ is proposed by the death kernel. We compute the acceptance ratio of the birth kernel α_{B} and the death kernel α_{D} in the same way as proposed in [106], given by

$$\alpha_{\mathrm{B}}(\mathbf{s}, \mathbf{s}') = \min\left(1, \frac{p_{\mathrm{D}}}{p_{\mathrm{B}}} \frac{\mu(\mathcal{F})}{\#(\mathbf{s}) + 1} \frac{f(\mathbf{s}')^{1/T}}{f(\mathbf{s})^{1/T}}\right), \tag{2.11}$$

$$\alpha_{\mathrm{D}}(\mathbf{s}, \mathbf{s}') = \min\left(1, \frac{p_{\mathrm{B}}}{p_{\mathrm{D}}} \frac{\#(\mathbf{s})}{\mu(\mathcal{F})} \frac{f(\mathbf{s}')^{1/T}}{f(\mathbf{s})^{1/T}}\right). \tag{2.12}$$

The LT kernel chooses a line segment s randomly, and then modifies its model parameters: $s = (\mathbf{x}, (\ell, \theta)) \rightarrow s' = (\mathbf{x} \pm d\mathbf{x}, (\ell \pm d\ell, \theta \pm d\theta))$, where $d\mathbf{x}$, $d\ell$, and $d\theta$ denote changes of center position, length, and orientation, respectively. The LT kernel draws a new configuration $\mathbf{s}' = \mathbf{s}\backslash\{s\} \cup \{s'\}$. The acceptance ratio of the LT kernel is defined by

$$\alpha_{\mathrm{LT}}(\mathbf{s}, \mathbf{s}') = \min\left(1, \frac{f(\mathbf{s}')^{1/T}}{f(\mathbf{s})^{1/T}}\right). \tag{2.13}$$

Algorithm 1 RJMCMC sampler with delayed rejection

Initialize: $X_0 \leftarrow \mathbf{s}_0$ (or $X_0 \leftarrow \varnothing$), $t \leftarrow 0$, $T \leftarrow T_0$
while $T > T_{\min}$ **do**
  $\mathbf{s} \leftarrow X_t$
  Choose a transition kernel ξ_{m} according to probability p_m
  Propose a new configuration $\mathbf{s}'$ with $\xi_{\mathrm{m}}(\mathbf{s}, \mathbf{s}')$
  if $\alpha_{\mathrm{m}}(\mathbf{s}, \mathbf{s}') > \mathtt{rand}[0,1]$ **then**
    $X_{t+1} \leftarrow \mathbf{s}'$
  else
    Propose an alternative segment $\tilde{\mathbf{s}}$ based on $\xi^2_{\mathrm{LT}}(\mathbf{s}, \mathbf{s}', \tilde{\mathbf{s}})$
    if $\alpha^2_{\mathrm{LT}}(\mathbf{s}, \mathbf{s}', \tilde{\mathbf{s}}) > \mathtt{rand}[0,1]$ **then**
      $X_{t+1} \leftarrow \tilde{\mathbf{s}}$
    else
      $X_{t+1} \leftarrow \mathbf{s}$
    end if
  end if
  $t \leftarrow t + 1$
  Decrease the temperature: $T \leftarrow T_t$
end while

The LT kernel can be extended by the delayed rejection scheme [72]. The main idea of the delayed rejection scheme is to give a second chance to a rejected sample point. The acceptance ratio of delayed rejection is defined by

$$\begin{aligned}\alpha^2_{\mathrm{LT}}(\mathbf{s}, \mathbf{s}', \tilde{\mathbf{s}}) &= \min\left(1, \frac{\xi_{\mathrm{LT}}(\tilde{\mathbf{s}}, \mathbf{s}')}{\xi_{\mathrm{LT}}(\mathbf{s}, \mathbf{s}')} \frac{\xi^2_{\mathrm{LT}}(\mathbf{s}', \tilde{\mathbf{s}}, \mathbf{s})}{\xi^2_{\mathrm{LT}}(\mathbf{s}, \mathbf{s}', \tilde{\mathbf{s}})} \frac{[1 - \alpha_{\mathrm{LT}}(\tilde{\mathbf{s}}, \mathbf{s}')]}{[1 - \alpha_{\mathrm{LT}}(\mathbf{s}, \mathbf{s}')]} \frac{f(\tilde{\mathbf{s}})^{1/T}}{f(\mathbf{s})^{1/T}}\right), \\ &\simeq \min\left(1, \frac{f(\tilde{\mathbf{s}})^{1/T} - f(\mathbf{s}')^{1/T}}{f(\mathbf{s})^{1/T} - f(\mathbf{s}')^{1/T}}\right). \end{aligned} \tag{2.14}$$

where $\mathbf{s}' = \mathbf{s} \setminus \{s\} \cup \{s'\}$, $\tilde{\mathbf{s}} = \mathbf{s} \setminus \{s\} \cup \{\tilde{s}\}$, and $\xi^2_{\mathrm{LT}}(\mathbf{s}, \mathbf{s}', \tilde{\mathbf{s}})$ is the transition kernel for the delayed rejection. In order to reduce the burn-in time, we add heuristics to design the delayed rejection kernel. When we propose an alternative line segment $\tilde{s}$, we look for the closest endpoints from both ends of s', which is rejected from the first trial. The line segment $\tilde{s}$ is generated by interpolation of the retrieved points; we force the connectivity of the neighboring segments, so that a probability of being accepted increases in terms of prior energy. Fig. 2.2 summarizes the process of the delayed rejection kernel, and Algorithm 1 provides the pseudo-code of the RJMCMC sampler with delayed rejection.

2.2.2 Curvilinear structure extraction via integration of line hypotheses

While the MPP allows to design complex prior knowledge of the object distribution, its performance is very sensitive to the selection of modeling parameters and hyperpa-

rameters. For clarity, we note that the modeling parameters are related to the physical characteristics of the line segments (*e.g.,* range of length and orientation). The hyperparameters denote the weighting coefficients of energy terms (*i.e.*, w_d^m, w_d^v, and $\mathbf{w}_p$). The modeling parameters can be chosen empirically since the values are related to the image resolution (see Sec. 2.2.4); however, it is hard to estimate the hyperparameters via trial-and-error for different types of dataset. Our goal is to maximize the probability density without estimating hyperparameters.

2.2.2.1 Generation of K line hypotheses

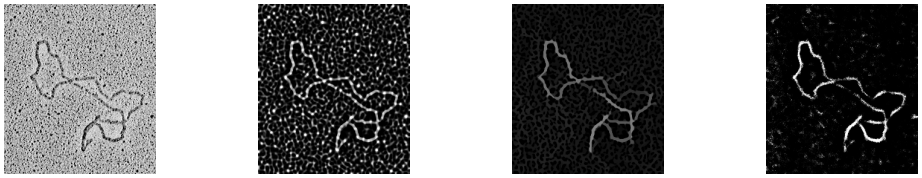
Let $\mathbf{w} = [\omega_d^m, \omega_d^v, \omega_p^s, \omega_p^c, \omega_p^a, \omega_p^r]^\top$ be a hyperparameter vector which consists of the weighting coefficients of the proposed probability density. Suppose that we have K different hyperparameter vectors, $\mathbf{w}^1, \ldots, \mathbf{w}^K$. For each hyperparameter vector, we substitute k-th hyperparameter vector $\mathbf{w}^k$ into the proposed probability density $f(\mathbf{s}; \mathbf{w}^k)$. Then, we look for its optimal configuration $\hat{\mathbf{s}}^k$ via Monte Carlo sampler proposed in Sec. 2.2.1.3.

For a practical reason related to the implementation, we bound the values of $\mathbf{w}$. Specifically, we sweep the weighting coefficients of the prior energy $\mathbf{w}_p$ according to the gradient magnitude and noise level of the input image. Let $\chi = -\ell_{\min} \times \mathbb{E}[\nabla I] + \mathrm{Var}[I_{\sigma^2}]$ be a baseline to accept a new line segment into the current configuration without considering spatial interaction, where I_{σ^2} denotes a smoothed image using a Gaussian kernel with $\sigma^2 = \{1.5, 2.25, 3.5\}$. To reduce computation overhead, we fix the weighting factors of data likelihood energy as $\omega_d^m = -1$ and $\omega_d^v = 1$. We set $\mathbf{w}^1 = [-1, 1, \chi, 0.1\,\chi, 0.01\,\chi, 0.01\,\chi]^\top$, and gradually change χ by 10% of increments, *i.e.*, $\mathbf{w}^2 = [-1, 1, \chi_2, 0.1\,\chi_2, 0.01\,\chi_2, 0.01\,\chi_2]^\top$, where $\chi_2 = 1.1\,\chi$. In our experiments, we set $K = 15$ to create line hypotheses.

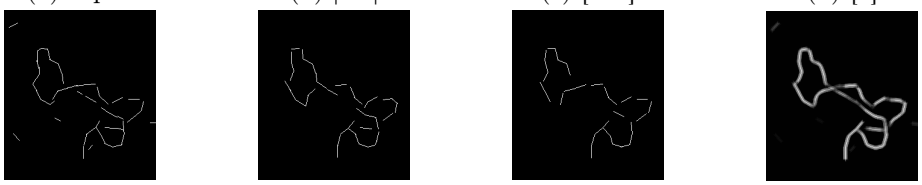
2.2.2.2 Combination of line hypotheses into a probability map

We now have a family of line hypotheses $\hat{\mathcal{S}} = \{\hat{\mathbf{s}}^1, \ldots, \hat{\mathbf{s}}^K\}$ obtained from K different hyperparameter vectors. We jointly use the image data and the line hypotheses. More specifically, the final solution $\mathbf{s}^*$ maximizes not only the probability density but also the consensus among line hypotheses. For each optimal configuration $\hat{\mathbf{s}}^k$, we compute a probability map $\mathcal{P}_k$ of being a line in the image site. Then, we integrate K probability maps into a mixture density $\mathcal{P}_{\hat{\mathcal{S}}}$:

$$\mathcal{P}_k(\mathbf{x}) = \begin{cases} 1 & \text{if } \exists s_i^k \in \hat{\mathbf{s}}^k, \mathbf{x} \in s_i^k, \\ \frac{1}{2} & \text{if } \exists s_i^k \in \hat{\mathbf{s}}^k, \mathbf{x} \in A(s_i^k), \\ 0 & \text{otherwise,} \end{cases} \qquad \mathcal{P}_{\hat{\mathcal{S}}}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{P}_k(\mathbf{x}). \tag{2.15}$$



(a) Input (b) $|\nabla I|$ (c) [177] (d) [6]



(e) $\hat{\mathbf{s}}^1$ (f) $\hat{\mathbf{s}}^2$ (g) $\hat{\mathbf{s}}^3$ (h) $\mathcal{P}_{\hat{\mathcal{S}}}$

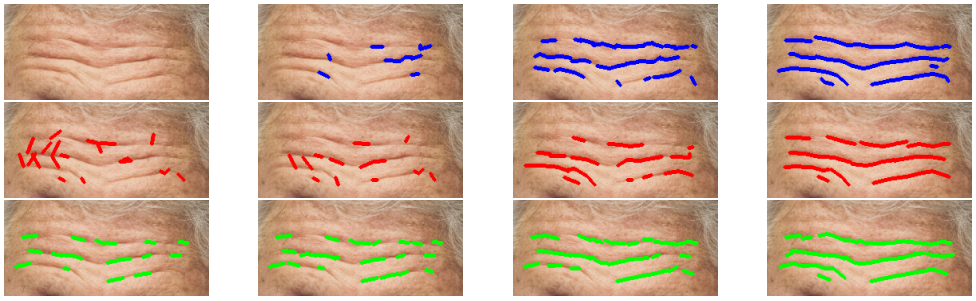
Figure 2.3: Given the input image (a), we compute the gradient magnitude (b). Mathematical morphology operator, path opening [177], is applied on such gradient magnitude image (c). Linearity score of each pixel is drawn by the supervised feature learning algorithm [6] (d). We provide line hypotheses (e)–(g) associated with different hyperparameter vectors. Composition result (h) is equivalent to mixture probability density, and it highlights pixels corresponding to linear structures.

Fig. 2.3 compares image gradient magnitude, morphological filtering [177], supervised feature learning [6], line hypotheses, and the mixture density. Since the input image contains many high frequency components, its gradient also highlights non-linear structures in the background. While the morphological filter accentuates linear structures, its performance depends on the setting of path length. Supervised learning method requires high quality of a training dataset and corresponding ground truth images. Depending on the setting of hyperparameter vectors, the MPP model leads incomplete detection results as shown in Fig. 2.3. (e)–(g). We integrate line hypotheses of the proposed MPP model into a mixture density $\mathcal{P}_{\hat{\mathcal{S}}}$. The mixture density shows the consensus between line hypotheses in the sense that the pixels corresponding to line structures are more highlighted when compared to [6, 177].

We assume that the most promising hyperparameter vector draws a configuration which is more akin to the mixture density. We compute the *correlation-coefficient* (CC) between $\mathcal{P}_{\hat{\mathcal{S}}}$ and $\mathcal{P}_k$'s to analyze coherence of line detection results. That is

$$k^* = \underset{k=\{1,\ldots,K\}}{\operatorname{argmax}} \operatorname{CC}(\mathcal{P}_{\hat{\mathcal{S}}}, \mathcal{P}_k), \tag{2.16}$$

$$\operatorname{CC}(\mathcal{P}_{\hat{\mathcal{S}}}, \mathcal{P}_k) = \frac{\sum_{\mathbf{x}} \left(\mathcal{P}_{\hat{\mathcal{S}}}(\mathbf{x}) - \mathbb{E}[\mathcal{P}_{\hat{\mathcal{S}}}]\right) \left(\mathcal{P}_k(\mathbf{x}) - \mathbb{E}[\mathcal{P}_k]\right)}{\sqrt{\sum_{\mathbf{x}} \left(\mathcal{P}_{\hat{\mathcal{S}}}(\mathbf{x}) - \mathbb{E}[\mathcal{P}_{\hat{\mathcal{S}}}]\right)^2 \sum_{\mathbf{x}} \left(\mathcal{P}_k(\mathbf{x}) - \mathbb{E}[\mathcal{P}_k]\right)^2}}, \tag{2.17}$$



(a) $T_0 = 100$ (b) $T_{10^2} = 21.67$ (c) $T_{10^3} = 14.47$ (d) $T_{10^4} = 10.86$

Figure 2.4: We provide intermediate sampling processes when the temperature parameter T_t is decreasing. The results shown in first row are obtained without specifying seed segment. For the second row, we randomly set 20 seed segments and run the algorithm. For the third row, we initialize 20 line segments which are highly corresponding to underlying curvilinear structures. The algorithm converges toward almost the same solution regardless of the initial state.

where k^* represents the index of the most reliable hyperparameter vector.

2.2.3 Curvilinear structure extraction from reduced sampling space

The line hypotheses span a configuration space $\mathbb{S} \subset \Psi$ which will be considered as a new sample space. Since the size of $\mathbb{S}$ is significantly reduced compared to the original sample space Ψ, the optimization process becomes more tractable in terms of convergence time and detection accuracy.

We redefine the data energy by adding a new energy term as follows:

$$U'_d(s_i) = U_d(s_i) + U^h_d(s_i), \quad U^h_d(s_i) = \int_0^1 -\log \mathcal{P}_{\hat{\mathcal{S}}}(s_i(t))\, dt, \tag{2.18}$$

where $U^h_d(s_i)$ quantifies the consensus among line hypotheses with respect to the line segment s_i. We stimulate the modified probability density over the reduced sample space $\mathbb{S}$ with the most promising hyperparameter vector $\mathbf{w}^{k^*}$:

$$\mathbf{s}^* = \underset{\mathbf{s}\epsilon\mathbb{S}}{\operatorname{argmin}} \sum_{i=1}^{\#(\mathbf{s})} U'_d(s_i) + \sum_{i\sim j} U_p(s_i, s_j; \mathbf{w}^{k^*}). \tag{2.19}$$

2.2.4 Experiments

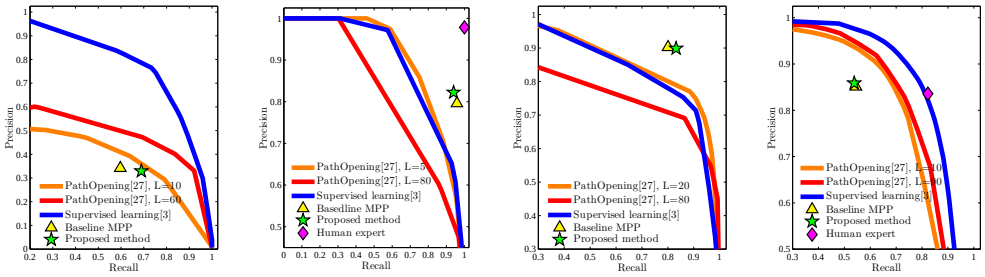
We test the proposed method on a wide range of datasets: road cracks, facial wrinkles, DNA filaments[1], and retinas. Test images of the defects on the road pavements and ground-truth are courtesy of Chambon *et al.* [23]. The facial wrinkle images are collected on the Internet, and forehead areas are manually selected for the experiments. We use the DRIVE dataset [173] to test the proposed algorithm on retina images.

For all test sequences, we fix the modeling parameters as follows: $\ell_{\min}$ is set to 5 pixels and $\ell_{\max}$ = 20 pixels. The orientation θ is varying from $-90°$ to $90°$ with increments of $2°$. The marginal distance of connected segments ϵ is fixed to 2 pixels, and the maximum angular difference of aligned segments τ is $30°$. For the SA, the initial temperature T_0 is set to 100, and it follows the logarithm cooling schedule $T_t = {T_0}/{\log(1+t)}$, where t denotes the number of the current iteration. We start the sampling process with the empty configuration. However, careful choice of initial segments can speed up the convergence of the algorithm (see Fig. 2.4). The computational time depends on the image resolutions; it takes less than a minute for the experimental images having 300×400 pixels, approximately. We use a PC with a 2.9 GHz CPU (4 cores) and 8 GB RAM.

To compare the performances of the proposed method with the state-of-the-art techniques, we apply the path opening operator [177] on the gradient magnitude images by controlling the length parameters. For the supervised feature learning algorithm [6], we train 15 images for each dataset. In our experiments, we use the original implementations of path opening operator[2] and supervised feature learning algorithm[3].

Fig. 2.5 shows the precision-and-recall curves for four test images. To obtain the curve of the comparison methods [6, 177], we tune thresholds on line detection results. The baseline MPP is selected from the line hypotheses among which it shows the best performance. The performances of the supervised learning algorithm are controlled by the quality of the training set; hence, it shows low performances on WRINKLE and DNA datasets, which are composed of noisy images with various sizes. In particular, the ground truth set of the WRINKLE dataset is based on subjective perception. While the morphology operator enhances linear structures on gradient magnitude images, it is required to specify the length of the linear structures according to the target applications. Since the pixelwise comparison fails to incorporate the geometry similarity with the ground-truth, the proposed algorithm shows lower scores on the CRACK and RETINA

[1] `https://www.biochem.wisc.edu/faculty/inman/empics/dna-prot.htm`
[2] `http://hugues.zahlt.info/91.html`
[3] `http://cvlab.epfl.ch/page-108936-en.html`



(a) CRACK (b) WRINKLE (c) DNA (d) RETINA

Figure 2.5: Precision-and-recall curves for pixelwise segmentation of curvilinear structures using path opening operator [177] with different setups of length, supervised feature learning [6], baseline MPP, and the proposed method.

datasets. More specifically, the proposed algorithm detects slightly shifted lines for the CRACK image.

Fig. 2.12 compares the detection results of the proposed MPP model with the manually labeled image by human expert, morphology operator [177], supervised feature learning algorithm [6], and baseline MPP. For a fair comparison, we set the threshold values of the competing algorithms [6, 177] to obtain the closest recall scores to the proposed algorithm. Blue pixels denote perfectly matching regions as compared with the ground-truth. Green and red pixels show over-detected and under-detected results, respectively. The main strength of the proposed algorithm is that it ensures stable performances for all datasets without any parameter estimation procedure. The proposed algorithm extracts the most salient line structures in the input image. On the other hand, the proposed algorithm suffers from under-detection when the width of the line structure is varying, for example, see the result for the RETINA. Such drawback can be overcome if we introduce an additional parameter for width of the line segment in our MPP model.

2.2.5 Concluding remarks

I have presented an MPP-based model to reconstruct curvilinear structures via vectorized line segments. For the data likelihood, the density function computes rotated gradient magnitude and intensity variance. Prior energies of the proposed MPP model define interactions of the local configuration in terms of coupling energy states and overlapping areas. We have proposed a new optimization scheme which is not biased by the parameter selection in the MPP model. We used an advanced RJMCMC sampler with different hyperparameter vectors to obtain line hypotheses. The line hypotheses span a feasible

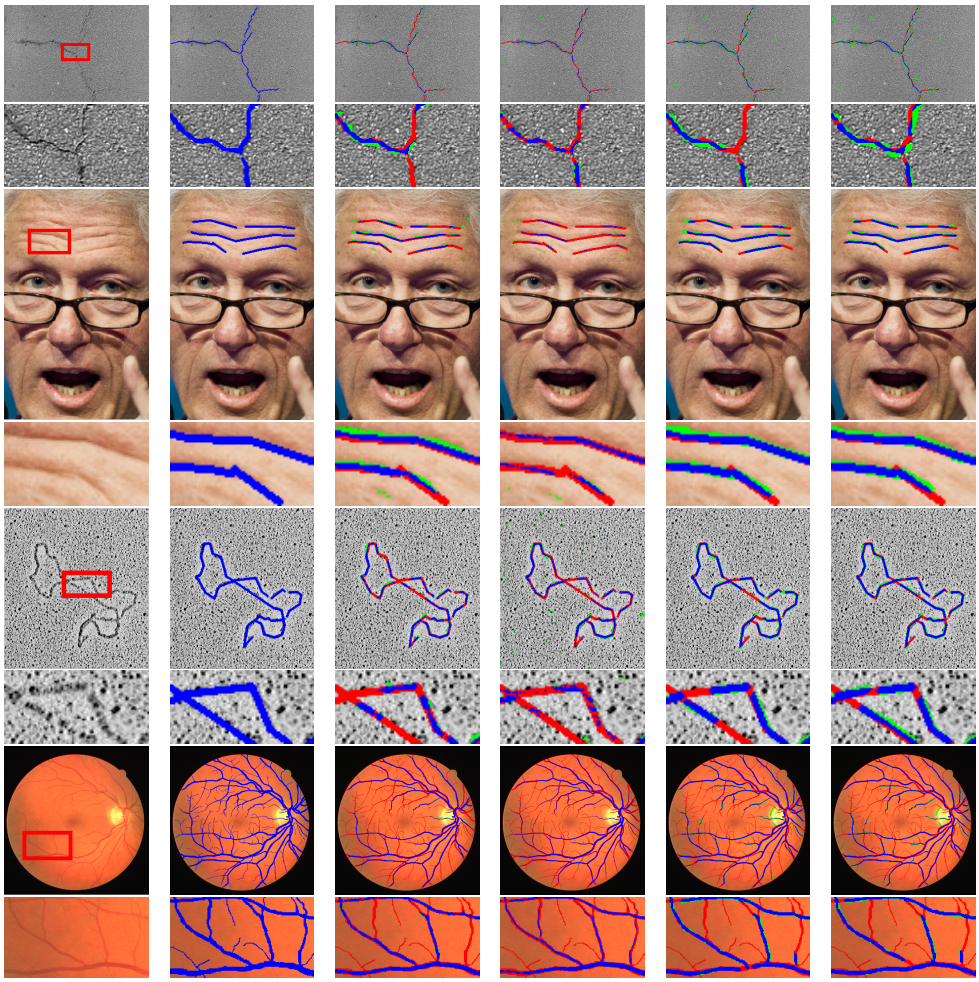


Figure 2.6: We visualize the localization of the curvilinear structures on input images (a). We compare with the results of a manually labeled image by a human expert (b), morphological filtering [177] (c), supervised feature learning [6] (d), baseline MPP (e), and the proposed algorithm (f). Threshold values of (c) and (d) are chosen to achieve the closest recall scores to the proposed method. We use blue pixels to indicate areas which are completely corresponding to (b). Green and red pixels denote over-detected and under-detected areas, respectively, as compared with ground-truth. The name of the test images is from top to bottom: CRACK, WRINKLE, DNA, and RETINA.

sample space, so that the final solution interprets underlying curvilinear structures more faithfully. We have shown line detection results on a wide rage of datasets, and compared the performances of the proposed method with morphological filtering [177], supervised learning [6], and baseline MPP method. The whole optimization process is friendly designed to the parallel implementation; thus, the computational time can be further reduced by applying the parallel Monte Carlo sampler [195].

The main limitation of the method concerns its accuracy and suitability for real-life applications. Even though we showed that the proposed unsupervised approach performs fairly well for different kinds of images, the obtained accuracy may be not sufficient to be applied in a real-world scenario. Therefore, the current trend consists in applying supervised learning for applications dealing with curvilinear structure extraction.

2.3 Inference of curvilinear structure

In the following, I present our model based on supervised learning. More specifically, we employ structured SVM to learn the ranking function that predicts the correspondence between the given line segments and the latent curvilinear structures. To reconstruct the curvilinear structure, we build a graph using the output rankings of the line segments, and then look for the longest geodesic paths within this graph.

2.3.1 Overview of the proposed method

We first define notations and provide an overview of the proposed method (see Figure 2.7). Assume that an image I contains a curvilinear structure. We denote the latent curvilinear structure $\Omega : I \mapsto \{0, 1\}$ for any pixel $\mathbf{x}$ of the image I:

$$\Omega(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is on the curvilinear structure,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.20}$$

This function is also related to a ground truth map, which is manually labeled, for the machine learning framework and for the performance evaluation. We compute a curvilinear feature map $\phi : I \mapsto \mathbb{R}$ that represents oriented gradient information (see Section 2.3.2.1). Since information embedded in a single pixel is limited to infer the latent spatial patterns, we exploit image patches to compute input feature vectors. Let $\mathbf{P}_\mathbf{x}$ be a patch of the feature map values within $\sqrt{M} \times \sqrt{M}$ size of square window centered at $\mathbf{x}$, *i.e.*, $\mathbf{P}_\mathbf{x} = \{\phi(\mathbf{x}') \mid \|\mathbf{x} - \mathbf{x}'\|_\infty \leq \frac{\sqrt{M}}{2}\}$. Using a rotation matrix $\mathbf{R}_\theta$ defined on Euclidean image space, we can rotate a patch with respect to the given orientation θ such as

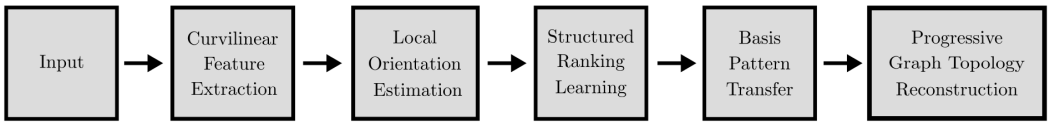


Figure 2.7: Overview of the proposed inference method.

$\mathbf{P}_{\mathbf{x},\theta} = \{\phi(\mathbf{x}') \mid \|\mathbf{R}_\theta^\top(\mathbf{x}-\mathbf{x}')\|_\infty \leq \frac{\sqrt{M}}{2}\}$.

We use graph theory for shape simplification of the complex curvilinear structure. For the pixels showing higher rankings, we build an undirected and weighted graph $G = (V, E)$. Then, we look for the longest geodesic path which corresponds to the coarset curvilinear structure in the image. We iteratively reconstruct the curvilinear structure by collecting paths that connect the remotest vertices in the graph. Consequently, the proposed algorithm can represent the different levels of detail in the latent curvilinear structure using the minimum number of pixels.

2.3.2 Curvilinear Feature

This section is devoted to compute the curvilinear feature descriptor that is used for the inputs of the learning system. We perceive the latent curvilinear structure based on inconsistency of background textures and its geometric characteristics. In other words, a sequence of pixels corresponding to the curvilinear structure has different intensity values compared to its surroundings, and shows thin and elongated shape. Thus, we compute multi-direction and multi-scale image gradients to detect locally oriented image features.

2.3.2.1 Curvilinear feature extraction

We obtain oriented gradient maps using a set of steerable filters [54, 86, 150]. Before applying the convolution operations, we normalize the training and test images to remove the effects of various illumination factors:

$$\tilde{I} = \frac{1}{1 + e^{-\frac{I - \mathbb{E}[I]}{\max(I) - \min(I)}}}, \tag{2.21}$$

where $\mathbb{E}[I]$ is the sample mean of the image.

The steerable filters are created by the second order derivative of isotropic 2D Gaussian kernels. Let $\mathbf{f}_{\theta,\sigma}$ be a steerable filter that accentuates image gradient magnitude for direction θ at scale σ. To take into account varying orientations and widths of the

curvilinear structure, a feature map ϕ is combined by multiple filtering responses:

$$\phi = \frac{1}{|\Theta|} \sum_{\theta} \max_{\sigma} \{\mathbf{f}_{\theta,\sigma} * \tilde{I}\}, \tag{2.22}$$

where $|\Theta|$ denote the total number of orientations. We first find the maximum filtering responses of the scale spaces, and then average for all directional responses. In this work we consider 8 orientations and 3 scales.

2.3.2.2 Local orientation estimation

In the previous subsection, we evaluate the presence of curvilinear structure from the amount of gradient magnitudes in image patches. Complex shaped curvilinear structure also consists of various local orientations.

Assume that $\mathbf{B} \in \{0,1\}^M$ be a basis spatial pattern of the curvilinear structure for the baseline orientation ($\bar{\theta} = 0°$). We manually design $\mathbf{B}$ as a simple line shape template which highlights the middle of image patch (see Figure 2.8). If an image patch $\mathbf{P_x}$ contains a part of curvilinear structure, pixels on the curvilinear structure are sequentially accentuated towards a particular direction θ. Recall that the geometric properties of the curvilinear structure, it is rotatable and symmetric. We can rotate image patch to be aligned with the basis spatial pattern.

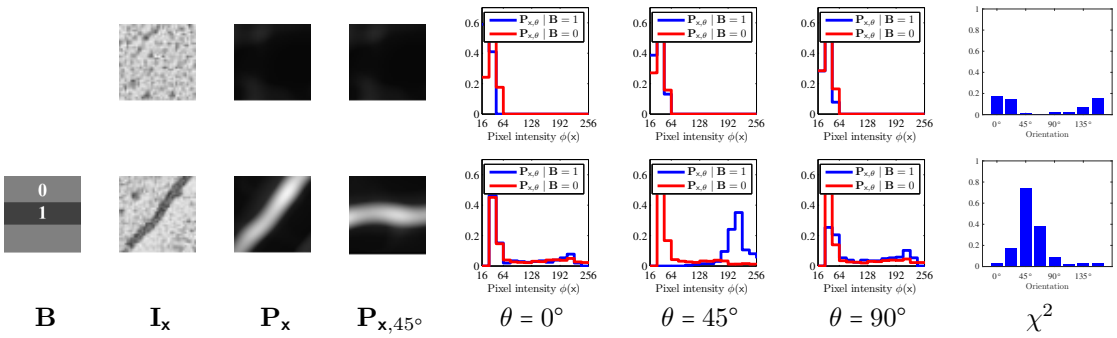


Figure 2.8: In this figure, we compare the statistics of pixel values in the rotated image patches with respect to the given binary mask $\mathbf{B} \in \{0,1\}^M$. Assume that $\mathbf{B}$ encodes a basis spatial pattern of the curvilinear structure toward the baseline orientation ($\bar{\theta} = 0°$), where darker gray color refers to 1. To estimate local orientation of the input image patch, we rotate it and find the orientation which maximizes the statistical difference of pair distributions. If the input image patch does not contain curvilinear structure (upper row), there is no meaningful statistical difference for any orientation.

More specfically, we estimate the local orientation shown in image patches using

statistical difference of the areas determined by $\mathbf{B}$. For a rotated image patch $\mathbf{P}_{\mathbf{x},\theta}$, we compute normalized histograms of the pixels which are labeled as curvilinear structure $p_{\mathbf{x},\theta} = \mathsf{hist}\left(\mathbf{P}_{\mathbf{x},\theta} \mid \mathbf{B}\right)$ and its counterpart $q_{\mathbf{x},\theta} = \mathsf{hist}\left(\mathbf{P}_{\mathbf{x},\theta} \mid \neg\mathbf{B}\right)$, where $\mathsf{hist}\left(\cdot\right)$ denotes histgram of the given distribution with 32 bins. Similar to [2], we employ χ^2 test to compute statistical difference of two distributions:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, \chi^2\left(p_{\mathbf{x},\theta}, q_{\mathbf{x},\theta}\right), \tag{2.23}$$

$$\chi^2(p,q) = \frac{1}{2}\sum_i \frac{\left(p(i)-g(i)\right)^2}{p(i)+g(i)}. \tag{2.24}$$

Figure 2.8 compares statistics of the image patches with respect to different orientations. If an image patch contains curvilinear structure, χ^2 test score shows uni-modal distribution which shows a peak at the dominant orientation.

2.3.3 Learning

We regulate the orientation and the shape of the input patches with respect to the basis spatial pattern $\mathbf{B}$. In this section, we aim to learn a function that predicts structured output rankings of the input image patches. Thus, the output rankings indirectly infer the spatial patterns of the curvilinear structure for image patch.

Let $\mathsf{vec}\left(\cdot\right)$ be an operator to convert an image patch into a column vector. For a pixel $\mathbf{x}_i$, the input feature vector is defined as $\mathbf{z}_i = \mathsf{vec}\left(\mathbf{P}_{\mathbf{x}_i,\theta_i} \mid \mathbf{B}\right) \in \mathbb{R}^N$ and $y_i \in \mathbb{R}^+$ denote the corresponding rank value. For a feature vector, we exploit subsampled pixels on the linear template to avoid data imbalance. Thus, the dimension of feature vector N is small when compared to the total number of pixels within the patch, *i.e.*, $N \leq M$.

For the setup of machine learning, a training dataset $\mathcal{D} = \{(\mathbf{z}_i, y_i)\}_{i=1}^K$ consists of the K input-and-output pairs. Let $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\} \in \mathcal{Z}$ be an unordered list of the input feature vectors and $\{y_1, \ldots, y_K\} \in \mathcal{Y}$ be the corresponding classes. The input feature vector $\mathbf{z}$ is built from the curvilinear feature map ϕ. Our goal is to learn a ranking function $h(\mathbf{z})$ which assigns a global ordering (ranking) of feature vectors: $h(\mathbf{z}_i) > h(\mathbf{z}_j) \Leftrightarrow y_i > y_j$. The inner product of the model parameter and a feature vector $\mathbf{w}^\top\mathbf{z}$ is used to predict ranking score of the given image patch.

In our work **Structured SVM** framework [90, 139] is employed to exploit the structure and dependencies within the output space $\mathcal{Y}$. To encode the structure of the output space, a loss value Δ of each input feature vector is defined: $h(\mathbf{z}_i) > h(\mathbf{z}_j) \Leftrightarrow \Delta_i < \Delta_j$. The loss value evaluates the quality of the learning system. Intuitively, the input image patches containing curvilinear structure are comparable to the basis spatial pattern.

Hence, as an image patch is similar to the basis spatial pattern, we should put it on a higher ranking. Specifically, we compute the loss value as the overlapping ratio between image patch from the groundtruth map Ω and the basis $\mathbf{B}$:

$$\Delta_i = 1 - \frac{|\mathbf{\Omega}_{\mathbf{x}_i,\theta_i} \cap \mathbf{B}|}{|\mathbf{\Omega}_{\mathbf{x}_i,\theta_i} \cup \mathbf{B}|}, \tag{2.25}$$

where $\mathbf{\Omega}_{\mathbf{x}_i,\theta_i}$ denotes a rotated image patch referring to groundtruth map Ω. The objective function of Structured SVM with a single slack variable ξ is given by:

$$\begin{aligned} \min_{\mathbf{w},\xi\geq 0} \quad & \tfrac{1}{2}\mathbf{w}^\top\mathbf{w} + C\xi \\ \text{s.t.} \quad & \tfrac{1}{|\mathcal{N}|}\sum_{(i,j)\in\mathcal{N}} c_{ij}\mathbf{w}^\top(\mathbf{z}_i - \mathbf{z}_j) \geq \tfrac{1}{|\mathcal{N}|}\sum_{(i,j)\in\mathcal{N}} c_{ij} - \tfrac{\xi}{\Delta_j - \Delta_i}, \\ & \forall (i,j) \in \mathcal{N},\ \forall c_{ij} \in \{0,1\} \end{aligned} \tag{2.26}$$

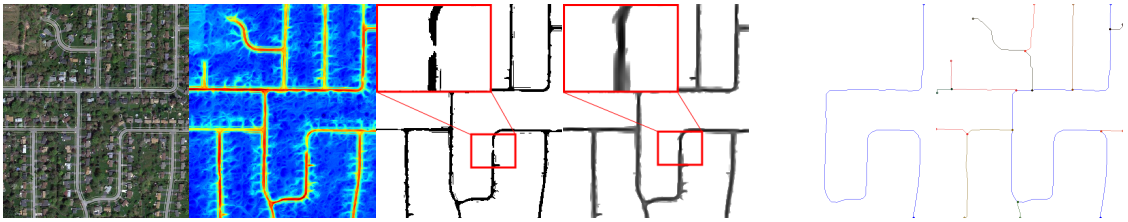
where c_{ij} is the indicator variable to reduce the number constraint in linear complexity $O(K)$:

$$c_{ij} = \begin{cases} 1 & \text{if } \Delta_i < \Delta_j \text{ and } \mathbf{w}^\top\mathbf{z}_i - \mathbf{w}^\top\mathbf{z}_j < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.27}$$

Cutting plain algorithm [188] is employed to solve (2.26). For the details about the Structured SVM optimization, we refer the reader to [90, 139, 188]. In the following section, we plot the initial segmentation map from the structured rankings scores. We also exploit these ranking scores for the shape simplification based on graph traversal algorithm.

2.3.4 Curvilinear Structure Reconstruction

To detect and represent the latent curvilinear structure, various local measures have been proposed. The score of the local measure is regarded as a likelihood probability whether a pixel is on the latent curvilinear structure. Most of the previous works represent the curvilinear structure as a binary map based on the linearity scores of the pixels. This approach easily misinterprets the topological features. In this section, we propose a novel structured score map based on the output ranking scores and the basis spatial pattern. We also develop a graph-based model which is able to organize the topological features of the structure in different levels of detail. Figure 2.9 shows step-by-step processing results to reconstruct the latent curvilinear structure. In other words, we have the information how each pixel is relatively important to represent the underlying curvilinear structure.



(a) (b) (c) (d) (e) (f)

Figure 2.9: For the given image (a), we compute (b) the structured output ranking with Structured SVM. We retain binary map (c) from the highest to $\rho|I|$-th rankings. However, it could be broken. We exploit the basis spatial pattern $\mathbf{B}$ to reconstruct (d) the structured score map. Graph-based representation provides a tool to analyze the topological features of the curvilinear structures. (e) The coarsest curvilinear structure evolves to define (f) branches.

2.3.4.1 Structured score map

We have obtained a score function $h(\mathbf{z}; \mathbf{w})$ that evaluates the compatibility between the input features and the underlying curvilinear structure based on the structured output rankings. From the training dataset, we pre-compute the proportion ρ of pixels being part of the curvilinear structure. Note that the value ρ maximizes F_1 test scores for the groundtruth maps in the training dataset. During the test phase, we retain the output ranking according to the output ranking scores from the highest to the $\rho|I|$-th rank. As shown in Figure 2.9 (c), a binary map can be disconnected.

To avoid unwilling breakpoints, we composite the structured score map $\Pi : I \mapsto \mathbb{R}$ using the binary map, induced by ranking scores $\mathbf{w}^\top \mathbf{z}$, and the basis spatial pattern $\mathbf{B}$. Recall that we initially estimate the ranking score using the shape similarity between oriented image patch of ground truth and the basis spatial pattern in (2.25). Thus, it can be seen as the inverse mapping from output ranking score to the latent curvilinear structure. Let $\mathbf{b} = \mathsf{vec}(\mathbf{B})$ be the column vector version of the binary mask $\mathbf{B}$ and $\mathbf{Q}_{\mathbf{x}_i,\theta_i}$ be the patch of structured score map Π centered at $\mathbf{x}_i$ with θ_i. We minimize the following cost function with respect to $\boldsymbol{\pi}_i = \mathsf{vec}\left(\mathbf{Q}_{\mathbf{x}_i,\theta_i}\right)$:

$$J = \min \sum_{\mathbf{x}_i \in I'} \|\mathbf{b} - (\mathbf{w}^\top \mathbf{z}_i)\boldsymbol{\pi}_i\|_2^2. \tag{2.28}$$

This is a least square problem, so that the solution is found at points which satisfy $\frac{\partial J}{\partial \boldsymbol{\pi}} = 0$. To reconstruct the structured segmentation map, we synthesize obtained patches $\mathbf{Q}_{\mathbf{x}_i,\theta_i} = \mathsf{vec}^{-1}(\hat{\boldsymbol{\pi}}_i)$ at the subsampled grid points on image $\mathbf{x}_i \in I'$, where $\hat{\boldsymbol{\pi}}_i = (\mathbf{w}^\top \mathbf{z}_i)^{-1}\mathbf{b}$. We refer the readers to [105] for the implementation details of the related texture synthesis

Algorithm 2 Progressive curvilinear path reconstruction

Inputs:
  $G' = (V', E')$ ~ a subgraph of G; and
  $\hat{\ell}$ ~ a minimum length of the curvilinear structure
Output:
  P ~ a set of vertices corresponding to the simplified curvilinear structure
$P \leftarrow \varnothing$
`longest_path_length` $\leftarrow |V'|$
while do
    Compute the longest geodesic path T in G' using [30]
    `longest_path_length` $\leftarrow |T|$
    if `longest_path_length`$< \hat{\ell}$ **then**
        break
    end if
    $P \leftarrow P \cup T$
    $w(\mathbf{u}, \mathbf{v}) \leftarrow 0$, $\forall \{\mathbf{u}, \mathbf{v}\} \in T$, Update all edge weights on the path T as 0 for the given subgraph G'
end while

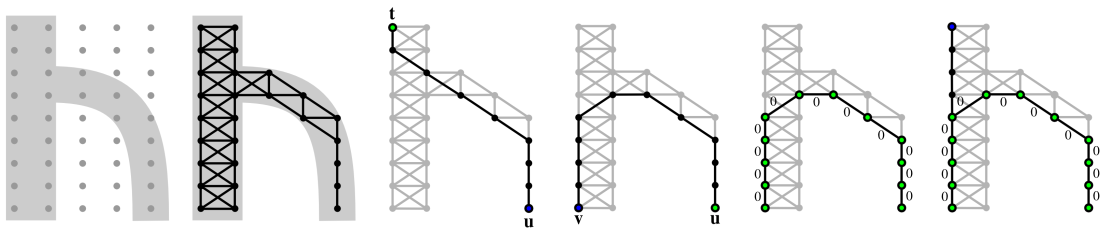
technique.

2.3.4.2 Progressive curvilinear path reconstruction

We consider the graph $G = (V, E)$ where V is the set of pixels. Two pixels $\mathbf{u}$ and $\mathbf{v}$ are connected by the edge if and only if $\|\mathbf{u} - \mathbf{v}\| \leq \sqrt{2}$. Moreover, we assign a weight for each edge $\{\mathbf{u}, \mathbf{v}\} \in E$ as $w(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\| \frac{\Pi(\mathbf{u}) + \Pi(\mathbf{v})}{2}$. A path P in the graph G is a sequence of distinct vertices such that consecutive vertices are adjacent. The length of P is the sum of the weights of its edges, and the distance $\mathsf{dist}(\mathbf{u}, \mathbf{v})$ between two vertices $\mathbf{u}$ and $\mathbf{v}$ is the minimum length of a path from $\mathbf{u}$ to $\mathbf{v}$. The eccentricity $\mathsf{ecc}(\mathbf{v})$ denotes the maximum distance from the vertex $\mathbf{v}$ to a vertex $\mathbf{u} \in V$, *i.e.*, $\mathsf{ecc}(\mathbf{v}) = \max_{\mathbf{u} \in V} \mathsf{dist}(\mathbf{u}, \mathbf{v})$. The diameter $\mathsf{diam}(G)$ of G equals $\max_{\mathbf{v} \in V} \mathsf{ecc}(\mathbf{v})$, *i.e.*, it is the maximum distance between two vertices in G.

To simplify the latent curvilinear structure, we look for long geodesic paths in the subgraph G' of G induced by the pixels with structured score map Π. More precisely, our algorithm computes a diameter of G', *i.e.*, a shortest path T with length $\mathsf{diam}(G')$. This path T is added in the simplified curvilinear structure, then the path T is contracted into a single (virtual) vertex. For the implementation, the weight of all edges of T become 0. We repeat this process till the diameter of the subgraph is larger than pre-defined path length $\hat{\ell}$. The entire procedure is summarized in Algorithm 2.

The time-consuming part of the proposed algorithm is the computation of a diameter of G'. Rather than computing all pair distances (which requires a linear number of



(a) (b) (c) (d) (e) (f)

Figure 2.10: Toy example of the proposed curvilinear structure inference algorithm: (a) input image contains a curvilinear structure which is denoted by gray color; (b) subgraph G' is induced from the structured segmentation map; (c) and (d) show the intermediate processes of the 2-sweep algorithm starting from vertex $\mathbf{t}$ to find a diameter of the subgraph; (e) we assign 0 weight for all edges on the path; and (f) we repeat the process and add branches if the path length is larger than pre-defined length $\hat{\ell}$.

applications of Dijkstra's algorithm), we use an efficient heuristic algorithm called *2-sweep algorithm* [30]. The 2-sweep algorithm randomly picks a vertex $\mathbf{t}$ in G', then performs Dijkstra's algorithm from $\mathbf{t}$ to find a vertex $\mathbf{u}$ at the maximum distance from $\mathbf{t}$, *i.e.*, $\mathsf{dist}(\mathbf{u},\mathbf{t}) = \mathsf{ecc}(\mathbf{t})$. Then, it computes (using Dijkstra's algorithm) a path from $\mathbf{u}$ to a vertex $\mathbf{v}$ at the maximum distance from $\mathbf{u}$. The length of the second path (from $\mathbf{u}$ to $\mathbf{v}$) is a good estimation of $\mathsf{diam}(G')$. Note that this algorithm is able to compute the exact diameter if the considered graph has a tree structure [16]. Topologically speaking, most of latent curvilinear structures are very close to trees [66, 190]. Therefore, the 2-sweep algorithm is well adapted to reconstruct the tree-like curvilinear structures. For a better understanding, we schematically explain the intermediate steps of the proposed curvilinear structure simplification algorithm in Figure 2.10.

2.3.5 Experimental results

In this section, we first discuss the parameters of the proposed algorithm and datasets. We then compare the quantitative and qualitative results of the proposed algorithm and those of competing models proposed by [52], [110], [5], and [169].

2.3.5.1 Parameters and Datasets

The proposed algorithm requires few parameters to compute the curvilinear feature descriptor ϕ. We use 8-different orientations in this work: $\Theta = \{0°, 22.5°, 45°, 57.5°, 90°, 112.5°, 135°, 157.5°\}$. The size of steerable filters is fixed to 21×21 pixels. The scale factors σ^2

and the minimum path length $\hat{\ell}$ are adaptively selected for each dataset to obtain the best performances. The binary spatial pattern $\mathbf{B}$ is based on the physical attribute of the dataset in that thickness τ of curvilinear structure shows various depending on the dataset. For Structured SVM training, we sample 2000 image patches on each dataset. All patch sizes are fixed to $M = 33^2$. C which controls the relative importance of slack variables is set to 0.1 for all datasets. We choose the set of parameters for the SSVM training via 3-fold cross validation [99] which maximizes the average F_1 score [134] of the training set.

We test our curvilinear structure model on the following public datasets:

- **Aerial** [169]: The dataset contains 14 remote sensing images of road networks. We select 7 images for the training and 7 images for the test, respectively. We use $\hat{\ell} = 80$, $\sigma^2 = \{4, 8, 12\}$, and $\tau = 9$.

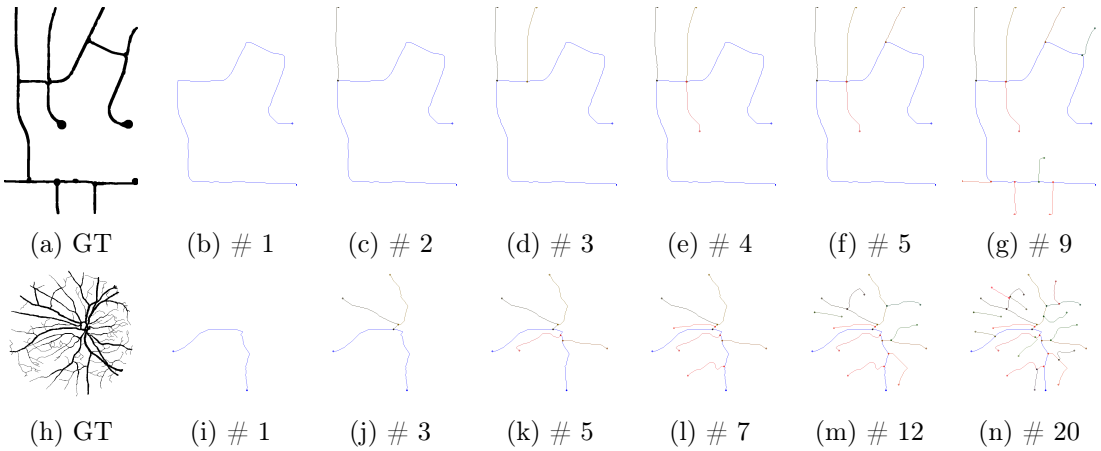
- **Cracks** [24]: Images of the dataset correspond to road cracks on the asphalt surfaces. We use 6 images to train and test the algorithms on different 6 images. For this dataset, we set to $\hat{\ell} = 30$, $\sigma^2 = \{2, 4, 8\}$, and $\tau = 3$, respectively.

- **DRIVE** [174]: The dataset consists of 40 retina scan images with manual segmentation by ophthalmologists to evaluate the blood vessel segmentation algorithms. We use 20 images for the training and 20 images for the test, respectively. The path length $\hat{\ell}$ is set to 40. The scale factors and thickness are set to $\sigma^2 = \{2, 4, 8\}$ and $\tau = 5$, respectively.

- **RecA** [88]: We collect electron microscopic images of RecA proteins on DNA which contain filament structure. We use 4 training images and 4 test images. We use $\hat{\ell} = 30$, $\sigma^2 = \{4, 8, 12\}$, and $\tau = 5$.

2.3.5.2 Evaluations

The proposed algorithm progressively reconstructs the curvilinear structure by adding a long path on the subgraph. Figure 2.11 shows the intermediate steps of the proposed curvilinear structure simplification algorithm for DRIVE dataset. Unlike the previous models, the proposed algorithm is able to show different levels of detail for the latent curvilinear structure. Such information to visualize shape complexity of the curvilinear structure cannot be retrieved by setting a threshold. In practice, a few number of iterations is required to converge the algorithm and each step to find a long path takes less than milliseconds for the computation. For the experiments, we used a PC with a 2.9



(a) GT (b) # 1 (c) # 2 (d) # 3 (e) # 4 (f) # 5 (g) # 9

(h) GT (i) # 1 (j) # 3 (k) # 5 (l) # 7 (m) # 12 (n) # 20

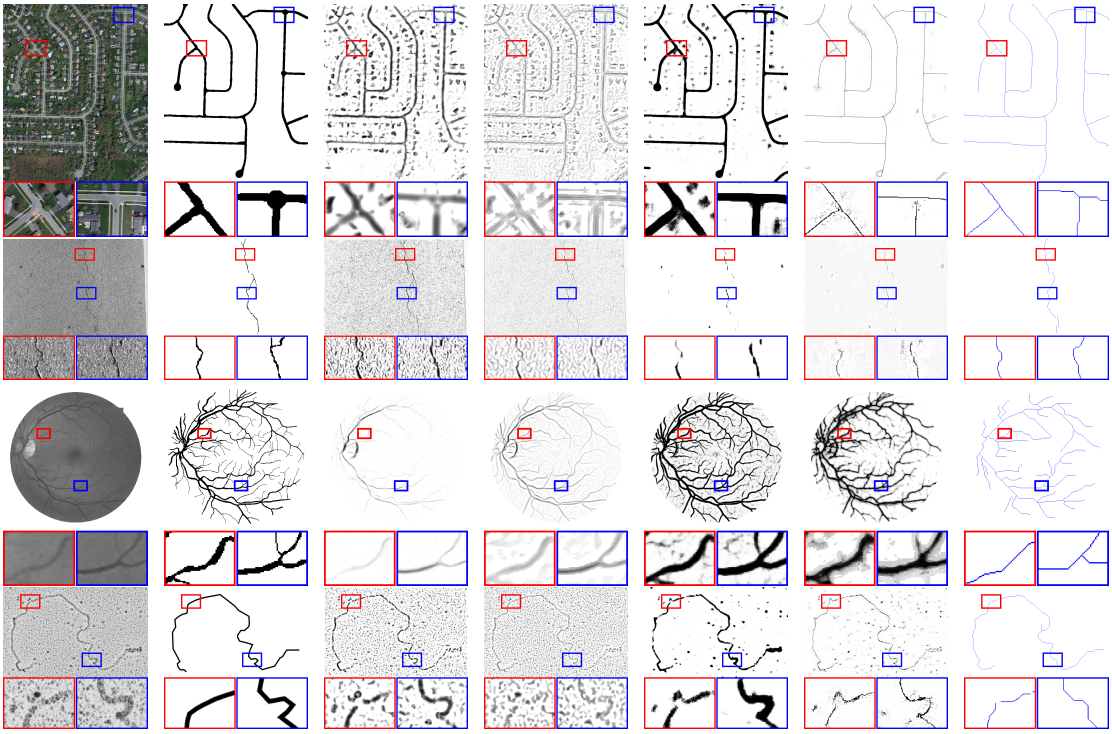
Figure 2.11: Intermediate steps of the curvilinear structure reconstruction for a retina image. We iteratively reconstruct the curvilinear structure according to topological importance orders. As the iteration goes on, detail structures (layer) appear.

GHz CPU (4 cores) and 8 GB RAM. Moreover, we visually compare the performance of the proposed algorithm with the competing algorithms. Figure 2.12 shows the results for Aerial, Cracks, DRIVE, and RecA datasets, respectively. The proposed algorithm is the most suitable to show the topological information of the latent curvilinear structures.

For the quantitative evaluation, we provide F_1 scores of the proposed algorithm and the state-of-the-art models in Table. 2.1. The measure of true positive is sensitive for the misalignment; therefore, we consider surrounding pixels of the detection results as the true positive if a predicted point is falling into the ground truth with a small radius (equivalent to its thickness parameter τ) similarly to [169]. We also provide the average proportion of pixels to represent the curvilinear structures. It shows that the proposed algorithm efficiently draws the curvilinear structures using smaller number of pixels than the other algorithms.

The proposed algorithm achieved the best F_1 scores for all except the DRIVE dataset. It is because the proposed algorithm use the fixed thickness τ to describe linear structure in the binary mask $\mathbf{B}$; however, the images consisting of DRIVE dataset exhibit varied thicknesses of blood vessels and many junction points. In other words, we obtain a model parameter regarding to the manually designed binary pattern $\mathbf{B}$. To overcome this drawback, we plan to exploit generative binary patterns based on the training images in the future, which is possible in our framework.

It is worth mentioning the risk of over-fitting in curvilinear segmentation task based on machine learning. Manual segmentation (ground truth) contains many errors around



(a) Input (b) GT (c) [52] (d) [110] (e) [5] (f) [169] (g) Ours

Figure 2.12: Illustration of the curvilinear structure reconstruction results on Road, Crack, DRIVE, and RecA datasets (top to bottom). We compare the results of Frangi *et al.*'s [52], Law & Chung's [110], Becker *et al.*'s [5], Sironi *et al.*'s [169], and the proposed algorithm (left to right).

boundaries and minutiae components. Also, the number of training data employed in this work is relatively small due to the difficulties of making the accurate annotations. Specifically, since RecA dataset has only four training images, there is a high risk of over-fitting to the training set. It is remarkable that the proposed algorithm shows good performance for all datasets without over-fitting problem when compared to other learning based algorithms [5, 169].

2.4 Concluding remarks

I have described two classification approaches which are based on a template concept and aim to reconstruct curvilinear structures from image data. The structures are modeled as a set of line segments with different orientations. The first approach is unsupervised and based on Marked point process and Monte Carlo sampling. One of key innovations

		DRIVE	RecA	Aerial	Cracks
(a) F_1	Frangi *et al.* [52]	0.33	0.33	0.32	0.056
	Law & Chung [110]	0.43	0.21	0.25	0.085
	Becker *et al.* [5]	0.50	0.45	0.53	0.23
	Sironi *et al.* [169]	**0.55**	0.50	0.55	0.27
	Proposed (Graph)	0.36	**0.59**	**0.59**	**0.38**
(b) $\rho(\%)$	Frangi *et al.* [52]	37.09	16.14	22.19	28.64
	Law & Chung [110]	19.60	29.82	29.51	33.00
	Becker *et al.* [5]	12.67	9.39	9.76	32.13
	Sironi *et al.* [169]	5.41	5.47	0.83	17.12
	Proposed (Graph)	**0.17**	**0.57**	**0.35**	**1.07**

Table 2.1: Comparison of the quantitative performances over the datasets. We provide results with various performance measures: (a) F_1 scores, and (b) average proportion ρ of the pixels being a part of curvilinear structure. Boldfaced numbers are used to show the best score in each test.

of the proposed method is that instead of manually fixing hyperparameters, we proposed a new optimisation algorithm which seeks to maximize the consensus between various line hypotheses, obtained by simulating several Markov chains with different parameter settings.

The second approach is based on the ranking learning system and graph theory. We learned a SSVM-based ranking function corresponding to the plausibility of the latent curvilinear structure. Furthermore, we proposed a novel graphical model that infers the curvilinear structure according to the topological importance. Across the various types of datasets, our model showed good performances to reconstruct the latent curvilinear structure with a smaller number of pixels when compared to the state-of-the-art algorithms.

The main limitation of the proposed methods lies in the lack of genericity, which is a well-known limitation of the template-based approaches. The approaches in this chapter have been tuned for applications dealing with curvilinear structures. In the next chapter, I will present a graph-based model which allows to introduce different soft shape priors into a classification process.

Chapter 3

Hierarchical Model for Image Classification

In this chapter, I present our study on the use of soft shape priors to improve the quality of classification. While classification is usually formulated as the minimization of an objective function, or *energy*, it is difficult to optimize such an energy if it involves shape priors because of their non-local nature [113]. The state-of-the art methods require either the design of an optimizer specific to the particular shape prior [69, 168], or a complex way of incorporating it, when possible, into energies minimizable by standard techniques [37, 194]. Moreover, in the context of remote sensing we require to treat classification as a multi-object multi-class problem, since we typically find multiple instances of the different classes in a single image.

We have proposed a method for shape-aware classification, which takes into account the typical shape of object classes in terms of shape descriptors, such as rectangularity, compactness and solidity. For this, we use a hierarchical data structure, the binary partition tree (BPT) [158], which can be easily augmented to include shape information. However, their traditional greedy construction approach does not yield a good utilization of the shape constraints. We thus proposed a method to iteratively optimize the structure of BPTs to produce better partitions with shape constraints. This enables us to perform multi-object multi-class classification with shape prior, and to enhance BPTs so that they better represent the underlying scenes.

3.1 Related work

Shape and optimization

In the literature regarding shape features for segmentation, we can distinguish contributions that focus on explicit shape models in a *template matching* manner and others geared at incorporating discriminative features, or soft shape priors (e.g., convexity, compactness).

In the context of *template matching*, a number of approaches have proposed to use the active contours framework [31, 114]. The evolution of the curves is usually slow and prone to get trapped in poor local minima. A second family of approaches finds global optimal solutions, but on high-dimensional specially constructed graphs. For example, Schoenemann and Cremers [162] proposed to look for minimal ratio cycles in the product graph of the input image and the shape template. These methods find a globally optimal segmentation in polynomial time on the high-dimensional graph. However, besides their complexity, these contributions are in general not suitable for fitting multiple templates in the same image [94]. Regarding graph-based optimization, a template fitting method to fit was proposed by Fredman and Zhang [53], which must be run repeatedly to account for non-rigid deformations. Other iterative models have been also proposed by coupling graphical models with shape cues [81, 102]. These approaches are computationally intensive, since every loop contains expensive operations.

To include *discriminative features* (e.g., compactness, ellipticity), Slaubaugh and Unal [170] proposed to repeatedly fit ellipses on minimal cuts for an elliptical prior. More recent contributions have managed to express certain shape priors (star-shape [194], compactness [37, 56]) in energies minimizable by traditional s-t cuts. These approaches rely on the ability to express the shape prior in the pairwise interaction term of a Markov random field which might be, when feasible, algorithmically complex (e.g., analyzing intersections with a rotating discrete line around a used-defined point [194]). The trust regions framework [68] can minimize high-order functionals, and has been adapted for certain shape priors (volume, shape moments [68]). It requires to provide a linear approximation of the energy around the current solution. Gorelick et al. [69] expressed the convexity prior as the count of '1-0-1' configurations along a discrete set of lines. They proposed a linear approximation and a dynamic programming algorithm for its efficient computation. It is not clear however how to directly adapt the framework to further priors (e.g., rectangularity index) or to combine different priors under the same scheme. Moreover, most of these techniques do not contemplate the occurrence of multiple object instances. For example, multiple convex segments are indeed penalized in [69] when

counting the '1-0-1' sequences in the scene.

Multi-object segmentation

To cope with the simultaneous detection of multiple objects, the marked point process (MPP) framework has been used to fit an unknown number of parametric shape models. Rectangles [145] and ellipses [41] are some of the geometric elements that have been considered, and I have described our MPP model for line elements in the previous chapter. The optimal solution is sought by stochastic optimization. Karantzalos and Paragios [94] have extended the active contours framework to fit multiple templates in a single image.

To our best knowledge, most recent contributions cited above (e.g., [53, 69, 194]) require to isolate every object (with prior knowledge on their location) and segment it individually.

Hierarchical trees

The notion of hierarchy has been particularly exploited in several image analysis applications, such as the detection of objects by its parts [51] and depth ordering [147].

A number of data structures have been designed to represent images as a hierarchy of regions, such as min/max-trees [159], α-trees [112] and binary partition trees [158]. In our work, we have focused on binary partition trees (BPTs), which have been particularly useful in various domains including remote sensing. Multi-label classifications can be extracted efficiently from a BPT by performing a *cut* on the tree that covers all pixels at arbitrary scales. Recent works have explored the use of shape descriptors during the cuts [192, 196].

BPTs are constructed by successively merging regions with similar colors. Even though BPTs can constitute a good hierarchical approximation to the underlying structure of an image, the bottom-up approach propagates and amplifies the errors produced at the lower scales. As a result, it is very likely that nodes in the BPT will not represent complete significant objects [125, 196]. Previous works mitigated this problem by adding penalties, such as the growth of perimeters [196], the elongation of the regions [104] or edge information [184]. The authors of [196] fitted templates on partially detected objects. These approaches can only alleviate the effect.

As another consequence of the bottom-up approach, shape information cannot be used during construction: the ultimate shape of an object in a branch cannot be predicted by a portion of it. As a result, the criteria used at construction and processing of the trees are different (as in [192, 196]), which limits the feasibility of the tool *as it is* to perform

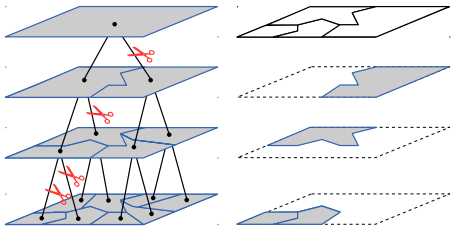


Figure 3.1: A binary partition tree (BPT) is a hierarchical subdivision of an image. An exhaustive partitioning can be extracted by "cutting" branches at different scales.

segmentation with shape priors.

Optimization of hierarchical trees

Optimization of hierarchical trees has been carried out in the area of computational phylogenetics, on the construction of *phylogenetic trees*. These trees represent the evolutionary relationships among species [117]. Several common optimization algorithms (hill climbing, simulated annealing, genetic algorithms) have been applied on the tree structures [63]. The standard moves (known as *branch-swapping* or *swappers*) are nearest neighbor interchange, subtree pruning and regrafting (SPR) and tree bisection and reconnection (TBR). SPR is the pruning/paste of a subtree into another location, while TBR rearranges the subtree before pasting. The most common objective is to maximize the *parsimony* of the tree, i.e., to explain the observed data with the least evolutionary change. We have incorporated the idea of regrafting tree branches. Our optimization objective is however different, and our context requires to define moves that preserve the parent, child and spatial adjacency relations of BPTs.

3.2 Background on binary partition trees

Binary partition trees (BPTs) were pioneered by Salembier and Garrido [158] as a means to represent a set of meaningful image regions in a compact and structured manner. The root node corresponds to the entire image, the following level represents the subdivision of the entire image into two disjoint regions, and so on. It represents then a hierarchical abstraction of an image, which can be navigated to extract meaningful regions at different scales. The typical workflow involves an initial tree construction stage, followed by a second stage of information extraction from the tree. For example, once a tree is constructed, an exhaustive segmentation of the image can be obtained by performing a horizontal "cut" on the structure (see Figure 3.1). In this procedure, commonly referred

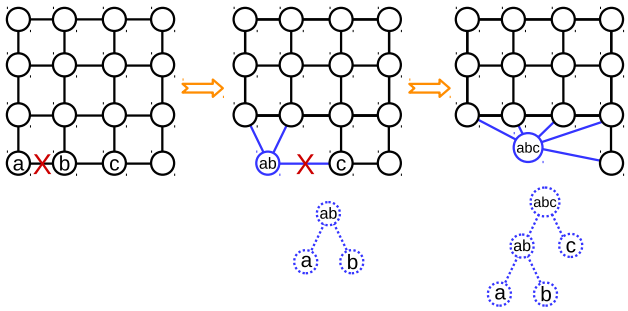


Figure 3.2: A BPT is constructed by iteratively removing edges in a region adjacency graph (RAG). The resulting BPT encodes the history of the merges.

to as *pruning*, branches can be selected at different scales, an inherent advantage of such hierarchical structure. Visual browsing [158], object localization [196] and depth ordering [147] are some of the application domains where BPTs have been used, beyond remote sensing.

The construction of a BPT is performed in a bottom-up fashion, by iteratively clustering pairs of similar regions. The starting point is an initial subdivision of the image represented by a region adjacency graph (RAG), where every node conveys a region and the edges link spatial neighbors (i.e., candidates for merging). The typical initial RAG is the pixel grid, though nothing prevents the approach from being used with other inputs too (e.g., a RAG of small regions containing similar pixels, known as superpixel segmentation). Every edge in the RAG is labeled with a *dissimilarity* value that compares the two associated regions.

BPTs are constructed by following a global mutual best fitting region merging approach [109]: at each iteration, the two most similar regions in the current subdivision are merged together (i.e., the least weighted edge out of all edges in the RAG). When a merge occurs, a new region is added to the BPT, connected to its two corresponding children, as illustrated in Figure 3.2. The process finishes when there are no more edges left in the RAG. A BPT thus records the history of merges that occurred during the execution of a region merging algorithm.

The overall process can be implemented efficiently by using an updatable priority queue structure on top of the RAG edges to keep track of the highest priority element. Such a structure is first constructed in linear time and every subsequent update incurs in a logarithmic time cost. When two regions R_1 and R_2 are merged into a new region R_{12}, one must update the RAG (and the associated priority queue). The edge connecting R_1 and R_2 must be removed, but let us also remark that all the edges adjacent to R_1 and

R_2 must also be eliminated from the RAG, since none of both regions exists anymore. We must then add the adjacency relations of the new region R_{12}. The computation is straightforward: the neighbors of the new region R_{12} are nothing but the union of the neighbors of the old R_1 and R_2 (with special care to remove any duplicates that may arise). The dissimilarity value associated to each of these edges must be computed and pushed to the priority queue. The complexity of the overall BPT construction process is $O(n \log(n) M)$, n being the initial number of nodes and M the maximum number of neighbors of a merged region during the construction. Given that typically $M \ll n$, the algorithm is quasilinear in practice. Recent advances have further accelerated the algorithm [1]. Among other improvements, the priority queue only stores the edge with the lowest dissimilarity value for each node. The other edges are brought back into consideration if the surrounding nodes are affected by a merge.

The final tree contains exactly $2n - 1$ nodes, which is a very space-efficient representation. Let us remark though that only a subset of all possible planar subdivisions is represented by the tree, which highlights the importance of research efforts to construct a good initial tree that conveys meaningful objects of the underlying image.

The key elements to define the behavior of a BPT are the *region model*, i.e. how regions are represented, and the *dissimilarity function*, i.e. the function to compare the region models, used to define the priority of the merges during tree construction. We review next the contributions related to these two elements.

Region model

The object-based nature of BPTs yields rich representations of the region that go beyond pixel spectra. Every BPT node can convey *regional* information, describing the region as a whole and not as a set of individual pixels. Examples of the regional data that can be associated to every node are the standard variation of the spectral signatures in the region, or shape features such as compactness.

To represent the spectrum of a region (and then compare it to the spectra of other regions) there are essentially two alternatives: parametric and non-parametric models. A parametric model makes assumptions about the homogeneity or Gaussian distribution inside the regions. A typical parametric model is to represent the spectrum of a region as the mean spectrum of its pixels. Non-parametric models, on the contrary, consist of per-band histograms of the pixel values, hence they represent the real observed distributions. In hyperspectral imagery, non-parametric models have a better performance since they can describe the internal variability of a region [191]. For example, a texture might

correspond to several peaks in the histogram. When averaging spectra in regions with high variability, one might end up representing the region with a "false" spectrum that is not present in any of the individual pixels. In addition to spectral data, the model usually stores the area of the region, since it is commonly used in the dissimilarity function.

Dissimilarity function

To establish a priority for merging during BPT construction, it is required to provide a means to compare models of two regions. A dissimilarity function $O(R_1, R_2)$ typically used for this purpose comprises two factors as follows:

$$O(R_1, R_2) = \min(|R_1|, |R_2|)^{\beta} D(R_1, R_2), \tag{3.1}$$

where $|R_i|$ denotes the area of region R_i. The first part of (3.1), $\min(|R_1|, |R_2|)^{\beta}$, is the so-called *area-weighting* factor. This is an agglomerative force intended to cluster regions that are very small compared to the rest of the elements in the RAG. When no area-weighting is used (i.e., $\beta = 0$), the resulting BPT might isolate small noisy areas and connect them to the rest only near the root of the tree. With moderate values of β, small regions are merged at some point, forcing the tree to better look like a hierarchical subdivision. When β is too large, the trees may become too biased toward being balanced, hampering their representation capabilities. Even though this parameter is barely discussed in the literature, being mostly set to $\beta = 0.5$ or $\beta = 1$, we must point out that it is an arbitrary parameter that has to be selected. In our experience, no area-weighting leads to poor representations (e.g., the root containing two children: one noisy pixel and all the rest of the image), while low values of β solve this issue without biasing the trees too much. Alternatively, Calderero and Marques [20] proposed to keep track of the out-of-scale regions and force their merging at some point, while Valero et al. [191] used a weighted sum of pixel values in a window to initialize the histograms, as a way of smoothing out outliers.

The second factor, $D(R_1, R_2)$, compares both regions based on their spectra. Kullback-Leiber divergence and Bhattacharyya distance are popular choices both in hyperspectral imagery and other types of images [20, 191]. Spectra are seen as probability distributions and compared using standard information theory concepts. Every bin of one histogram is compared against the corresponding bin of the other histogram. However, using cross-bin measures, which go beyond individual bins, has proven to be more robust [191]. The average of Earth Mover's Distances [156] among histograms of all bands has also been used as a robust and efficient cross-bin dissimilarity function [147]. Every distribution is

seen as a pile of dirt, and the difference between two distributions is seen as the amount of work required to turn one pile into the other one.

3.3 Proposed method

In the following, we first pose our problem as the minimization of an energy, based on probability distributions of spectrum and shape features. We then describe how we build an initial BPT and process it to find the best partition represented by this tree. Finally, we describe an optimization algorithm to modify the initial BPT, in order to extract a classification that minimizes our energy.

3.3.1 Energy formulation

Let $I = (I_j)_{1 \leqslant j \leqslant n}$ be an input image containing n pixels. We assume we are given a set of possible object classes, as well as priors for each class. Multi-label segmentation consists in an exhaustive partitioning of the pixels into a non-overlapping set of regions $\mathcal{R} = (R_i)$, together with associated class labels $\mathcal{L} = (L_i)$, where labels L_i belong to a set Ω of available labels. It can be stated as an optimization problem: minimize

$$E(\mathcal{R}, \mathcal{L}) = E_C(I, \mathcal{R}, \mathcal{L}) + \sum_{R_i \in \mathcal{R}} E_S(R_i, L_i), \tag{3.2}$$

where E_C expresses the color prior (quantifying how the segmentation fits the image spectrum), and E_S, the shape prior.

3.3.1.1 Color prior

For each object class, we suppose we are given training examples, from which the color distribution can be estimated and used as a prior. Given a candidate segmentation $(\mathcal{R}, \mathcal{L})$, the color prior is defined as follows:

$$E_C(I, \mathcal{R}, \mathcal{L}) = \sum_{R_i \in \mathcal{R}} \sum_{I_j \in R_i} -\log P(L_i | I_j). \tag{3.3}$$

One way of obtaining the posterior $P(L_i|I_j)$ is to train classifiers based on the samples' colors, using support vector machines (SVM), and to extend them to output probability estimates as usual in classification problems [206].

3.3.1.2 Low-complexity shape features in BPTs

Similarly, the shape prior term is defined as follows:

$$E_S(R_i, L_i) = -|R_i| \log P(L_i|\mathbf{S}_i), \tag{3.4}$$

$|R_i|$ being the area of region R_i, and $P(L_i|\mathbf{S}_i)$ being the probability of assigning the label L_i to the region R_i, given a vector $\mathbf{S}_i$ of shape features of that region. Common regularization (such as boundary length [143]) can be incorporated as part of this term. The weight on the area makes the per-pixel contribution of the color prior and the per-region contribution of the shape prior equally important.

We wish to enrich the nodes of BPTs by including shape information of the corresponding regions. Given that the optimization of the trees involves recomputing region descriptors, we must design a pool of features that can be computed efficiently from children nodes. In addition, the errors in estimating the shape descriptors in the finer levels should not be amplified in the upper ones.

Area can be efficiently computed by adding the areas of the children. **Rectangularity** and **elongatedness** shape descriptors have been used in the context of hierarchical methods [104, 183, 192], though no details on how to efficiently implement these features are provided. An oriented rectangle B of height h and width w is said to be the *minimum area enclosing rectangle* (MAR) of a region R if it is the rectangle of minimal area that entirely contains R. Rectangularity measures the resemblance to a rectangle, and is computed as $|R|/(h \cdot w)$. Elongatedness measures the resemblance to a line, and is defined as w/h.

We propose to store the **convex hull** of the region at every node. When two regions are merged, the convex hull of the new region can be computed by merging the convex hulls of its children. This can be done in linear time in the size of the input polygons by using *rotating calipers* [185]. The MAR can be efficiently computed after it [185]. Rectangularity, elongatedness and other descriptors like **solidity** [210] (filled fraction of the convex hull) are then directly derived. In a balanced tree, which is enforced by our construction function, convex hulls incur in an $O(n \log(n))$ increase of the storage required. Their computation does not increase the complexity of tree construction (proofs in Appendix A.2).

In a discrete environment, the errors of computing the region areas converge as the areas grow, so do the convex hulls. As a consequence, the aforementioned features tend to be more precise in the upper levels of the trees.

Another useful shape descriptor is **compactness** [141], related to the resemblance to

a circle. It is typically defined as $\delta R^2/(4\pi|R|)$, where δR is the perimeter of R. However, this formulation imposes difficulties in a discrete environment [141] due to the fact that the error in the estimation of δR does not converge. Li et al. [119] proved the robustness of computing compactness as $|R|^2/(2\pi I_g)$, I_g being the moment of inertia of the shape with respect to its centroid. Given that the centroid and moment of inertia of a region can be computed in constant time from the children, we propose this method to measure compactness in BPTs.

Let us assume that a probability density function $p(s|L)$ is available for every feature and class. These densities can be obtained by smoothing histograms of training samples [165]. Let us call $\mathbf{S} = s_1, ..., s_m$ a vector of shape features. Assuming features' conditional independence, we have:

$$P(L|\mathbf{S}) \propto \prod_{k=1}^{m} P(L|s_k) = \prod_{k=1}^{m} \frac{p(s_k|L)}{\sum_{L_j \in \mathcal{L}} p(s_k|L_j)}. \tag{3.5}$$

3.3.1.3 Total energy

Combining Eq. (3.2), (3.3) and (3.4), the energy criterion to minimize is formulated as:

$$E(\mathcal{R}, \mathcal{L}) = -\sum_{R_i \in \mathcal{R}}^{|\mathcal{R}|} \Big(\sum_{j \in R_i} \log P(L_i|I_j) + |R_i| \log P(L_i|\mathbf{S}_i) \Big) \tag{3.6}$$

3.3.2 Tree construction and processing

In this section we describe our proposed BPT construction approach, as well as the algorithm to extract the optimal classification from a BPT.

3.3.2.1 Supervised BPT construction

In Section 3.2, it was mentioned the region merging algorithm used to construct BPTs requires to define a *model* to represent the regions and a *dissimilarity* function to compare the regions. We choose a non-parametric model (i.e., spectral histograms) to represent the region, due to their well-known advantages (see Section 3.2). We have proposed, however, a modified dissimilarity function to construct the BPT [127].

In BPT-based classification methods (e.g., [191]), the dissimilarity measure used to construct the trees is purely based on the comparison of spectral histograms. In fact, commonly used dissimilarity functions (Eq. 3.1) always penalize the merging of dissimilar regions. Let us recall that non-parametric models were introduced to represent and

compare inhomogeneous regions, useful for textures and light gradients. However, internal class variability (e.g., an object composed by areas of different spectra) is not at all considered. In an unsupervised context, where there is no notion of object class, there is little hope to deal with this, since there is no reason to cluster dissimilar regions. However, when class probabilities are available we propose to include an additional force that clusters regions belonging to the same class, despite being spectrally dissimilar. The new function is as follows:

$$O(R_1, R_2) = \min(|R_1|, |R_2|)^{\beta}\Big[(1-\alpha)D(R_1, R_2) - \alpha \log P(L_{R_1}{=}L_{R_2})\Big]. \tag{3.7}$$

As in the original dissimilarity function (3.1), there is an area-weighting factor and an unsupervised term $D(R_1, R_2)$, which is computed by comparing spectral histograms of regions without any preliminary training. We here use the Earth's Mover Distance, due to its robustness to changes in illumination and its efficient computation [156, 147]. Equation 3.7 adds a *supervised* term $P(L_{R_1}{=}L_{R_2}|R_1, R_2)$, the probability of assigning the same label to both regions. This way, while the unsupervised term penalizes spectral dissimilarity, the supervised term will encourage merging regions that are likely to belong to the same class. The trade-off between both terms is controlled by parameter α.

The term $P(L_{R_1}{=}L_{R_2}|R_1, R_2)$ is computed by marginalizing over the classes as follows:

$$P(L_{R_1} = L_{R_2}|R_1, R_2) = \sum_{j=1}^{K} P(L_j|R_1)P(L_j|R_2), \tag{3.8}$$

where K denotes the number of classes and $P(L_j|R_k)$, with $k \in \{1, 2\}$, represents the probability of assigning a certain label L_j to segment R_k. We must now define a way to compute $P(L_j|R_k)$ based on the posteriors of the individual pixels contained in the region. One way to do this is to compute the probability of assigning the label to all pixels, conditioned by the fact that all labels are known to be equal inside the region:

$$P(L_j|R_k) = \prod_{\mathbf{x}_i \in R_k} P(L_j|\mathbf{x}_i) \Big/ \left[\sum_{\omega_m \in \Omega} \prod_{\mathbf{x}_i \in R_k} P(\omega_m|\mathbf{x}_i)\right]. \tag{3.9}$$

Alternatively, one can estimate $P(L_j|R_k)$ by averaging the individual pixel probabilities:

$$P(L_j|R_k) = \frac{1}{|R_k|} \sum_{\mathbf{x}_i \in R_k} P(L_j|\mathbf{x}_i). \tag{3.10}$$

While the first expression is closer to a strict Bayesian interpretation, we found the second one to be a simple yet useful approximation.

By introducing (3.7) we expect to better cluster semantically significant objects, according to the classifier's output.

3.3.2.2 Best segmentation represented by a given tree

The common processing on BPTs consists in selecting the highest or lowest branches satisfying a given condition [125, 191]. However, some contributions have formulated the problem in terms of energy minimization [157, 158]. In particular, Salembier et al. [157] have interpreted segmentation as a horizontal s-t *cut* on the tree (see Fig. 3.1), i.e., with a source at every leaf and a sink at the root. Let us denote τ a tree and $C(\tau)$ the energy of the cut on τ with minimal (3.6) among all possible cuts. Our task is to find such a minimal cut.

Considering that the branches in the tree are independent, the globally optimal cut can be found by a dynamic programming algorithm. Let us denote by

$$\mathcal{E}(R) = \min_{L \epsilon \Omega} E(\{R\}, \{L\})$$

the lowest possible energy of a region R (by assigning the label that incurs the lowest cost). The tree is traversed in a bottom-up manner. Whenever a region R is visited, the following property is evaluated:

$$\mathcal{E}(R) \leqslant C(R_{left}) + C(R_{right}), \tag{3.11}$$

where R_{left} and R_{right} are the children of R. If the property does not stand, we set $C(R) = C(R_{left}) + C(R_{right})$ and keep the best cuts of both children. Otherwise, we set $C(R) = \mathcal{E}(R)$ and replace the cuts by R with label L. This process is executed recursively until reaching the root of the tree. The overall algorithm is linear in the image size, since only one BPT traversal is required, and it guarantees the optimal cut in the space of solutions represented by the BPT.

3.3.3 Optimizing the trees

Even though the globally optimal cut on a BPT can be found efficiently, the organization of the nodes in the tree structure restricts the possible cuts that can be done on them. In Fig. 3.3 a toy example illustrates this issue. Let us suppose that an aerial shot of a city captures a house with a non-uniform roof. During the construction of the tree, a and b are merged together because they feature the lower dissimilarity among every pair of regions. Even by using our improved construction function (3.7), we cannot expect

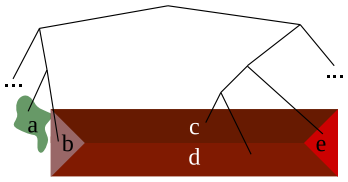


Figure 3.3: Faulty BPT: the object ($bcde$) is not represented in a single node, since a part of it (b) merged first to something else.

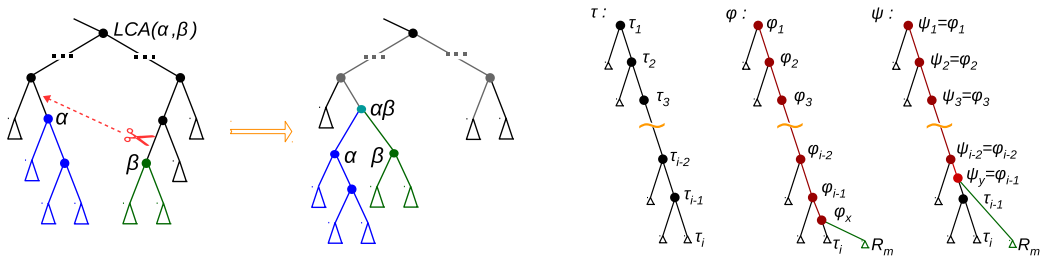
much better results, unless the classifier properly labels all regions. If we wish to include shape priors to enhance the results, let us point out that at the moment a and b were merged, it was impossible to know that b would eventually form a more significant object under a different sequence of merges, with the typical shape of a building. In other words, following this greedy construction approach we cannot predict the final shape of the objects in the output classification map. The resulting tree does not allow to perform any cut that would include the whole building into the same object, not even using strong shape priors, given that it is split through different branches. This is why we now propose to *optimize the tree itself.*

Our method consists in constructing an initial BPT with the usual region merging greedy algorithm, and then optimize it to extract a classification map that minimizes Eq. 3.6, thus incorporating the shape prior. To optimize BPTs we follow a local search approach, in which a solution is iteratively modified by performing local transformations on the trees, named *moves*.

3.3.3.1 Moves and associated updates

We propose a *prune-and-paste* move that prunes a branch of the tree and inserts it into another part of the tree. The pruned node must be pasted in a spatially adjacent location. Fig. 3.4(a) illustrates such a move: α is the paste place and β is the pruning place. We denote by $\mathrm{LCA}(\alpha, \beta)$ their lowest common ancestor in the tree. The move creates a new node $\alpha\beta$ in the paste side that comprises α and β. In the pruned side, the tree is collapsed after β is removed. In a balanced tree, which is encouraged by setting $\alpha > 0$ in (3.7), the number of possible moves is bounded by $O(n \log(n))$ (see proof in Appendix A). The neighborhood system is much richer than, for instance, Markov random fields (MRFs) on the pixel grid, considering that it comprises pairs of adjacent regions at several scales.

We store at each node R the branch cost $C(R)$ of the best possible cut within its branch. When applying a move as depicted in Fig. 3.4(a), it is necessary to recompute the branch cost C till the ancestry of α and β only. The rest of the branches are unaffected,



(a) *Prune-and-paste* move.

(b) Alternative paste places.

Figure 3.4: Family of moves to optimize BPTs.

as observed in (3.11). Among the ancestry, only the nodes below $\mathrm{LCA}(\alpha, \beta)$ require to recompute their models (shape and color features), given that further up the regions represented by the nodes do not change. Thus we recompute the features (and thus $\mathcal{E}(R)$ and $C(R)$) only in that part of the tree, and for the nodes in the tree above $\mathrm{LCA}(\alpha, \beta)$ we recompute only their branch cost $C(R)$, which simply involves reassessing (3.11) without reevaluating $\mathcal{E}(R)$ nor their features.

In a balanced tree there are at most $O(\log(n))$ ancestors of α and β, and, as stated before, for some of these ancestors the model of the regions must be updated. If we denote by K the complexity of updating a model (i.e., merging two children), the computation of the new costs C is $O(K \log(n))$. Usually $K \ll \log(n)$, therefore the time is $O(\log(n))$ in practice.

The adjacency relations in a BPT can be derived from the children by observing that the ancestors of two adjacent nodes are also adjacent, until the LCA. At the finest level the adjacency is known (e.g., a 4-connected grid).

3.3.3.2 Properties of the moves

We now explore some properties of the *prune-and-paste* move. In particular, we will show that finding all possible energy decreasing moves does not require to exhaustively evaluate the energy gain of every possible move. Proofs are available in Appendix A.

Let us consider the situation of Fig. 3.4(b). In the second tree, a node was pasted at the position τ_i. We wish to compare the effect of pasting higher instead (as in the third tree). Let us denote by $\tau_i < \tau_j$ the *is-a-descendant-of* relation.

Proposition 1. *Given a tree τ, suppose a node R_m is pasted at $\tau_i < \tau_1$ leading to a new tree φ. Let us consider an alternative move that pastes R_m at τ_j, with $\tau_i < \tau_j < \tau_1$, producing a tree ψ. In the cases where either $C(\varphi_1) - C(\tau_1) \leqslant 0$ or $C(R_m) \geqslant C(\varphi_1) - C(\tau_1)$, then $C(\psi_1) \geqslant C(\varphi_1)$.*

The proposition states that, if a move reduces the energy in the branch, a higher paste place will not do better. Under certain assumptions, if the move increases the energy, pasting higher will also increase it. Intuitively, pasting lower is more general.

Proposition 2. *Let us consider a case where Prop. 1 hypotheses do not apply. There might then exist a higher paste place τ_α so that $C(\psi_1) < C(\varphi_1)$. Let us suppose that instead of pasting at τ_α we paste at τ_β, with $\tau_\alpha < \tau_\beta < \tau_1$, leading to a tree ρ. Then $C(\rho_1)$ would monotonously decrease as the paste place τ_β is located higher.*

When the hypotheses of Prop. 1 do not apply, there might exist a favorable paste place higher in the branch. However, we know that the higher it is, the most beneficial it can be. As a result, we can just consider the highest possible paste place (right below the LCA). However, any paste place between the location of the original cut and the LCA would lead to the same energy. These cases happen when it is preferable to cut the pruned node apart. The higher we paste R_m, the less we condition the way the rest of the tree must be cut.

Following these properties, an exhaustive search of energy decreasing moves can be achieved as follows: for every possible pruning place we check the energy gain of the moves for only the lowest paste places. If the move is beneficial, we keep that move as a candidate. If the move is not beneficial we can discard it as well as all the paste places in the branch (but we do this only after performing one additional check with the paste place at some point between the cut and the LCA).

3.3.3.3 Optimization algorithm

We propose the following optimization scheme, which must be iterated:

1. Construct a heap of all moves according to the branch cost variation ΔC.
2. (a) Either apply the best move, or
   (b) apply the best k moves (when still appropriate).

The first step involves exploring the whole search space, featuring an $O(n \log^2(n))$ complexity. Moves are simulated to measure the energy gain but are not applied. Propositions (1-2) can be used to reduce the execution time of this step, and the energy gain can be evaluated in parallel.

In the second step, energy decreasing moves are applied. As soon as a move is applied, the tree is restructured and the effect of some other moves may be affected. New energy

decreasing moves may also arise. 2a) just applies the best move and reiterates. As a shortcut, we can just update the entries in the heap of the moves that may have been affected. This option is still costly because ΔC must be recomputed for any move that could have been possibly affected, even though in practice this might be the case for just a few of them. This approach guarantees to apply the best move each time. Considering the fact that there might be many unrelated energy decreasing moves in the tree, 2b) proposes to apply a number k of best moves, but verifying for each move that ΔC did not increase as a result of the previous transformations done on the tree. This second approach will apply a number of independent moves first, ignoring the fact that some new energy decreasing moves might arise, which will be dealt with in the next iteration. The loop stops when there are no more moves whose ΔC is negative.

The lowest scale both on the pruning and paste side can be set to the pixel level. A coarser scale on the pruning side can be used to adjust the precision of the moves. If a coarser scale is set on the paste side, we limit the minimum object size of the partitions, since we ignore moves that lower the cut below a certain level. This can be adjusted by observing the density functions of the area feature.

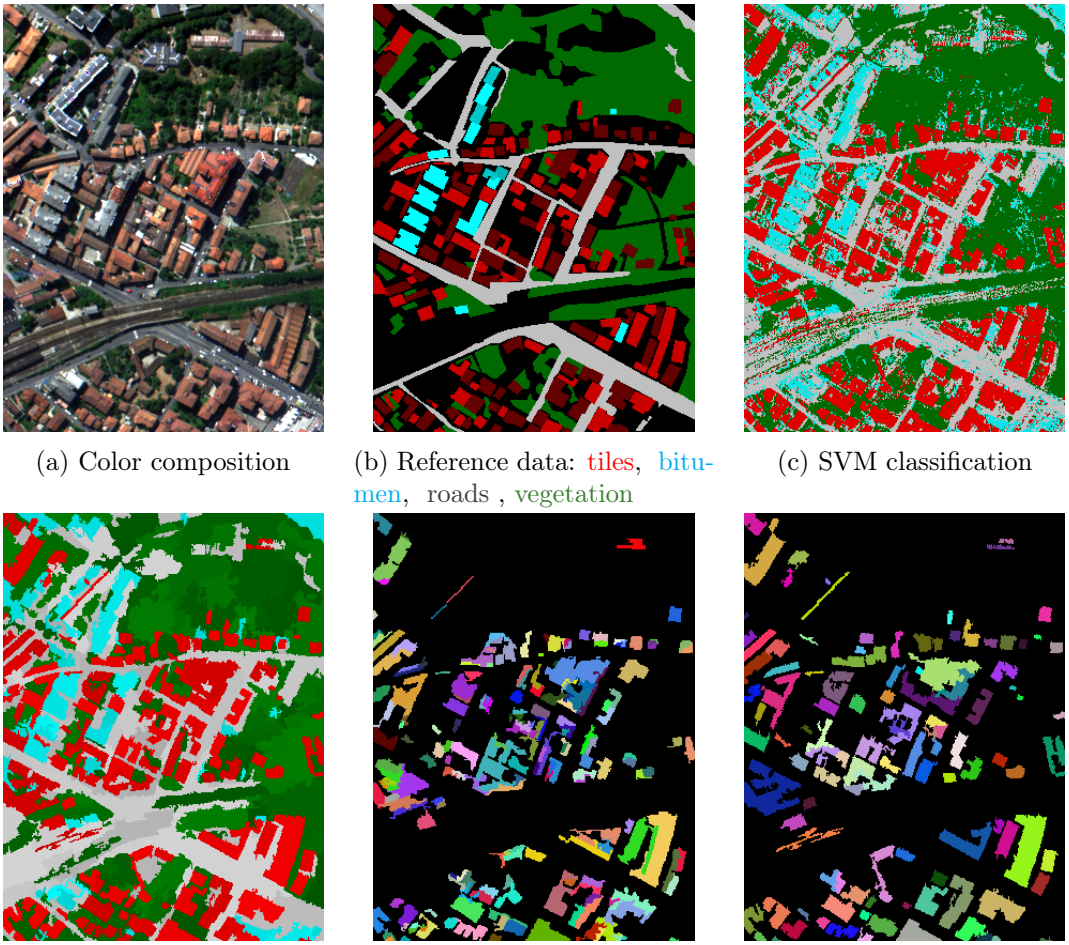
3.4 Experiments

We first validate our supervised BPT construction approach. We do this on hyperspectral images, since they contain a rich and discriminant spectrum, and evaluate whether our function (3.7) indeed clusters regions together based on the classifier's probabilities. However, such classification power based solely on pixel's values is notoriously degraded in other types of images. We therefore introduce shape features in order to enhance classification on different types of inputs, including urban aerial remote sensing images.

3.4.1 Supervised BPT construction

We perform experiments on the *Pavia Center* hyperspectral dataset, acquired with the Reflective Optics System Imaging Spectrometer (ROSIS-03). The image has spatial dimensions 400×300 and contains 102 bands, covering a range from 0.43 to 0.86 μm and 1.3 m spatial resolution. A color composition of the image is shown in Figure 3.5(a).

A reference image that labels entire objects was built, including four classes (see Figure 3.5-b). This reference was constructed by combining the labeling of isolated pixels provided with the original image, visual inspection and publicly available official Italian records of building boundaries. Since the boundaries of buildings are well defined,



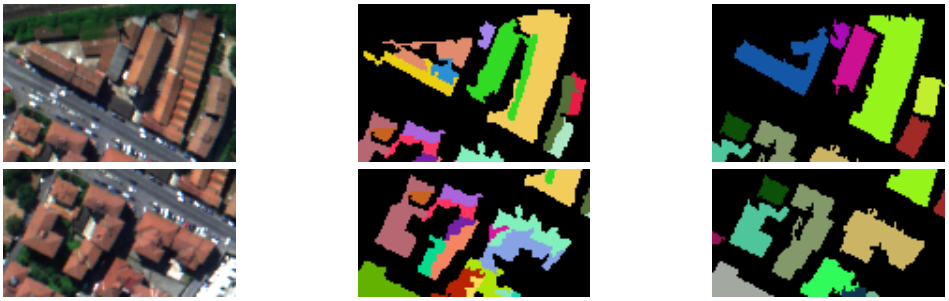
(a) Color composition (b) Reference data: tiles, bitumen, roads , vegetation (c) SVM classification

(d) BPT classification ($\alpha = 0.5$) (e) Tile building detection (BPT with $\alpha = 0$) (f) Tile building detection (BPT with $\alpha = 0.5$)

Figure 3.5: Experiments on *Pavia Center* hyperspectral image.

there is a particular interest in analyzing the performance of BPTs to extract buildings.

An SVM with a Gaussian radial basis function kernel is first trained on 100 randomly selected samples of each class. The SVM parameters are set by 5-fold cross validation ($C = 128$, $\gamma = 2^{-5}$). The SVM classification is shown in Figure 3.5(c). A BPT is then constructed on top of the SVM probabilities. We use a non-parametric model with 30 histogram bins per band and mild area-weighting ($\beta = 0.1$). Two variants were tested: **a)** totally unsupervised construction, i.e., setting $\alpha = 0$ in (3.7), which is equivalent to the previous function (3.1); **b)** supervised construction with equal contribution from both terms in (3.7), i.e., $\alpha = 0.5$. Instead of using complex shape features, now we simply



(a) Color composition (b) BPT with $\alpha = 0$ (c) BPT with $\alpha = 0.5$

Figure 3.6: Unsupervised (b) versus supervised (c) BPT construction. In the supervised case, regions are better clustered together to represent significant objects.

Table 3.1: Numerical evaluation on the *Pavia Center* dataset.

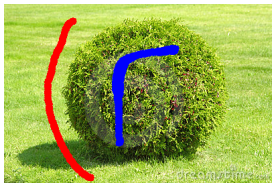
	SVM	Graph cut	$\text{BPT}_{\alpha=0}$	$\text{BPT}_{\alpha=0.5}$
Building overlap	0.51	0.51	0.54	**0.56**
Overall accuracy	0.88	**0.94**	0.91	**0.94**

regularize the cut by the total number of regions:

$$E(\mathcal{R}, \mathcal{L}) = \lambda ||\mathcal{R}|| - \sum_{R_i \in \mathcal{R}} \sum_{I_j \in R_i} \log P(L_i | I_j), \tag{3.12}$$

where λ controls the coarseness of the output. We experimentally set $\lambda = 40$ to optimize the accuracy. The resulting classification map with supervised tree construction ($\alpha = 0.5$) is shown in Figure 3.5(d). We also executed a graph cut with α-expansion [19] on the probabilities derived from the SVM, which proved to be effective in the past for hyperspectral image classification [182]. Its regularity parameter was also set empirically to optimize the accuracy.

Figures 3.5(e-f) and the close-ups of Figure 3.6 compare the results obtained by applying the unsupervised and supervised approaches for BPT construction. These figures isolate the *tile* objects from the rest and assign a random color to every individual object. From these illustrations we can clearly appreciate that including class probabilities during BPT construction yields an improved tree that better clusters the objects together. To validate this numerically we compute the overlap between every building (belonging either to *tiles* or *bitumen* classes) in the reference data and the most overlapping building region in the BPT output. The overlap is measured with Dice's coefficient [42] defined as: $2|R_1 \cap R_2|/(|R_1| + |R_2|)$. The resulting overlap coefficients are averaged over all reference buildings to estimate how the BPT output matches the reference data from an



(a) Input (400 × 267)

(b) Gorelick et al.

(c) BPT Opt

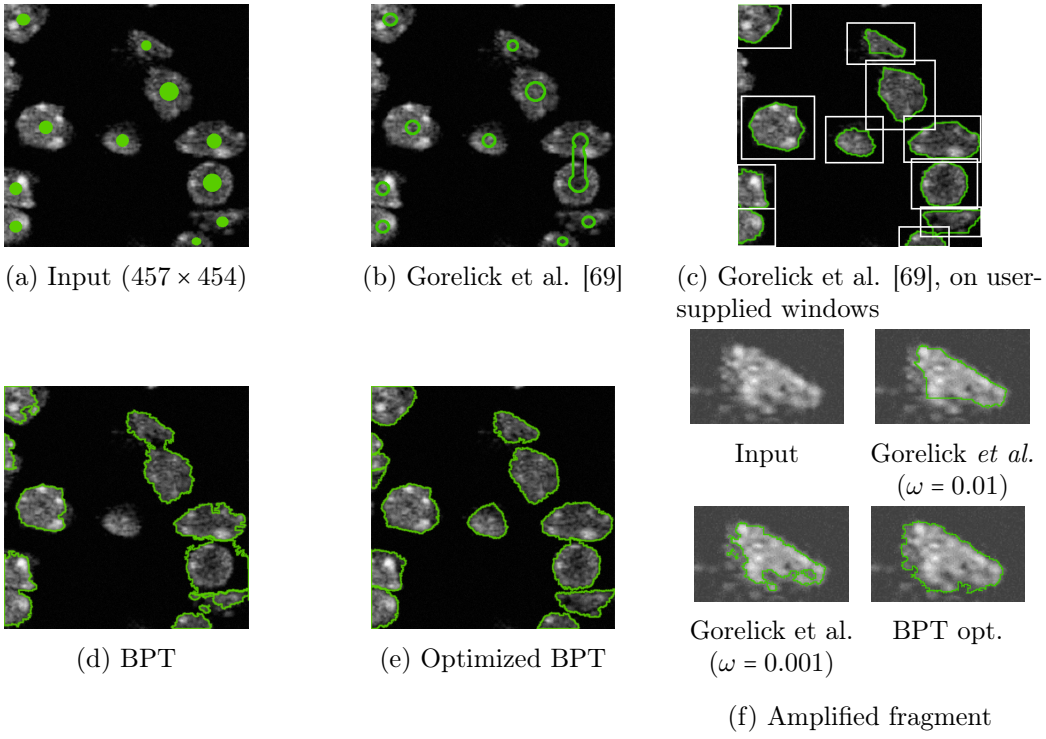
Figure 3.7: Convex object. Method [69] and optimized BPTs.

object-based perspective. The numerical results, together with the overall accuracy, are summarized in Table 3.1, which also includes the values for SVM and graph cut. A first observation we can make is that $BPT_{\alpha=0.5}$ performs better than $BPT_{\alpha=0}$, corroborating the visual result from Figs. 3.5(e-f). Secondly, while graph cut is known to improve the SVM classification, we can verify that while this is true from a pixelwise perspective (in terms of overall accuracy), it is not from an object-based perspective (in terms of building overlap). Finally, the use of $BPT_{\alpha=0.5}$ outperforms the other methods in terms of object overlap. This validates the idea of including class probabilities during tree construction.

3.4.2 Classification with shape priors

In a first series of experiments we constrain our method to binary segmentation with convexity shape prior, in order to compare it with a recent state-of-the-art technique designed for this purpose. In a second series of experiments we move to multi-object multi-class segmentation in remote sensing imagery and compare the behavior of our algorithm against the common approaches in the field.

An algorithm for a soft convexity shape prior in image segmentation was introduced by Gorelick et al. [69]. We will first show that we can perform similarly in a single convex object, while our technique is not specifically designed for this prior. A natural image extracted from [69] and the markers used are shown in Fig. 3.7(a). We set ω = 1, a contrast sensitive Potts model (weight 10^{-4}) and an 8×8 orientation stencil, the parameters required by [69]. We found this parameter setting to yield the most visually pleasant result. We ran our BPT optimization approach with a data term learned from markers' histograms and with density functions favoring solidity and compactness for the foreground class, which convey the notion of convexity. For BPT construction we set $\alpha = 0.5$ throughout all the experiments, to make the terms in (3.7) equally relevant. We sample moves that involve regions of at least ten pixels. In this case we apply all good moves in the queue at every iteration, which are in practice less than 10. We did not impose hard constraints on the marker locations. The resulting segmentations are



(a) Input (457×454) (b) Gorelick et al. [69] (c) Gorelick et al. [69], on user-supplied windows

(d) BPT (e) Optimized BPT (f) Amplified fragment

Figure 3.8: Multiple convex objects (cell nuclei). Gorelick et al. [69] and BPT optimization.

depicted in Figs. 3.7(b-c). Our BPT optimization approach achieves similar segmentation performance.

Fig. 3.8(a) shows a slice of a laser scanning microscopy image of brain tissue, where we wish to identify cell nuclei (green markers). If we apply the method by Gorelick et al. [69] to the whole image with all the markers, the result is inaccurate because the technique is not designed to segment more than one object at once (Fig. 3.8-b). In Fig. 3.8(c) we overlap the result of applying [69] to different fragments of the image that include each individual object (parameter ω in [69] is set to 0.01). The technique individually outlines each of the nuclei in the absence of the rest, but prior knowledge about their location is required.

We constructed a BPT on this image (α in Eq. 3.7 is set to 0.5 in all the experiments). As expected, the best-cut segmentation without optimizing the tree (Fig. 3.8-d) is not competitive, because many tree nodes do not satisfactorily represent objects. We ran our BPT optimization approach with a data term learned from markers' histograms and with density functions favoring solidity and compactness for the foreground class, which

convey the notion of convexity. We sample moves that involve regions of at least ten pixels. In this case we apply all energy decreasing moves in the queue at every iteration, which are in practice less than 10. We did not impose hard constraints on the marker locations. After optimizing the BPT with the method proposed in Sec. 3.3.3, each object is segmented individually and adjacent objects are delineated separately (Fig. 3.8-e). Our approach also produces accurate boundaries despite the low foreground/background contrast. The method by Gorelick et al. tends to either oversmooth the boundary to enforce convexity, or produce a very non-convex object, depending on the parameter ω (see the amplified nucleus in Fig. 3.8-f).

In the context of multi-object multi-class segmentation, we tested our method on images of urban scenes extracted from Google Maps screenshots. Figs. 3.9(a) and 3.10(a) show two color images acquired over New York City and the area of Brest, respectively. For both images, the manual segmentation was based on visual inspection combined with cadastral records available online. The list of the considered object classes is given in Fig. 3.9(b) (no instance of the *internal road* class is present in the Brest image). In the particular case of buildings, cadastral information was used to delineate every object independently even when they are spatially adjacent.

To evaluate the performance of the proposed method, we use two criteria:

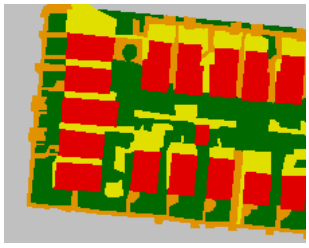
1) Overall accuracy $\mathcal{A}$, defined as the proportion of correctly classified pixels.

2) Building's overlap $\mathcal{D}$. For every building in the manually segmented image, we search for the most overlapping *building* region in the segmentation map in terms of Dice's coefficient [42]. Criterion $\mathcal{D}$ is estimated by averaging the computed coefficients.

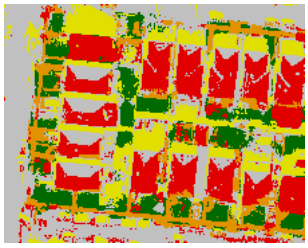
As in Section 3.4.1, an SVM with a Gaussian radial basis function kernel was used for the data term, tuned by tenfold cross-validation. The criterion (3.4) involved area, rectangularity and elongatedness shape descriptors. The distributions were trained by kernel density estimation on a set of sample objects from an adjacent image. In the area covered by these objects, 100 random pixels per class were selected to train the SVM.

We compared the performance of the proposed approach with the following methods: 1) SVM; 2) graph cut with α-expansion [19] (GC); 3) cut on the BPT, regularized by the number of regions without using shape priors (TC) [157]; 4) cut on the same BPT with our shape formulation (3.6), but without tree optimization (TSC). Figs. 3.9(c–f) illustrate the output of these techniques for the image of New York. Figs. 3.9(g–h) depict the results obtained by applying our optimization method with different values of k (see Sec. 3.3.3.3). The SVM classification exhibits many issues, notably the assignment of some roof parts to the wrong class. GC and TC smooth the results though do not correct the main mistakes in the classification. In the initial cut with shape priors on the unoptimized

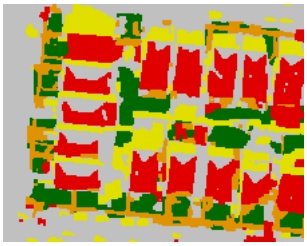
(a) Input (225×180)



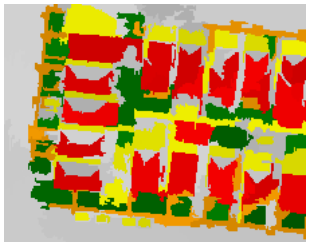
(b) Reference: tiles, roads, internal roads , veget., shadow



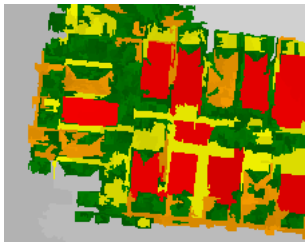
(c) SVM
($\mathcal{A} = 0.64$, $\mathcal{D} = 0.8$)



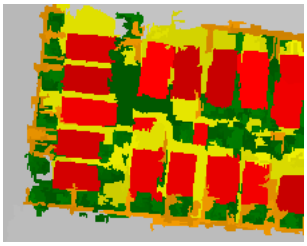
(d) GC
($\mathcal{A} = 0.68$, $\mathcal{D} = 0.81$)



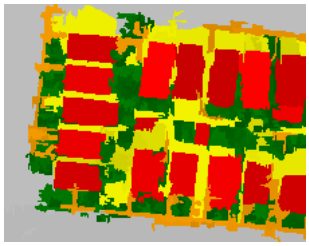
(e) TC
($\mathcal{A} = 0.71$, $\mathcal{D} = 0.71$)



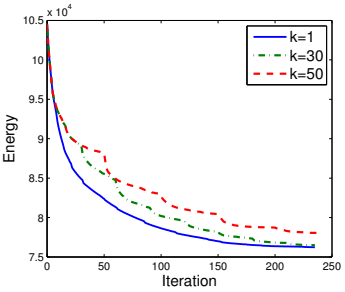
(f) TSC
($\mathcal{A} = 0.65$, $\mathcal{D} = 0.50$)



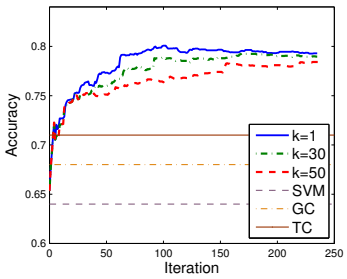
(g) BPT opt., $k = 1$
($\mathcal{A} = 0.79$, $\mathcal{D} = 0.89$)



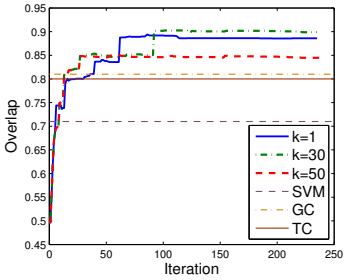
(h) BPT opt., $k = 30$
($\mathcal{A} = 0.79$, $\mathcal{D} = 0.90$)



(i) Evolution of the energy



(j) Evolution of overall accuracy



(k) Evolution of building's overlap

k	Time[s]	Time[s] (Props. 1-2)
1	707	334
30	23.08	9.02
50	12.47	4.69

(l) Execution time

Figure 3.9: Experimental results for the image over New York City.

tree (TSC), some regions are enhanced but some others are significantly deteriorated with respect to the previous methods. This is due to the faulty tree construction that does not represent the entire objects in unique nodes. Figs. 3.9(g–h) show that the optimization of the tree copes with these issues, not only enhancing the initial cut on the tree but also outperforming the other techniques.

The evolution of the energy (3.6), the accuracy $\mathcal{A}$ and the building's overlap $\mathcal{D}$ with respect to the number of iterations are depicted in Figs. 3.9(i–k). As expected, the energy curve becomes less smooth as k increases. For small enough values of k, the segmentation maps are almost identical. This validates the fact that many branch moves are independent and can be applied prior to reconstructing the heap.

The BPT for this image is constructed in 1.25 seconds on an 8-CPU 2.7 GHz processor. The optimization time, summarized in Fig. 3.9(l), is considerably faster when Propos. 1-2 are used.

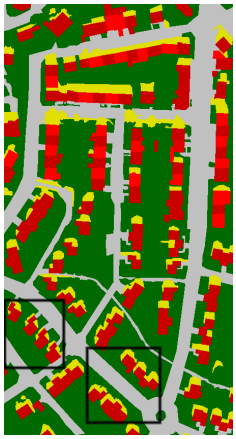
Fig. 3.10 illustrates experimental results for the image of Brest. Our method was executed with $k = 600$. Fig. 3.10(c) shows the obtained segmentation map. Two fragments of the map (boxed in Fig. 3.10(b)) are amplified for comparison in Fig. 3.10(d). These results validate the previous observations. The BPT is built in 13 seconds and the optimization takes 84 seconds using Propositions. 1-2 (see Section 3.3.3.2), against 214 s.

We highlight in another portion of image over Brest (Fig. 3.11) that our method is able to separate an entire building blob into rectangles. In addition, the use of shape features permits to "switch" a small building to the correct class, even though its color was assigned to *road* by the classifier.

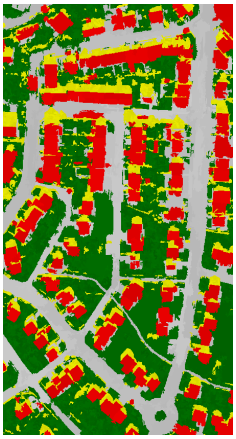
3.5 Concluding remarks

I have presented a hierarchical model for the multi-class multi-object classification of images with shape priors. The problem is formulated as an energy minimization task, using the learned probability distributions of spectrum and shape features. We construct a BPT of the image and search for the horizontal cut on the tree which minimizes the proposed energy function. One of the key innovations of the proposed method is that instead of using the standard BPTs, we optimize them by pruning and regrafting their branches at different scales in order to minimize the best classification that can be extracted from the tree. BPTs are a good departure point for optimization, since they allow to perform moves of variable region sizes. Conversely, in an MRF approach defined on the pixel grid, each time we perform a move we would have to search for a relevant region to perform a label switch (e.g., by using minimum spanning forests). A BPT

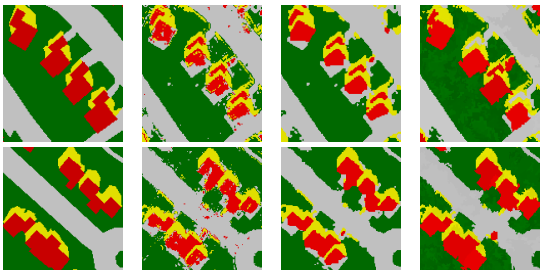
(a) Color image



(b) Manual classification



(c) BPT optimization



Manual SVM GC BPT opt.

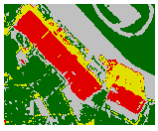
(d) Comparison on close-ups of the image

	$\mathcal{A}$	$\mathcal{D}$
SVM	0.764	0.50
GC	0.799	0.49
TC	0.804	0.47
BPT opt.	**0.813**	**0.69**

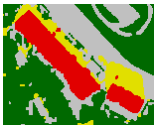
(e) Performance

Figure 3.10: Experimental results for the image over Brest.

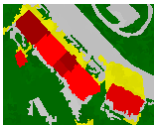
(a) Color image.



(b) SVM



(c) Graph cut



(d) BPT opt.

Figure 3.11: BPT optimization detects individual buildings.

can in fact be seen as a way of dynamically storing a set of candidate adjacent regions at different scales. Our theoretical study permits to reduce the space of branch moves. We also proposed an improved BPT construction function that adds a force to cluster together regions that are similar in terms of the classifier's output.

The experiments show that the method effectively incorporates shape information into classification, even when multiple classes and objects are present in the image, thus gaining a competitive edge with respect to recent literature in the field. The main limitation of the method concerns its scalability, both in terms of potential for parallelisation and of the choice of features. We will address how to handle these limitations later in this manuscript.

In the next chapter, I will present another graph-based model, which aims at incorporating a shape prior for segmentation of time series of noisy images. This time, we focus on segmenting shapes which only grow or shrink in time, and our model is designed to suit this growth or shrinkage constraint, respectively.

Chapter 4

Video Segmentation with Shape Growth or Shrinkage Constraint

Automatic segmentation of objects in videos is a difficult endeavor in computer vision [116]. This task becomes even more challenging for image sequences with low signal-to-noise ratio or low contrast between intensities of spatially adjacent objects in the image scene. Such challenging data are recorded frequently, for instance, in satellite remote sensing or medical imaging.

In this chapter, we aim at segmenting objects with shapes which can only grow or shrink in time, from sequences of extremely noisy images. Examples of growing shapes are forest fires or expanding cities in satellite images and organ development in medical imaging. In the image sequences we consider, both foreground and background intensity distributions can vary significantly over time: foreground can be heavily occluded or undistinguishable from a part of the background, and data for some pixels can be missing (see Fig. 4.2 and 4.7 for examples of such sequences). Most of previously-proposed spatio-temporal methods rely on coherence of foreground/background intensity distributions in successive image frames, and are therefore not suited for segmenting such noisy data sets. Few approaches have been specifically designed for spatio-temporal segmentation of magnetic resonance image (MRI) sequences with low signal-to-noise ratio [153, 205]. Applied to multi-temporal time series that show a monotonously growing or shrinking structure, however, these smoothing methods bias results towards the mean shape obtained from averaging consecutive segmentations and, hence, underestimate rapid growth or shrinkage events.

To address this issue, we have proposed a new omniscient segmentation framework based on graph cuts for the joint segmentation of a multi-temporal image sequence. It

introduces growth or shrinkage constraint in graph cuts by using directed infinite links, which connect pixels at the same spatial locations in successive image frames. By minimizing a submodular energy computed on the resulting spatio-temporal graph of the image sequence, the proposed method yields a *globally optimal solution.* Differently from the state-of-the-art omniscient techniques, it does not rely on the coherence of the intensity in time, but only on the coherence of the shape.

We have validated the performance of the proposed framework on three applications with very noisy image sequences. The first one deals with the segmentation of multiyear sea ice floes in a set of satellite images acquired through different satellite sensors. The new method returns accurate melting profiles of sea ice, which is important for building climate models. The second application segments growing burned areas from time series of optical satellite images with missing data. The third application addresses the segmentation of brain tumors from longitudinal sets of multimodal MRI volumes, where we impose additional inter-modal inclusion constraints for the joint segmentation of different image structures (brain tissues).

4.1 Related work

Segmentation of time series

Image segmentation methods applied independently to each frame [204, 25] produce unstable results, while temporal coherence in video sequences yields a lot of information not available for a single image. There are two main categories of approaches for the spatio-temporal segmentation of image sequences. *Causal*, or *feedforward*, techniques consider only past data for segmenting each next frame [148, 179]. *Omniscient* approaches take advantage of both past and future data by analyzing the video as a 2D+T = 3D spatio-temporal pixel volume [164, 40, 73]. In this case, the segmentation of the entire image set supports each of the individual segmentations.

Graph-based methods gained popularity among omniscient approaches, in particular those using hierarchical model [73], normalized cuts [164] or graph cuts [205]. These techniques do not impose any shape prior knowledge. It was proven that introducing shape priors into image segmentation, *i.e.* favoring segmentations similar in some sense to a given shape, allows to drastically improve segmentation of objects in the presence of strong noise and occlusions [32, 154]. However, imposing shape priors increases significantly both algorithmic and computational complexity of segmentation algorithms [50]. Schoenemann and Cremers proposed to compute minimal ratio cycles in a large product

graph spanned by the image and the shape template for finding globally optimal segmentations, which are consistent both with edge information and with a shape prior [160, 161]. In order to speed up the algorithm, they implemented it in parallel on graphics hardware. This algorithm can be applied for segmenting shapes in time series in a causal way, so that the template determined for the last frame is matched to the next. However, it does not guarantee globally optimal solution over the whole temporal sequence.

Globally optimal segmentation

Extensions of graph cuts to multi-class segmentation have been proposed but generally do no guarantee optimal solutions, except *e.g.* in the case of Ishikawa's construction [84], which requires labels to be ordered and the interaction term to be a convex function of their differences. This graph construction makes intensive use of infinite links to constrain the min-cut solutions to satisfy desired properties required to interpret them as image segmentation solutions. This was the source of inspiration for our work.

A study related to shape constraints can be found in [39], where one image has to be segmented in several possibly-overlapping objects. Infinite links are used for imposing common boundaries, inclusion or exclusion conditions between objects in a same single image. A similar approach [118] segments jointly two surfaces in a same volumic image, under the constraint that they should be separated by a given minimal margin. There is however no work related to shape growth or shrinkage in time series.

Wolz et al. [205] applied graph cuts for simultaneous segmentation of serially acquired MRI volumes. They defined temporal edge weights as the intensity differences of voxels at the same spatial locations. The same smoothness constraint was applied both in space and time, and the segmentations at different timepoints were forced to be consistent in areas where a small intensity difference between the images exist. This type of temporal constraint is suboptimal in image series where intensity distributions of foreground and background vary significantly over time. To the best of our knowledge, our work is the first to use infinite links to enforce a temporal growth constraint, and we illustrate in Sec. 4.3 the advantage of the new method over previous approaches such as [205].

4.2 Proposed method

In the following, I first recall a formulation of the graph cut, and then present the proposed method. Graph cut is an optimization tool coming from graph theory, based on the rewriting of image segmentation problems as (s,t)-min-cuts in graphs, on the

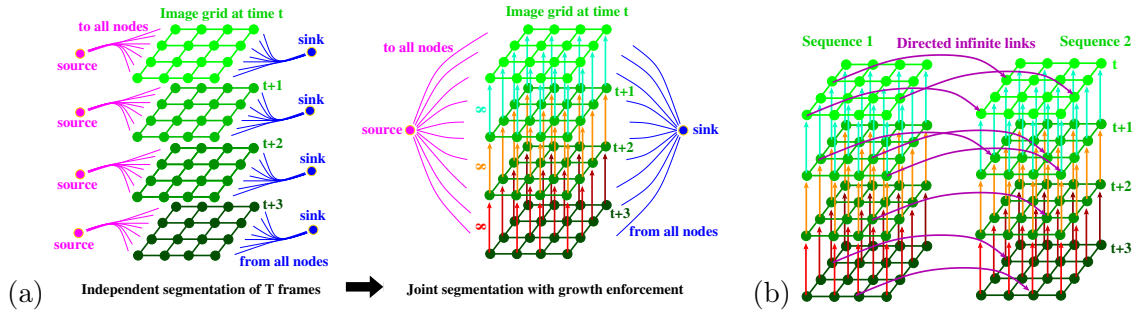


Figure 4.1: (a) Enforcing shape growth in an image sequence. (b) Segmenting jointly two sequences S1 and S2, by enforcing the foreground of S1 to contain the foreground of S2, with directed infinite links from S1 towards S2, between all pixels of coordinates (x, y, t, s_1) and (x, y, t, s_2).

equivalence of (s,t)-min-cut and max-flow problems, and on the existence of efficient algorithms to solve the latter ones [18, 35, 65]. In practice in computer vision, it can be used to find the globally optimal binary segmentation of images where the segmentation criterion is related to a Markov Random Field with submodular interaction terms, *i.e.* a criterion E of the form:

$$E(L) = \sum_{\text{pixels } i} V_i(L_i) + \sum_{i \sim j} W_{i,j}(L_i, L_j), \tag{4.1}$$

where L is the binary labelling function to be found (L_i is the label of pixel i), individual potentials V_i are any binary real-valued functions measuring the disagreement between a prior probabilistic model and the observed data, $i \sim j$ denotes a pair of neighboring pixels (any neighborhood system can be used), and $W_{i,j}$ are any real-valued interaction terms between neighboring pixels expressing spatial coherency of labels, satisfying

$$W_{i,j}(0,0) + W_{i,j}(1,1) \leqslant W_{i,j}(0,1) + W_{i,j}(1,0). \tag{4.2}$$

A directed infinite link between two pixels expresses precisely the constraint that this pair of pixels cannot have the pair of labels (0,1), by assigning an infinite cost to such an interaction. The remaining possible pairs of labels are thus (0,0), (1,1) and (1,0), which means that either both pixels have the same label, or the order of labels is predefined (1 for first pixel and 0 for the second one). In the case of image binary segmentation, if 0 stands for the background and 1 for the foreground object, then this means that the second pixel may belong to the foreground only if the first one already does.

4.2.1 Growth/shrinkage constraint

Given a sequence of images $I(t)$ preliminarily aligned, shape growth can be easily expressed as the property that the foreground object cannot lose any pixel when time advances. Otherwise said, if a pixel belongs to the foreground object at time t_1, then it belongs also to the foreground object for all times $t_2 > t_1$. Equivalently, and simpler: a pair of pixels $((x,y,t),(x,y,t+1))$, sharing the same location and immediately successive in time, cannot have the pair of labels (1,0), with the same binary segmentation notations as above. This can be enforced by setting monodirectional infinite links from all pixels to their immediate predecessor in time.

Given T images $I(t)$, with $t \in [1,T]$, and as many associated submodular segmentation criteria E^t, we transform the problem of segmenting independently each image $I(t)$ according to its criterion E^t, into a joint segmentation of all images together, by enforcing the shape growth constraint with directed infinite links (see Fig. 4.1(a)). Thus, instead of applying graph cut T times independently to planar grids of the size of the images $W \times H$, we apply graph cut once to a 3D grid $W \times H \times T$, consisting of the same nodes and edges, but with additional monodirectional infinite links in time. The criterion to be minimized is then $E = \sum_t E^t$ under the constraint of shape growth:

$$E(L) = \sum_{\text{pixels } i} V_i(L_i) + \sum_{i \sim j} W_{i,j}(L_i, L_j) + \infty \sum_t \delta_{L_i^t > L_i^{t+1}}. \tag{4.3}$$

Since the problem is binary and submodular, the solution found by graph cut is globally optimal.

Manifestly, one can enforce shape shrinkage instead of shape growth, by reversing the direction of the infinite links. Another straightforward extension, needed in Sec. 4.3.3, consists in applying this approach to the case of sequences of 3D images. The directed infinite links are then set for all pairs of voxels of the form $((x,y,z,t),(x,y,z,t-1))$ to enforce 3D shape growth.

In some applications, it may happen that growth (or shrinkage) is only very probable, but not with probability 1, *i.e.* growth should be considered as a probable hint but should not be enforced strictly at all locations at all times. In that case, one may replace directed infinite links by directed finite links: the weights of these links will encourage growth (more or less strongly depending on the weight), but sufficiently disagreeing potentials V_i may make the shape locally shrink instead. Thus, shrinkage would be discouraged but not forbidden.

4.2.2 Inter-sequences inclusion constraint

It is also possible to segment jointly several image sequences $I(s)(t)$ with the constraint that the foreground object in some sequences should be included in the foreground object of some other sequences. This can be done similarly by considering together the graphs associated to all sequences, and, for each inclusion constraint, by adding directed infinite links between pixels of the desired sequences s_1 and s_2, sharing same location and time: such links from (s_1, x, y, t) to (s_2, x, y, t) for all x, y, t will force the foreground object in sequence s_1 to contain the one of sequence s_2 (see Fig. 4.1(b)). An example of such application is given in Sec. 4.3.3, where image sequences correspond to four different MRI modalities, aligned both in space and time.

Naturally, instead of imposing an inclusion constraint over the whole time span and the whole image space, it is possible to specify spatio-temporal domains of constraints, for instance to express that the inclusion property between two sequences has to be satisfied inside a pre-defined region and/or during a pre-defined time span $[t_1, t_2]$ only, by adding directed infinite links in these sets only.

4.2.3 Weighting frames by reliability

As presented earlier, enforcing shape growth explicitly is particularly important when facing noisy sequences of images, as a joint segmentation incorporates information from different frames. With our approach, a fully noisy frame in the middle of several good-quality frames will be automatically ignored in practice, because statistically the (bad) potentials V_i at any pixel i in that frame will be neglectable with respect to all other (correct) potentials at the same pixel in other frames. Hence, the solution we bring will be strongly robust to any kind of noise, provided that the noise is not coherent in time during long time spans.

The quantity of noise that the method can handle depends on the number of neighboring frames with good-quality information at the same spatial location. One way to increase even more the trade-off between admissible level of noise and quantity of good information consists in estimating the reliability of the information given, in each image or even at each pixel, and in weighting accordingly the associated energies : $E = \sum_t w_t E^t$. For instance, if a strong level of noise is detected globally in one image $I(t)$, one may multiply the corresponding energy E^t by a small reliability factor $w_t < 1$, in order to make it less influential than other frames. Such a reliability factor could be computed for instance from preliminary image processing steps (for example, correlation between an image $I(t)$ and its neighboring ones $\{I(t') \; ; \; 0 < |t - t'| \leqslant \delta t\}$).

4.2.4 Complexity

The precise theoretical worst case complexity depends on the graph cut algorithm used, and is the same as for usual single-image graph cuts, but with T times more nodes and edges. Denoting by N the number of nodes and M the number of edges, this worst case complexity is $O(NM^2)$ for the Edmonds-Karp algorithm, $O(N^2M)$ for the Dinitz blocking flow algorithm, which goes down to $O(NM\log(N))$ using dynamic trees. However, the computational time observed in practice is known to be much faster on typical image segmentation problems. We applied the binary graph-cut algorithm of Boykov and Kolmogorov [18], and we report a linear observed complexity with the total number of pixels $T\times W\times H$ (see for instance Fig. 4.5 for experiments of section 4.3.1). As a consequence, enforcing shape growth on a long sequence, or on complementary shorter bits of the same sequence, will take approximately the same time.

In the case of long sequences of big images, the memory space required may exceed the capacities of a computer. This is however not an issue, as there exist graph cut implementations for massive grids [38] meant for such cases, where all information is not stored in the memory at all times. This was not required for the experiments presented in this paper though.

4.2.5 Rewriting as a multi-label problem

We show now another point of view on sequence segmentation with growth constraint. The successive labels $L_i(t)$ of a given pixel i over time might change only once, and only from 0 (background) to 1 (foreground object). Hence, this vector of labels $L_i(t)$ is of the form $(0,0,\ldots,0,1,\ldots,1,1)$ and can be represented by just the time index τ_i of the first 1, *i.e.* the earliest time at which the pixel starts belonging to the object. This time τ_i is in $[1,T+1]$, with $T+1$ meaning "never". We have thus transformed a binary optimization problem on a sequence of images with shape growth constraint into a multi-label problem defined on one single image, without any constraint. This new problem can be expressed in the Markov Random Field form (4.1) with $V_i(\tau_i) := \sum_{t<\tau_i} V_i^t(0) + \sum_{t\geqslant\tau_i} V_i^t(1)$ and

$$\begin{aligned} W_{i,j}(\tau_i,\tau_j) := & \sum_{t<\min(\tau_i,\tau_j)} W_{i,j}^t(0,0) + \sum_{\tau_i\leqslant t<\tau_j} W_{i,j}^t(1,0) \\ & + \sum_{\tau_j\leqslant t<\tau_i} W_{i,j}^t(0,1) + \sum_{t\geqslant\max(\tau_i,\tau_j)} W_{i,j}^t(1,1) \end{aligned}$$

where (V_i^t) and $(W_{i,j}^t)$ define the energy E^t at time t, and where sets of summation may be empty (note that any time t appears in exactly one summation set only). The submodularity of the binary interaction terms W^t in each frame implies the submodularity of the multilabel interaction term W, *i.e.*

$$W_{i,j}(\tau_1, \tau_2) + W_{i,j}(\tau_1', \tau_2') \leqslant W_{i,j}(\tau_1, \tau_2') + W_{i,j}(\tau_1', \tau_2)$$

for all labels satisfying $\tau_1 \leqslant \tau_1'$ and $\tau_2 \leqslant \tau_2'$ (proof in Appendix). Thus, this energy can be minimized globally efficiently with now standard techniques (*e.g.* [36]). Note that in the particular case where interaction terms W^t do not depend on t, the interaction term W of this multi-label energy above can be rewritten as a convex function g of $(\tau_i - \tau_j)$, and then Ishikawa's construction [84] can be applied. It turns out that the graph built this way is precisely the graph that we built in our initial binary multi-frame problem. Our initial formulation is however more flexible, in that interaction terms can depend on t, and more natural, in that inclusion constraints can easily be enforced in spatial or/and time subregions only, while this would not be expressible with the multi-label formulation.

4.3 Experimental results

We applied the proposed method to three different applications, in order to validate its use to enforce shape shrinkage or growth, for 2D or 3D image sequences:

(Application 1) We impose shrinkage constraint for a 2D sequence, to segment a melting multiyear ice from a time series of satellite measurements.

(Application 2) We enforce shape growth for a 2D sequence, to segment burned areas from optical satellite images.

(Application 3) We use growth constraint for 3D image sequences, to segment growing brain tumors from multimodal MRI volumes.

The performance of our framework with monodirectional links, *[Mono=const]*, is compared with other graph-cut-based methods:

- *[w/o]*: Graph cut with no temporal links, *i.e.* independent segmentation of each frame.
- *[Feedforward]*: After segmenting the first frame with graph cut approach, foreground/background pixels are marked as seeds with infinite unary costs in the next frame for enforcing shape growth/shrinkage.
- *[Bi=const]*: Smoothing by introducing bidirectional temporal links, *i.e.* links in

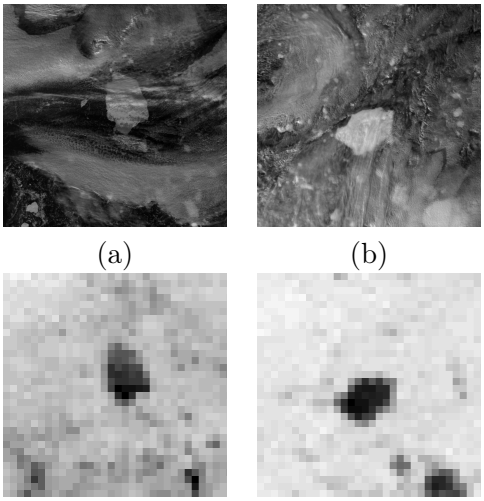


Figure 4.2: (a, b) Reprojected MODIS images captured on the 244th and 252th days, respectively. (c, d) Corresponding AMSR-E images.

both directions (from t to $t+1$ and from $t+1$ to t), with a constant weight (finite or infinite).

- *[Bi=variable]*: As proposed in [205], bidirectional temporal links are computed based on intensity differences between pixels in successive image frames, *i.e.* in the same way as spatial links.

For comparison between the methods we used the Dice score [42], $D = (2|\hat{M} \cap M|)/(|\hat{M}|+|M|)$, where $\hat{M}$ and M are manually and automatically segmented foreground regions, respectively.

4.3.1 Application 1: melting sea ice in satellite images

The melting of sea ice is correlated to the increases in sea surface temperature and to associated climatic changes. Thus it is very important to monitor sea ice evolution and to develop methods for automated analysis of satellite measurements. Previous works on ice floe segmentation attempted to extract temporal information by using ice percentages, area and shape parameters of the ice floes at the previous time moments as prior information to segment the floe at the next time moment [179, 79]. These feed-forward approaches were unable to accurately estimate melting ice profiles because of the low signal-to-noise ratio and of the lack of contrast in satellite data, producing area estimations that are noisy in time, while a multiyear ice floe can actually only melt in

Table 4.1: Results for Application 1. Mean and standard deviations of the Dice scores for the proposed method *[Mono* $= \infty$*]* and graph-cut-based approaches used for comparison.

Method	Feedforward	w/o	Bi = variable	Bi = 16	Mono $= \infty$
Dice score	.554 ± .128	.933 ± .099	.958 ± .048	.978 ± .007	.980 ± .007

the summer period.

We aimed at segmenting a multiyear ice floe from a 45-day sequence (summer period from the 227th to 271th day of 2008) of measurements over the polar regions by two Aqua satellite sensors: Advanced Microwave Scanning Radiometer - Earth Observing System (AMSR-E, 89 GHz, 6.25 km spatial resolution) and Moderate-Resolution Imaging Spectroradiometer (MODIS, band 1, 0.620-0.670 μm, 250 m spatial resolution). Fig. 4.2 shows MODIS and AMSR-E images at two time moments from the considered data set. The floe was tracked from the AMSR-E data, where multiyear ice has a low microwave emissivity (dark area in Fig. 4.2), and is in this way distinguishable from clouds and younger ice which has a higher emissivity (white area in Fig. 4.2). However, the low spatial resolution of these data does not allow to quantify the ice floes areas accurately. In accordance with the tracking measurements, a time series of $T = 75$ MODIS and upscaled AMSR-E images with the ice floe was built, with spatial dimensions of 800×800 pixels. We denote each MODIS image by I^t, and each upscaled AMSR-E image smoothed by Gaussian filter by $A^t, t = 1, ..., T$.

The objective is to compute T segmentation maps L^t, where each pixel (x, y) has label $L^t_{(x,y)} = 1$ if it belongs to the floe at time t, and 0 otherwise. In order to apply the proposed method with a shrinkage constraint to the selected time series, the images must be aligned, so that the property that the floe in the image I^{t+1} is included in the floe of the image at the previous time moment I^t can be expressed directly in terms of pixel locations.

$$\forall t, x, y, \qquad L^{t+1}_{(x,y)} = 1 \implies L^t_{(x,y)} = 1.$$

For this purpose, we estimated a reliable region of the foreground (*i.e.*, a region which can be considered with high probability as a part of the floe), R_F, and a reliable region of the background, R_B, from the AMSR-E images, where a multiyear ice floe is darker than water, young ice and clouds. From the AMSR-E tracking measurements, we derived a foreground seed point and an approximate area of the floe for each time moment t. We then grew a reliable region of the floe R_F in each A^t from the foreground seed point, until it reached approximately half of the size of the floe. Another region, $\overline{R_B}$, was grown in A^t from the same foreground seed point, but until an area half larger than the approximate

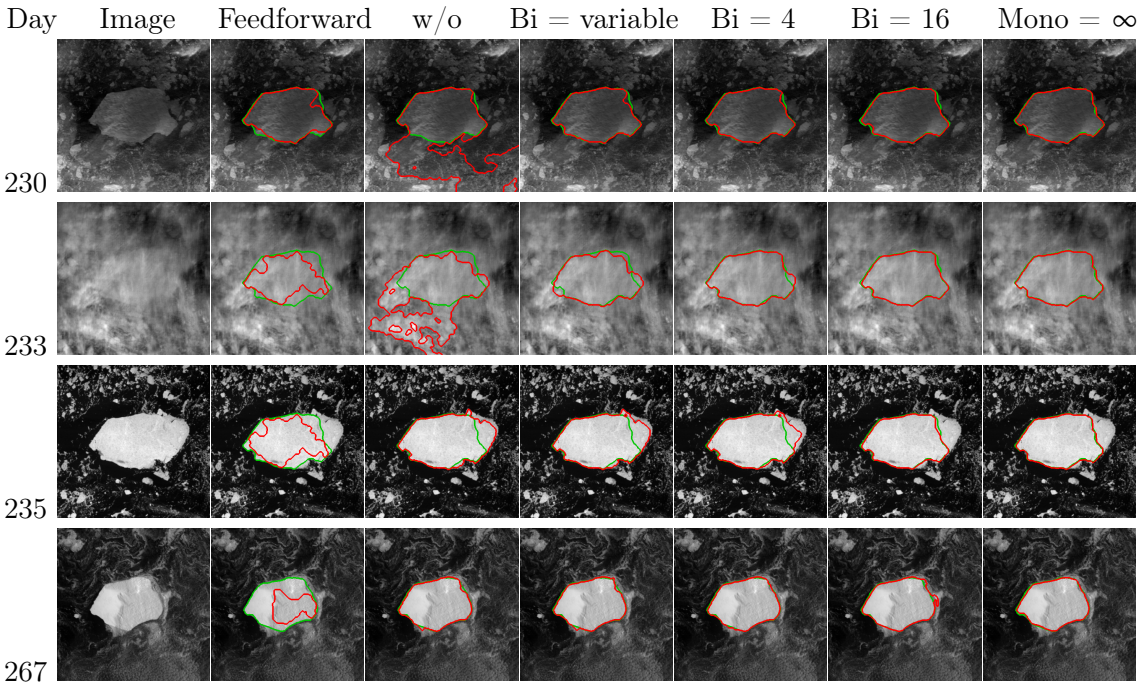


Figure 4.3: (From top to bottom) Aligned images and segmentation contours (red) for four time moments (days 230, 233, 235 and 267, respectively) computed by the graph-cut methods: (From left to right) Aligned image, *[Feedforward]*, *[w/o]*, *[Bi=variable]*, *[Bi=4]*, *[Bi=16]*, proposed method with monodirectional infinite links. Manual segmentation is shown in green. The rightmost part of the white area in the third row is not part of the object, but another ice floe who temporarily collided.

floe size. The complementary region R_B, of pixels not in $\overline{R_B}$, was considered with high probability as part of the background.

We then computed the histograms of the intensities I^t of the floe, $p^t(I|F)$, and of the background, $p^t(I|B)$, respectively, and a map of floe probabilities as

$$p^t(F|I) = \frac{p^t(I|F)P^t(F)}{p^t(I|F)P^t(F) + p^t(I|B)P^t(B)}, \tag{4.4}$$

$$P^t(B) = \frac{A^t - \min\limits_{x,y} A^t_{(x,y)}}{\max\limits_{x,y} A^t_{(x,y)} - \min\limits_{x,y} A^t_{(x,y)}}, \quad P^t(F) = 1 - P^t(B).$$

The quantities above depend on the pixel location. The images I^t were aligned by exhaustive searching over rigid motions (rotations and translations) to maximize the cor-

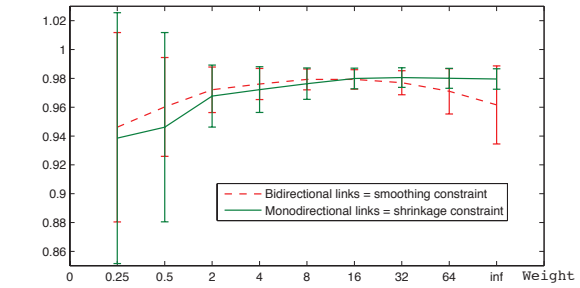
relation between maps of foreground probabilities at the current and previous moments. This search is largely affordable given the low frame rate. We computed potentials and interaction terms between neighboring pixels as:

$$V_i^t(1) = -ln[p^t(F|I)], \quad V_i^t(0) = -ln[p^t(B|I)], \tag{4.5}$$

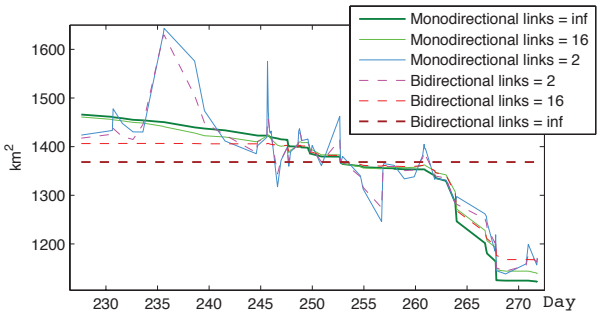
$$W_{i,j}^t = \delta_{L_i \neq L_j}\, \beta \exp\left[-\frac{(I_i^t - I_j^t)^2}{2\sigma^2}\right], \tag{4.6}$$

where σ is a standard deviation of I^t, β is a parameter that controls the importance of the spatial interaction energy term. We found experimentally that setting $\beta = 2$ yields robust results. The proposed method was applied with monodirectional temporal links to enforce floe shrinkage, as described in Sec. 4.2.1. We performed several experiments, with different values of the constant w standing for the temporal link weight, from 0.25 to ∞. The results *[Mono=0.25...∞]* are compared with those obtained with other graph-cut-based approaches (listed in the beginning of Sec. 4.3) in Table 4.1 and in Fig. 4.3-4.4. Both graph-cut with no temporal links and feedforward approaches show the worst performances, and prove to be not well suited for segmenting such noisy data sets. When a feedforward method encounters a frame with the part of the floe obscured by clouds and thus undistinguishable from the background, it segments only the visible part of the foreground, and then is trapped in a non-sense segmentation for the rest of future times. The method using gradient-based temporal links *[Bi=variable]* [205] also yields poor segmentation accuracies, because it is sensitive to both noise and variation of foreground/background intensities in consecutive frames.

We explain in Fig. 4.3-4.4 the advantage of using monodirectional infinite links *versus* bidirectional links in the temporal dimension. Bidirectional edges with low values of w enforce only smoothness of variation of the contour in time, and yield segmentation errors in the case of low foreground/background contrast. For example, in the third image of Fig. 4.3 (day 235), the floe of interest collided temporarily with another ice floe. When using a weak smoothness constraint (see segmentation contour *[Bi=4]*), the small encountered floe collided with the floe of interest during a certain number of consecutive frames would be considered as a part of the foreground. Enforcing more smoothness in space-time to avoid this has the undesirable effect of smoothing the foreground shape, so that the segmented foreground area is lower (underestimated) than the ground-truth for the first frames, and higher (overestimated) for the last frames (see results *[Bi=16]* in Fig. 4.3 and-4.4(b)). With the increase of w, the estimated foreground tends to the



(a)



(b)

Figure 4.4: Results for Application 1. (a) Mean and standard deviation for the dice score as a function of the temporal link's weight, when using mono- (green) and bidirectional (red) temporal links. (b) Area of a multiyear ice floe as a function of time, computed by using mono- and bidirectional links with different weights.

constant shape for all time moments, and the Dice score decreases.

When the proposed shrinkage constraint is used instead, the segmentation accuracy increases with w, and $w = \infty$ yields results with monotonous shrinkage of the shape area (see Fig. 4.4(a)). Moreover, the proposed method copes well with rapid shrinkage events, without underestimating preceding images, or overestimating the event itself at onset. Another advantage of using monodirectional infinite links is that there are no additional parameters to quantify temporal coherency.

Fig. 4.5 depicts the computational time for the proposed graph-cut-based optimization as a function of the number of frames. The total computational time grows linearly with the number of frames, and is approximately twice the time that would be taken by the independent segmentation of each frame.

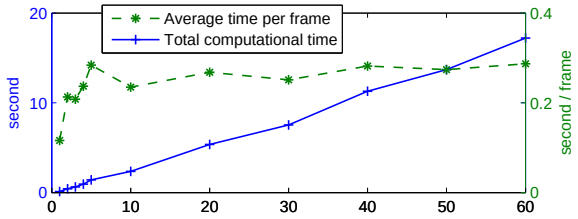


Figure 4.5: Computational time for the proposed segmentation of the ice floe set as a function of the number of frames.

4.3.2 Application 2: growing burned areas in satellite observations

Biomass burning has a significant impact on the Earth's climate system. Satellite remote sensors acquire data for the continuous monitoring of burned areas at both regional and global scales. Thus, there is a need to develop methods for automated fire mapping. While most of the existing techniques for mapping burned areas analyze temporal evolution of each pixel in an image scene [155], recent studies have proved the advantage of considering spatial contextual classification for accurate fire classification [115, 62]. Both works [115, 62] map fires from MODIS data by detecting and classifying persistent changes in a daily vegetation-index time series. Giglio *et al.* [62] exploit the closest fixed pixel's neighborhood to refine fire classification. Lewis [115] segments and analyzes change detection maps between two consecutive time moments. Manual post-processing is needed to correct classification errors, which are a consequence of either a cloud cover, or low contrast between burned and unburned areas.

In our study we analyzed two time series of Terra MODIS atmospherically-corrected Level 2G daily surface reflectance measurements over the tropical savannas in the Northern Australia ("MOD09GA" product, tile h31v10), each of the data sets being acquired during forty days of the dry season in September-October (days 244-283) of 2011 and 2013, respectively. Wildfires in this region of Australia are frequent and extensive. We used MODIS band 5 (1.240 μm) 500-m land surface reflectance data as they provide the highest burned-unburned separability and are largely insensitive to smoke aerosols [155]. Each time series comprised a set of $T = 40$ images with spatial dimensions $W \times H = 400 \times 400$ pixels. Fig. 4.7(a) shows three images from each of the considered sets, where black pixels denote missing data (MODIS does not provide 100% daytime coverage of the terrestrial surface every day).

We used MODIS Collection 5.1 Direct Broadcast Monthly Burned Area Product (MCD64A1, see Fig. 4.7(b)) [62] for learning and testing the proposed method, *i.e.* for

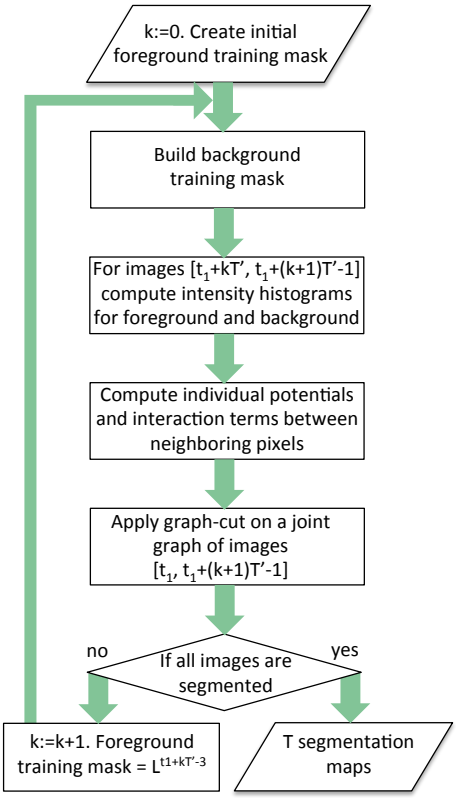


Figure 4.6: Flowchart of the segmentation algorithm applied for Application 2.

computing an initial histogram of burned areas and comparing fire maps, respectively. The MCD64A1 product contains fire classification maps, where each pixel is associated with either an estimated day of burn, or an unburned flag, or an unmapped flag due to insufficient data. These maps are computed by applying the approach from [62] on two 500-m MODIS channels coupled with 1-km MODIS active fire observations.

We segmented each of the considered image time series $[t_1, ..., t_T]$ by applying the following iterative procedure (the flowchart is shown in Fig. 4.6):

Initialization: $k := 0$. The initial training mask of burned areas R_k^B is built using MCD64A1 product, by selecting the pixels burned during the days $[t_1 - D, t_1 - 1]$. This mask can also be created based on ground observations of the considered area on the day $(t_1 - 1)$.

1. The training mask of unburned areas R_k^U is constructed by dilating R_k^B with a disk of radius r [171] and selecting the complementary of the resulting image.

2. For a subset of T' images $t = [t_1 + kT', t_1 + (k+1)T' - 1]$, intensity histograms of

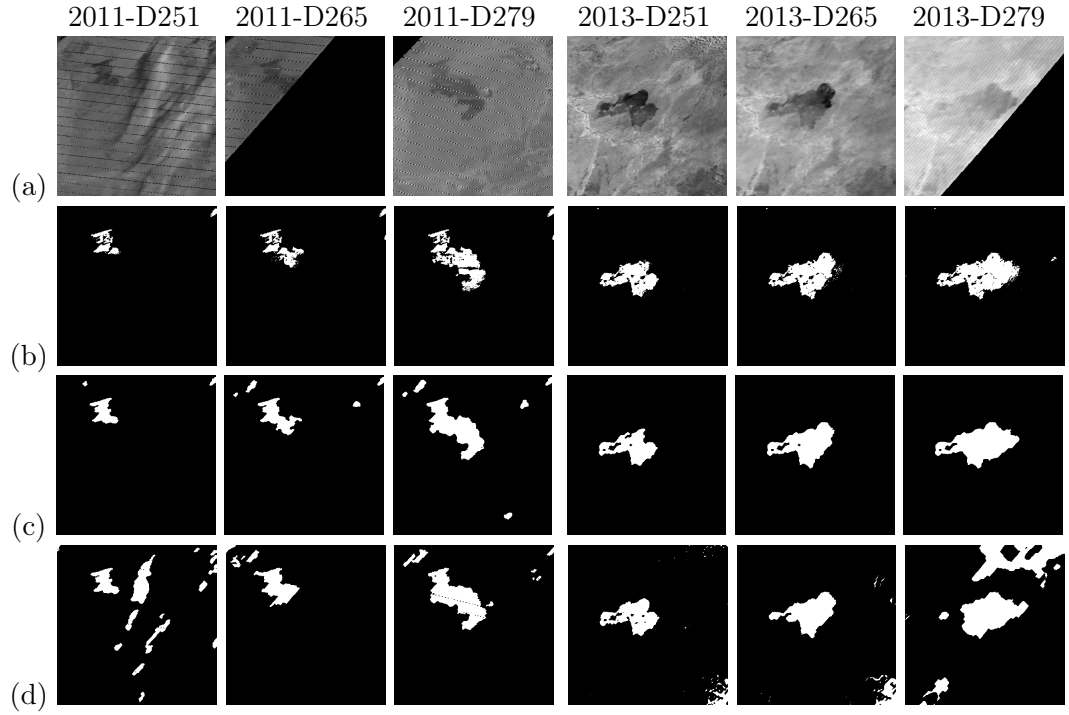


Figure 4.7: Two time series of MODIS images acquired in 2011 and 2013 (left and right, respectively). (a) MODIS band 5 images for three time moments: days 251, 265 and 279. Black pixels denote missing data. (b) Maps from the MCD64A1 product of areas burned during the days 213-251, 213-265 and 213-279, respectively (white pixels = burned areas). (c-d) Segmentation maps for images (a) computed by: (c) the proposed method with growth constraint; (d) *[w/o]* method with no temporal constraints.

the MODIS band 5 for burned $p^t(I|B)$ and unburned $p^t(I|U)$ areas are computed, using the masks R_k^B and R_k^U, respectively. If the data for some pixels is missing, these pixels are not considered when computing histograms.

3. For the images $[t_1 + kT', t_1 + (k+1)T' - 1]$, potentials are computed, assuming equal priors $p^t(B) = p^t(U) = 1/2$:

$$V_i^t(1) = -ln[p^t(B|I_i^t)] = -ln\left[\frac{p^t(I_i^t|B)}{p^t(I_i^t|B) + p^t(I_i^t|U)}\right], \tag{4.7}$$

$$V_i^t(0) = -ln[p^t(U|I_i^t)] = -ln\left[\frac{p^t(I_i^t|U)}{p^t(I_i^t|B) + p^t(I_i^t|U)}\right]. \tag{4.8}$$

If data I_i^t is missing for pixel i at time t, we set $V_i^t(1) = V_i^t(0) = 0$ (no prior).

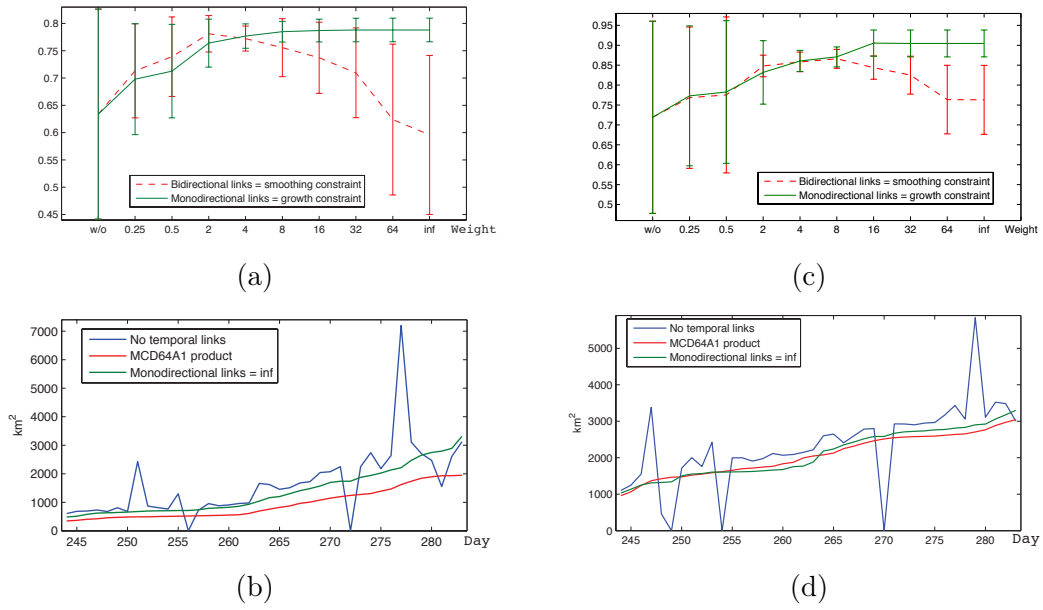


Figure 4.8: Results for Application 2, for two MODIS time series acquired in 2011 (a-b) and 2013 (c-d), respectively. (a, c) Mean and standard deviation for the dice score as a function of the temporal link's weight, when using mono- (green) and bidirectional (red) temporal links. (b, d) Burned area as a function of time, when using no temporal links (blue), monodirectional infinite links (green) and MCD64A1 product (red).

Interaction terms are computed using Eq. 4.6.

4. The graph-cut optimization is applied on a joint graph of the images $[t_1, t_1 + (k+1)T' - 1]$, yielding $(k+1)T'$ segmentation maps.

5. If the whole set of T images is segmented, exit the algorithm. Otherwise: $k := k+1$. The segmentation map $L^{t_1+kT'-3}$ is used as the new training mask of burned areas R_k^B. We do not use the segmentation result of the last frame $(t_1 + kT')$, because extreme frames benefit from less information from neighboring frames, and are therefore more subject to segmentation errors. Go to step 1.

We applied the described algorithm with the parameters empirically set as $D = 31$, $r = 20$, $\beta = 2$ and $T' = 20$, and with different temporal regularizations, *i.e.*, *[w/o]*, *[Mono=0.25...∞]* and *[Bi=0.25...∞]*. Neither *[Feedforward]* nor *[Bi=variable]* methods are suited for segmenting image sequences with missing data. Fig. 4.8(a, c) gives the resulting dice scores for both data sets. We have found that a weak temporal regularization (both bi-/monodirectional) outperforms a segmentation without temporal constraint. Increasing the bidirectional temporal regularization towards high values, however, decreases

the performance. On the opposite, introducing monodirectional infinite links to impose shape growth yields the most accurate results. As can be seen from Fig. 4.7 and 4.8(b, d), the proposed approach achieves comparable results with the method [62] by using only one MODIS channel and no post-processing, while [62] used two MODIS bands coupled with 1-km MODIS active fire observations and post-processing. Furthermore, the new method copes better with the missing or noisy data thanks to the introduced spatio-temporal graph.

4.3.3 Application 3: growing tumor in 3D medical scans

Glioma is the most frequent primary tumor of the brain. The tumor is known to grow steadily, and lesions are evaluated with respect to volume change in different magnetic resonance image (MRI) modalities. In our experiment we evaluated a set of 760 multimodal image volumes – each comprising T1 MRI, contrast-enhanced T1 MRI (T1c), T2 MRI, and T2 FLAIR MRI – acquired from ten patients initially diagnosed with low grade glioma. The time series have 3-14 time points, with 3-6 months in between any two acquisitions. All image volumes were rigidly registered and three 2D slices intersecting with the tumor center were manually annotated through an expert in every volume, representing an approximate truth. Full 3D segmentations for images of each individual time point were obtained using a generative model for multimodal brain tumor segmentation [136]. This algorithm models the lesion with a latent atlas class [153] amending the tissue atlas of the standard EM segmenter [202, 93]. We applied it to each multimodal data set of each time point in an independent fashion. The segmentation model delineates the lesion individually in each modality. It assumes that changes of the core (visible in T1c) will occur within the larger edema regions (visible in T2 or FLAIR) and, hence, to only have class transitions from *healthy* to *edema* and from *edema* to *core*. As the tumor grows steadily, we can assume that negative volume changes stem from imaging artifacts, such as local intensity changes, a common problem in MRI. To this end we model the tumor to be either stable, in this case regularizing the segmentation along time and suppressing noise, or to expand in volume between any two time points.

We identified the foreground label F with tumor (*edema* and *core*) and background B with healthy tissue. Then the potential $V_i^{s,t}(L_i^{s,t})$ of label $L_i^{s,t}$ at voxel i, time point t, and imaging sequence s is equal to $V_i^{s,t}(0) = p_{s,t}(F|I^{s,t})$ and $V_i^{s,t}(1) = p_{s,t}(B|I^{s,t}) = 1 - p_{s,t}(F|I^{s,t})$. The tumor probabilities were calculated from image volumes I using the generative model [136], and we identified tumor subclasses with $p(F|I^{s=T1,t})$ for *core*, and $p(F|I^{s=T2,t})$ with *edema*. We modeled the 3D spatial constraints through a 26 neighbor-

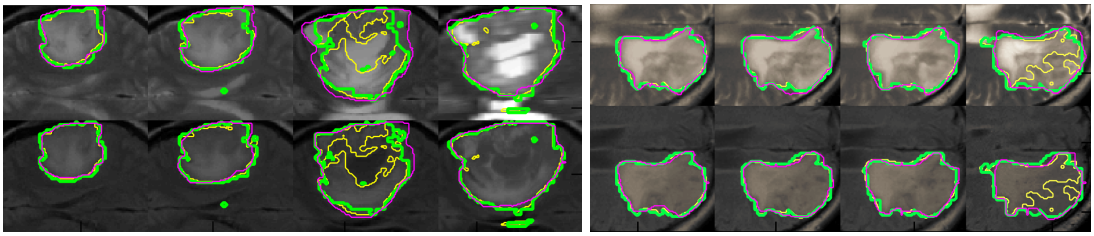


Figure 4.9: Two time series of T2 and FLAIR MR image volumes, each of which acquired about 3-6 months apart. The left case is rapidly growing between the second and fourth scene, the right case displays intensity modifications in the last scene, leading to a suboptimal performance of the initial multimodal segmentation (yellow) [136]. The proposed multi-temporal segmentation with growth constraints (green) delineates areas similar to the manual evaluation (magenta), being more robust against intensity variations of the MR images. It does not smooth out outlines of rapidly growing tumors as conventional bi-directional temporal constraints would do.

hood ($\mathcal{N}$) linking the central voxel with all its immediate neighbors. Interaction terms $W_{i,j}^{s,t}(L_i^{s,t}, L_j^{s,t})$ between neighboring voxels in each sequence $s \in [T1, T2]$ are computed from the channel-specific intensity differences as

$$\delta_{L_i \neq L_j}\ \beta\ \frac{\alpha(i,j)}{\alpha_{\text{tot}}}\ \exp\left(-\left(\frac{I^{s,t}(i) - I^{s,t}(j)}{A}\right)^2\right)$$

with $\beta = 0.5$, $\alpha(p,q) = \frac{1}{\text{distance}(p,q)}$, $\alpha_{\text{tot}} = \sum_{q \in \mathcal{N}(\text{pixel } p)} \alpha(p,q)$ and $A = \frac{1}{3}\left(\max I^{s,t} - \min I^{s,t}\right)$. We impose growth constraint in 3D+t as explained in Sec. 4.2.1, and inclusion constraints as in Sec. 4.2.2 : the foregrounds in T1 and T1c modalities are required to be included in the one of T2, which is included in the one of FLAIR.

In our test we first segmented the images from each time point independently, and calculated Dice scores averaged over all images of a time series as an estimate of the baseline performance. Then we tested different regularizations in time, *i.e.*, *[w/o]*, *[Mono]* and *[Bi]*, calculate Dice scores, and compared them against the baseline results from the unregularized segmentations. Fig. 4.9 shows results for two exemplary time series, and Fig. 4.10(a) reports differences between Dice scores of regularized segmentations and baseline segmentations. In this experiment "weak" regularizations refer to small regularization parameters ($w \ll 1$), while "strong" regularizations refer to $w \gg 1$, representing the infinite mono- and bi-directional links. Fig. 4.10(a) shows results for all ten time series. As for the previous two applications, both bi-/monodirectional temporal regularization with low values of w yields better accuracies when compared to the results without tem-

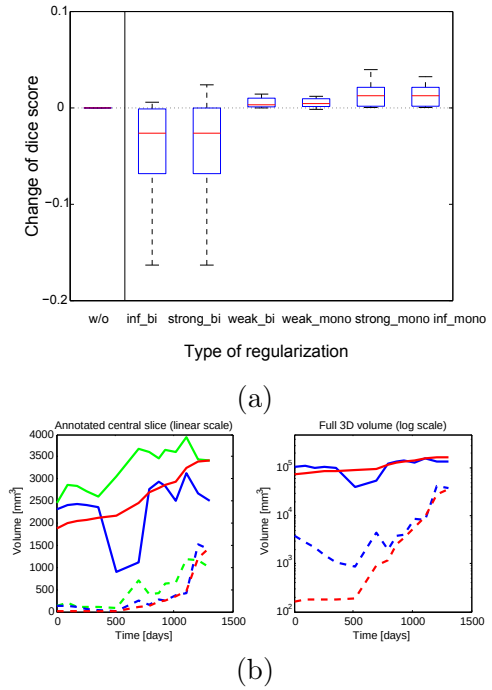


Figure 4.10: Results for Application 3. (a) Boxplots report changes in the average segmentation performance of the ten image sequences when testing different regularization approaches. (The box indicated quartiles, the whiskers outliers.). Using a strong monodirectional regularization acting as a growth prior yields the best results. (b) Volume-time plot for a patient with 14 observations (cmp. video in supplement). Solid lines indicate *edema*, dashed indicate tumor *core* that starts growing with constant rate at around day 500. The segmentation with growth constraint (red) returns results similar to the manual segmentation (green). Segmentations obtained by evaluating image volumes of each time point individually (blue) show significant variation, even obscuring the overall trend.

poral links. Enforcing more smoothness with bidirectional links decreases segmentation accuracies, while introducing monodirectional "growth" regularization through infinite links improves performance (Fig. 4.10(a)). The volume-time graphs (Fig. 4.10(b)) of the segmented tumor structures show as very regular pattern (red), even being smoother than the manual segmentation (green). Moreover, the *log*(volume)-time graph (Fig. 4.10(b), right) shows the exponential growth of the tumor core (dotted lines) that is associated with this disease [15], for this patient indicating a rapid tumor progression starting at about day 500. Longitudinal image segmentation, as obtained for the ten time series in our test, can be further analyzed, for diagnosis and treatment monitoring, *e.g.*, through algorithms estimating the speed of the tumor outlines under anatomical constraints [100].

Extension of the current 5D segmentation could integrate this speed estimation, or extend the multimodal tumor segmentation [136] for longitudinal data sets.

4.4 Conclusion and discussion

In this chapter, we addressed the problem of shape segmentation in 2D and 3D sequences of very noisy/low-contrast images, where shapes monotonously grow or shrink in time. In order to enforce shape growth or shrinkage, we proposed a new graph-cut-based method for computing the globally-optimal spatio-temporal segmentation satisfying that constraint. The main idea was to introduce monodirectional infinite links between pixels at the same spatial locations in successive image frames, which prohibit a shape to shrink or grow over time, and then to perform a graph cut optimization on the constructed graph. The limitation of the proposed method is that it can be applied only to a time series of images on the same scale and perfectly aligned with respect to the foreground object, so that in the case of shape growth the foreground at the moment $(t+1)$ contains all the foreground pixels at the previous moment t, sharing the same spatial locations. Thus, if the foreground object moves over time, images must be aligned before applying the new graph-cut-based technique. We also demonstrated the possibility to impose inter-sequences object inclusion constraints by adding directed infinite links to the joint graph associated to all sequences.

We validated the performance of the proposed approach for the segmentation of growing (burned areas) and shrinking (ice floe) shapes from 2D time series of satellite images, and for the segmentation of growing 3D tumor volumes from MR sequences. The method proved to be robust to important noise and low contrast, and to cope well with missing data. Moreover, it showed linear complexity in practice, so that globally optimal shape-consistent segmentations of image time series are obtained in a matter of seconds.

Another limitation of the presented method is that it is designed and tuned for the specific shape prior, and is thus not generic. Furthermore, even though the complexity of the method is linear, it is not easy to parallelise it, therefore its scalability is limited. During the last decade, the use of deep convolutional neural networks (CNNs) for machine learning tasks has been growing in the vision community. CNNs are able to automatically learn hundreds of task-specific features, which may allow us to easily scale the pool of contextual features considered. Moreover, they are able to incorporate high-level features in terms of simple operations that are highly parallelized in GPUs. With the growing availability of high volumes of satellite images, it is clear that highly parallel approaches will gain attention in the near future. In the next chapters I will describe our research

outcomes on the use of the deep neural network-based methods for remote sensing image classification.

Chapter 5

Deep Learning for Large-Scale Image Classification

There is a growing interest in analyzing remote sensing data at a large scale, i.e. from the scale of cities to the world-scale [109, 135, 138]. This implies several challenges, such as the computational complexity of the methods and intra-class variability. Due to the large spatial extent of the large-scale datasets, the objects belonging to the same semantic class may have considerably different appearance. To address this challenge and conduct an accurate classification, the high-level contextual information must be taken into consideration, such as the shape of objects and their location with respect to other objects.

There is an incipient trend both in the vision and remote sensing literature to design classifiers that automatically learn expressive high-level contextual features. In this context, *deep learning* and, in particular, convolutional neural networks (CNNs), have gained significant attention in the image analysis community over the last years. These have been originally devised for the image *categorization* problem, i.e., the assignment of one label to an entire image. For example, they have been used to categorize objects in natural scenes (e.g., *airplane*, *bird*, *person*) or land use in the case of aerial images (e.g., *forest*, *beach*, *tennis court*). CNNs jointly learn to extract relevant contextual features and conduct the categorization. In addition to the suppression of the feature design process, which is an interesting advantage itself, this technique has consistently beaten alternative methods in a wide range of problems [212]. Nowadays, one can reasonably expect to find CNN-based techniques scoring the best positions in the leaderboards of online image-related contests.

One of the challenges to apply CNN-based methods for remote sensing image classification consists in obtaining fine-grained high-resolution classification maps. This is due

to two main reasons:

a) There is a structural limitation of CNNs to carry out fine-grained classification. If we wish to keep a low number of learnable parameters, the ability to learn long-range contextual features comes at the cost of losing spatial accuracy, i.e., a trade-off between detection and localization. This is a well-known issue and still a scientific challenge [27, 124].

b) In the specific context of remote sensing imagery, there is a significant lack of spatially accurate reference data for training. For example, the OpenStreetMap collaborative database provides large amounts of free-access maps over the Earth, but irregular misregistrations and omissions are frequent all over the dataset. In such circumstances, CNNs cannot do better than learning rough estimates of the objects' locations, given that the boundaries are hardly located on real edges in the training set.

In this chapter, I present two approaches to address this challenge. One approach consists in designing new architectures specifically intended to provide high-resolution outputs. We conduct an in-depth analysis of the recent dense semantic labelling CNNs to establish the desired properties of an ideal semantic labeling CNN, and assess how the existing methods stand with regards to these properties. Out of these observations, we then derive a CNN framework specifically adapted to the high-resolution dense classification problem. In addition to learning features at different resolutions, it learns how to combine these features. By integrating local and global information in an efficient and flexible manner, it outperforms previous techniques when evaluated on public benchmarks of high-resolution aerial image labeling. I also present an aerial image labeling dataset we have developed to assess the capability of CNNs to generalize to different geographic regions.

Another approach is to use first the base CNN as a rough classifier of the objects' locations, and then process this classification using the original image as guidance, so that the output objects better align to real image edges. Different iterative enhancement algorithms have been presented in the literature to progressively improve the coarse CNN outputs, seeking to sharpen object boundaries around real image edges. However, one must carefully design, choose and tune such algorithms. Instead, we have proposed to directly learn the iterative process itself. For this, we formulate a generic iterative enhancement process inspired from partial differential equations, and observe that it can be expressed as a recurrent neural network (RNN). Consequently, we train such a network from manually labeled data with the purpose to correct image classification maps. The designed RNN automatically discovers relevant data-dependent features to enhance the classification and the specifics of an iterative process to do so.

5.1 High-resolution semantic labeling with convolutional neural networks

5.1.1 Background on convolutional neural networks

An artificial neural network is a system of interconnected neurons that pass messages to each other. When the messages are passed from one neuron to the next one without ever going back (i.e., the graph of message passing is acyclic), the network is referred to as feed-forward [14], which is the most common type of network in image categorization. An individual neuron takes a vector of inputs $\mathbf{x} = x_1 \dots x_n$ and performs a simple operation to produce an output a. The most common neuron is defined as follows:

$$a = \sigma(\mathbf{w}\mathbf{x} + b), \tag{5.1}$$

where $\mathbf{w}$ denotes a weight vector, b a scalar known as *bias* and σ an activation function. The weights $\mathbf{w}$ and biases b are the parameters of the neurons that define the function. The goal of training is to find the optimal values for these parameters, so that the function computed by the neural network performs the best on the task assigned.

The most common activation functions σ are sigmoids, hyperbolic tangents and rectified linear units (ReLU). For image analysis, ReLUs have become the most popular choice due to some practical advantages at training time, but novel activation functions have been recently proposed as well [29].

Instead of directly connecting a huge set of neurons to the input, it is common to organize them in groups of stacked layers that transform the outputs of the previous layer and feed it to the next layer. This enforces the networks to learn hierarchical features, performing low-level reasoning in the first layers (such as edge detection) and higher-level tasks in the last layers (e.g. [51], assembling object parts). For this reason, the first and last layers are often referred to as lower and upper layers, respectively.

In an image categorization problem, the input of the network is an image (or a set of features derived from an image), and the goal is to predict the correct category of the entire image. We can view the pixelwise semantic labeling problem as taking an image patch and categorizing its central pixel. Finding the optimal neural network classifier reduces to finding the weights and biases that minimize a loss L between the predicted labels and the target labels in a training set. Let Ω be the set of possible semantic classes. Labels are typically encoded as a vector of length $|\Omega|$ with values '1' at the position of the correct label and '0' elsewhere. The network contains thus as many output neurons as possible labels. A softmax normalization is performed on top of the last layer to

guarantee that the output is a probability distribution, i.e. the label values are between zero and one and sum to one. The multi-label problem is then seen as a regression on the desired output label vectors.

The loss function L quantifies the misclassification by comparing the target label vectors $\mathbf{y}^{(i)}$ and the predicted label vectors $\hat{\mathbf{y}}^{(i)}$, for p training samples $i = 1 \dots p$. In this work we use the common cross-entropy loss, defined as:

$$L = -\frac{1}{p} \sum_{i=1}^{p} \sum_{k=1}^{|\Omega|} y_k^{(i)} \log \hat{y}_k^{(i)}. \tag{5.2}$$

Training neural networks by optimizing this criterion converges faster, compared with, for instance, the Euclidean distance between $\mathbf{y}$ and $\hat{\mathbf{y}}$. In addition, it is numerically stable when coupled with softmax normalization [14].

Note that in the special case of binary labeling we can produce only one output (with targets '1' for positive and '0' for negative). In this case a sigmoid normalization and cross-entropy loss are analogously used, although a multi-class framework can also be used for two classes.

Once the loss function is defined, the parameters (weights and biases) that minimize the loss are found via gradient descent, by computing the derivative $\frac{\partial L}{\partial w_i}$ of the loss function with respect to every parameter w_i, and updating the parameters with a learning rate λ as follows:

$$w_i \leftarrow w_i - \lambda \frac{\partial L}{\partial w_i}. \tag{5.3}$$

The derivatives $\frac{\partial L}{\partial w_i}$ are obtained by *backpropagation*, which consists in explicitly computing the derivatives of the loss with respect to the last layer's parameters and using the chain rule to recursively compute the derivatives of each layer's outputs with respect to its weights and inputs (the inputs being the previous layer's outputs). In practice, instead of averaging over the full dataset, the loss (5.2) is estimated from a random small subset of the training set, referred to as *mini-batch*. This learning technique is known as *stochastic gradient descent*.

Convolutional neural networks (CNNs) are a particular type of neural network. The overall success of CNNs lies mostly in the fact the the networks are forced by construction to learn hierarchical contextual translation-invariant features, which is particularly useful in the context of image analysis.

CNNs contain so-called convolutional layers, a specific type of layer that imposes a number of restrictions compared to a more general fully connected layer, where every neuron is connected to all outputs of the previous layer [111]. Every neuron is associated

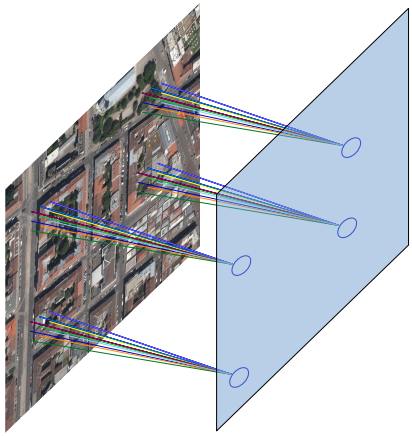


Figure 5.1: Convolutional layer. Connections are limited to a local spatial neighborhood and the weights are shared across the different neurons.

to a spatial location (i, j) in the input image (see Fig. 5.1). The output a_{ij} associated with location (i, j) in a convolutional layer is computed as:

$$a_{ij} = \sigma((\mathbf{W} * \mathbf{X})_{ij} + b), \tag{5.4}$$

where $\mathbf{W}$ denotes a kernel with learned weights, $\mathbf{X}$ the input to the layer and '$*$' the convolution operation. Notice that this is a special case of the neuron in Eq. 5.1 with the following constraints:

- The connections only extend to a limited spatial neighborhood determined by the kernel size;
- The same filter is applied to each location, guaranteeing translation invariance.

Multiple convolution kernels are usually learned in every layer, interpreted as a set of spatial feature detectors. The responses to every learned filter are thus referred to as *feature maps*. Note that the convolution kernels are actually three-dimensional: in addition to their spatial extent (2D), they span along all the feature maps in the previous layer (or eventually through all the bands in the input image). As this third dimension can be inferred from the previous layer it is rarely mentioned in the architecture descriptions.

Compared to the fully connected layer, a convolutional layer highly reduces the number of parameters by enforcing the aforementioned constraints. This results in an easier optimization problem, without losing much generality. This opened the door to using the image itself as an input without any feature design and selection process, as CNNs discover the relevant spatial features to conduct classification.

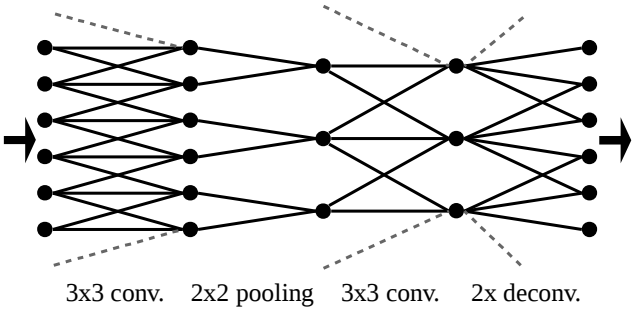


Figure 5.2: Lateral view of a fully convolutional network (dashed lines indicate inputs that have been padded in convolutional layers and cropped in the deconvolutional layer to preserve spatial dimensions).

Increasing the receptive field

In addition to convolutional layers, state-of-the-art networks such as Imagenet [101] involve some degree of downsampling, i.e., a reduction in the resolution of the feature maps with the purpose to increase the so-called *receptive field* of the neurons. The *receptive field* denotes the spatial extent of the input image connected to a certain neuron, possibly indirectly through other neurons in previous layers: it is the set of pixels on which a neuron depends. In other words, it quantifies how far a neuron can "see" in the image. In most applications, a large amount of spatial context must be taken into account in order to successfully label the images. For example, to deduce that a certain pixel belongs to a rooftop, it might not be enough to just consider its individual spectrum: we might need to observe a large patch around this pixel, taking into account geometry and structure of the objects, to infer its correct class.

Neural networks for image analysis should thus be designed to accumulate, through their layers, a large enough receptive field. While a straightforward way to do it is to use large convolution kernels, this is not a common practice mostly due to its computational complexity. Besides, this would aim at learning large filters all at once, with millions of parameters. It is preferable to learn a hierarchy of small filters instead, reducing the number of parameters while remaining expressive, and thus making the optimization problem easier.

The most common approach to reduce the number of parameters for a given receptive field size is to downsample the feature maps throughout the network. This is commonly achieved by interleaving downsampling layers with convolutional layers. This way, the resolution of the feature maps gets lower and lower as we traverse the layers from input to output.

To downsample the feature maps, the most popular approach is to use the so-called *max pooling* layer [17]. A max pooling layer takes a group of neighbors in the feature map and condenses them into a single output by computing the maximum of all incoming activations in the window. The pooling windows in general do not overlap, hence the output map is downsampled (see Fig. 5.2). For instance, if pooling is performed in a 2×2 window, the feature map is reduced to half of its resolution.

Computing the maximum value is inspired by the idea of detecting objects from their parts. For example, in a face detector it is important to identify the constituents of a face, such as *hair* or *nose*, while the exact locations of these components should not be such a determinant factor. The max pooling layer conveys then to which extent there is evidence of the *existence* of a feature in a vicinity. Other less popular forms of downsampling include average pooling and applying convolutions with a *stride*, i.e., "skipping" some of them (e.g., applying every other convolution).

Pooling operations (and downsampling in general) hard-code robustness to spatial deformations, a virtue that boosted the success of CNNs for image categorization. However, spatial precision is lost when downsampling. The increased receptive field (and thus recognition capability) comes at the price of losing localization capability. This well-reported trade-off [124, 27] is a major concern for dense labeling.

We could still imagine a downsampling network that preserves localization: it would learn features of the type "a corner at the center of the receptive field", "a corner one pixel left of the center of the receptive field", "a corner two pixels left of the center of the receptive field", and so on, multiplying the number of features to be learned. This would however discredit the use of downsampling to gain robustness to spatial variation in the first place. The recognition/localization trade-off must thus be properly addressed to design a high-resolution semantic labeling network.

5.1.2 Overview of high-resolution labeling CNNs

Fully convolutional networks (FCNs) [124] have become the standard in semantic labeling, including in remote sensing image analysis [198, 91]. They contain only convolutional layers, i.e., no fully connected layers. Therefore, they can be applied to images with various sizes: inputting a larger image patch produces a larger output, the convolutions being performed on more locations. When an FCN has downsampling layers, the output contains fewer elements than the input, since the resolution has been decreased. This gave birth to the so-called deconvolutional (or upconvolutional) layer, which upsamples a feature map by interpolating neighboring elements (as the last layer in Fig. 5.2). However,

(a) (b)

Figure 5.3: To classify the central gray pixel of this patch (and not to confuse it, e.g., with an asphalt road), we need to take into account a spatial context (a). However, we do not need a high resolution everywhere in the patch. It can be lower as we go away from the central pixel and still identify the class (b).

simply adding a deconvolutional layer to upsample the output on top of a network provides dense outputs but imprecise labeling results, because the upsampling is performed in a naive way from the coarse classification. This is dissatisfying in many applications, such as high-resolution aerial image classification, where the goal is to precisely identify and outline tiny objects such as cars. The open question is then which type of FCN would be able to conduct fine predictions that provide detailed high-resolution outputs, while still taking large amounts of context into account and without exploding the number of trainable parameters.

We now describe what we consider to be the elementary principle from which to derive efficient dense classification architectures. Let us then first observe that while our goal is to take large amounts of context into account, we do not need this context at the same spatial resolution everywhere. Suppose we want to classify the central pixel of the patch in Fig. 5.3(a). Such a gray pixel, taken out of context, could be easily confused with an asphalt road. Considering the whole patch at once helps to infer that the pixel belongs indeed to a gray rooftop. However, two significant issues arise if we take a full-resolution large patch for context: a) it requires many computational resources that are actually not needed for an effective labeling, and b) it does not provide robustness to spatial variation (we might actually not care about the *exact* location of certain features to determine the class). Conducting predictions from low-resolution patches instead is not a solution as it produces inaccurate coarse classification maps. Nevertheless, it is actually not necessary to observe all surrounding pixels at full resolution: the farther we go from the pixel we want to label, the lower the requirement to know the exact location of the objects. For example, in the patch of Fig. 5.3(b) it is still possible to

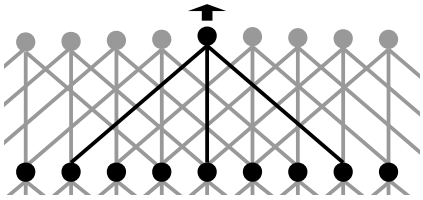


Figure 5.4: A dilated convolution (i.e., on non-adjacent inputs) with a dilation factor $S = 4$.

classify the central pixel, despite the outer pixels being blurry. Therefore, we argue that a combination of reasoning at different resolutions is necessary to conduct fine labeling, if we wish to take a large context into account in an efficient manner.

In the following, we analyze the main families of high-resolution classification networks that have been proposed in the past two years. For each of them we discuss the following aspects:

- How a solution to the fine-grained labeling problem is provided;
- Where this solution stands with respect to the principle of Fig. 5.3;
- General advantages and disadvantages, and computational efficiency.

Dilation Networks

Dilation networks are based on the shift-and-stitch approach or *à trous* algorithm [124]. This consists in conducting a prediction at different offsets to produce multiple low-resolution outputs, which are then interleaved to compose the final high-resolution result. For example, if the downsampling factor of a network is S, one should produce S^2 classification maps by shifting the input horizontally and vertically. Such an interleaving can also be implemented directly in the architecture, by using "dilated" operations [211], i.e., performing them on non-contiguous elements of the previous feature maps. This principle is illustrated in Fig. 5.4.

Dilations have been used with two purposes:

1. As an alternative to upsampling for generating full-resolution outputs [43, 124].
2. As a means to increase the receptive field [27, 211], by enlarging the area covered by a convolution kernel, without increasing the number of trainable parameters.

Regarding the first point, we must mention that there is no theoretical improvement compared to an FCN with naive upsampling, because the presence of pooling layers still

reduces spatial precision. Executing the prediction multiple times at small offsets still keeps predictions spatially imprecise.

Regarding the second point, we must remark that while dilated convolutions increase the receptive field, this does not introduce robustness to spatial variation per se. For example, a network with only dilated convolution layers would have a large receptive field but would only be able to learn filters of the type "a building in the center, with a car *exactly* five pixels to the left". This robustness would have to be thus learned, hopefully, by using a larger number of filters.

The use of an interleaved architecture at training time, implemented with dilations, has been however reported to be beneficial. In the context of aerial image labeling, Sherrah [163] recently showed that it outperforms its FCN/upsampling counterpart[1]. The major improvement compared to the FCN/upsampling network was measured in the labeling capabilities of the *car* class, which is a minority class with tiny objects, difficult to recognize [198]. While the dilation strategy is not substantially different from an architectural point of view compared to naive upsampling, some advantages in training might explain the better results: In the upsampling case the network is encouraged to provide a coarse classification that, once upsampled, is close to the ground truth. In the dilation network, on the contrary, the interleaved outputs are directly compared to individual pixels in the ground truth, one by one. The latter seems to better avoid suboptimal solutions that absorb minority classes or tiny objects.

The computational time and memory required by dilation networks are significant, to the point that using GPUs might become impractical even with moderately large architectures. This is because the whole network rationale is applied to many contiguous locations.

Overall, while dilation networks have been reported to exhibit certain advantages, they are computationally demanding and do not particularly address the principle of Fig. 5.3.

Deconvolution Networks (unpooling)

Instead of naively upsampling the classification score maps with one deconvolutional layer, a more advanced approach is to attach a multi-layer network to learn a complex upsampling function. This idea was simultaneously presented by different research groups [3, 144] and later extended to different problems (e.g., [209]). A simple way to implement this idea is to "reflect" an existent FCN, with the same number of layers and

[1] While such architecture is named a "no-downsampling" network in [163], a more appropriate name would probably be "no-upsampling", because there is indeed downsampling due to the max pooling layers.

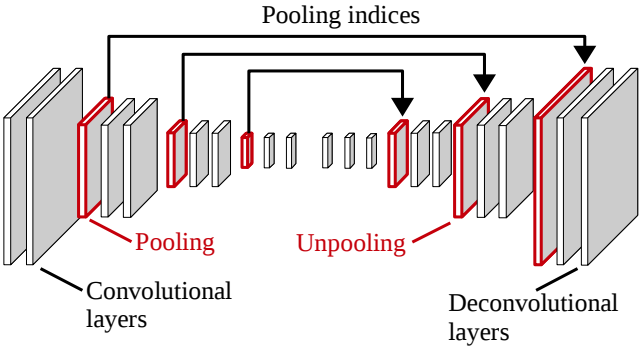


Figure 5.5: Deconvolution network. The CNN is "mirrored" to learn the deconvolution.

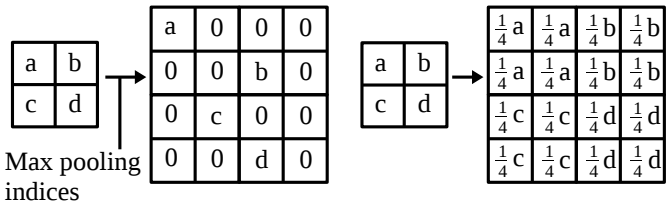


Figure 5.6: Max (left) and average (right) unpooling.

kernel sizes, to perform the upsampling. The convolutional layers are reflected as deconvolutional layers, and the pooling layers as *unpooling* layers (see Fig. 5.5). While pooling condenses several activations into one representative value (typically, the maximum activation), unpooling layers must reconstruct the original size of activations. In the case of max unpooling, the location of the maximal activation is recalled from the corresponding pooling layer, and is used to place the activation back into its original pooled location. The other elements in the unpooling window are set to zero, leading to sparse feature maps, as illustrated in Fig. 5.6. Unpooling was first introduced as part of a framework to analyze and visualize CNN features [212]. The arrows in Fig. 5.5 represent the communication of the pooling indices from the pooling layer to the unpooling layer. In the case of average pooling, the corresponding unpooling layer simply outputs at every location the input activation value divided by the number of elements in the target unpooling window(see Fig. 5.6). In this case, there is no need to transmit a location from pooling to unpooling.

This concept can be thought of as an "encoder–decoder", where the middle layer is seen as a common representation to images and classification maps, while the "encoder" and "decoder" ensure the translation between this representation and the two modalities. When converting an FCN to a deconvolution network, the final classification layer of

the FCN is usually dropped before reflecting the architecture. This way the interface between the encoder and the decoder is a rich representation with multiple features. The first layer of the encoder takes as input as many channels as the ones in the input image, and the last layer of the decoder produces as many feature maps as the number of classes required. In [3, 4], alternatively, the network outputs a larger set of features that are then classified with additional layers.

While pooling is used to add robustness to spatial deformation, the fact of "remembering" the location of the max activation helps to precisely locate objects in the deconvolution steps. For example, the exact location of a road might be irrelevant to do any higher-level reasoning later on, but once the network decides to label the road as a semantic object we need to recover the location information to outline it with high precision. This illustrates how deconvolution networks balance the localization/recognition trade-off.

Note however that if one happens *not* to choose max pooling for downsampling, then the unpooling scheme is not able to recover *per se* the lost spatial resolution. There is no memory about the location of the higher-resolution feature. Even though max pooling is very common, it has been shown that average or other types of pooling might be more effective in certain applications [17]. In fact, recent results [172] suggested that max pooling can be emulated with a strided convolution and achieve similar performance. The deconvolution network idea is however leveraged when max pooling is chosen for downsampling.

This certainly does not mean that a deconvolutional network is incapable of learning without max pooling layers. Convolution/deconvolution architectures without max pooling have been successfully used in different domains [198, 166]. For example, a recent submission to the ISPRS Semantic Labeling Challenge [198] is such type of network. The recognition/localization trade-off is not really alleviated in this case: the encoder should encode features of the type "an object boundary 5 (or 7, 10...) pixels away to the left", so that the decoder can really leverage this information and reconstruct a high-resolution classification map.

The depth of deconvolution networks is significantly larger, roughly twice the one of the associated FCN. This often implies a slower and more difficult optimization, due to the increase in the number of trainable parameters introduced by deconvolutional layers. While the decoding part of the network can be simplified [149], this adds arbitrariness to the design, since instead of just reflecting the encoder we must also design the decoder.

To conclude, the deconvolution scheme does address the recognition/localization trade-off, but only in the case where max pooling is used for downsampling. The in-

creased network depth can be a concern for an effective training.

Skip Networks

In the original paper about fully convolutional networks, Long et al. [124] proposed the so-called "skip" architecture to generate high-resolution classification outputs. The idea is to build the final classification map by combining multiple classification maps, obtained from intermediate features of the network at different resolutions (and not just the last one).

The last layer of an FCN outputs as many feature maps as classes, which are interpreted as score or "heat" maps for every class. Intermediate layers, however, tend to have many more features than the number of classes. Therefore, skip networks add extra layers that convert the arbitrarily large number of features of intermediate layers into the desired number of heat maps. This approach allows us to extract multiple score maps for each class from a single network, at different resolutions. The lower-level score maps are fine but have a small receptive field, while the higher-level ones can see farther but with less detail.

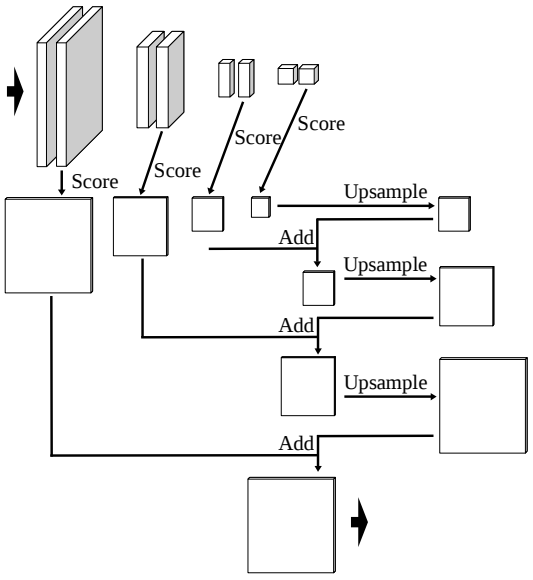


Figure 5.7: Skip network: multiple classification scores are obtained from intermediate CNN features at different resolutions, and are combined by element-wise adding and upsampling.

The score maps are then combined pairwise, from the lower scales to the higher scales. At every step, the lower-resolution score maps are upsampled to match the

higher-resolution ones. They are then added elementwise. This is repeated until all intermediate maps are processed. A skip network is illustrated in Fig. 5.7.

Skip networks address the trade-off between localization and precision quite explicitly: the information at different resolutions is extracted and combined. The original paper introduces this methodology as "combining what and where". This approach is closer to the principle described in Fig. 5.3 than the previous approaches reviewed above. The skip network mixes observations at different resolutions, without unnecessarily increasing the depth or width of the architecture (as in deconvolution and dilation networks respectively) and it does not impose a particular type of downsampling (as in deconvolution networks).

While the idea of extracting different resolutions is certainly very relevant, the skip model seems to be inflexible and arbitrary in how to combine them. First of all, it combines classification verdicts, instead of a rich set of features, coming from each of the resolutions. For example, it combines how a layer evaluates that an object is a building by using low-level information, with how another layer evaluates whether the same object is a building by using higher-level information. Let us recall that we use deep multi-layer schemes with downsampling because we actually consider that certain objects can only be detected at the upper layers of the network, when a large amount of context has been taken into account and at a high level of abstraction. It seems thus contradictory to try to refine the boundaries of an object detected at a high level, by using a classification conducted at a lower level, where the object might not be detected at all. Moreover, the element-wise addition restricts the combination of resolutions to be simply a linear combination. The skip links to combine resolutions are in fact parameterless (besides the addition of the scoring layers). We could certainly imagine classes that require a more complex nonlinear combination of high- and low-level information to be effectively classified.

It is worth noting that the creation of the intermediate score maps has also been referred to as a dimensionality reduction step [4]. It is however not by chance that the amount of reduced features coincides with the amount of classes: even though it is technically a dimensionality reduction, its spirit is to create a partial classification, not just to reduce the number of features. This is confirmed by the name of these layers in the public implementation by Long et al. [124]: "score" layers. Moreover, if this operation were indeed intended to be just a reduction of dimensionality, we could imagine outputting different amounts of feature maps from different resolutions. However, in that case there would be no way of adding them element by element as suggested. Hariharan et al. [78] also extract intermediate features ("skip" links) from a CNN. Additional convolutional layers are used to derive a fixed number of features from each of these intermediate layers,

and then added to give the final output.

To conclude, the skip network architecture provides an efficient solution to address the localization/recognition trade-off, yet this could be done in a more flexible way that enables a more complex combination of the features.

5.1.3 Learning to combine resolutions

In this section I describe our proposed alternative scheme for high-resolution labeling, derived as a natural consequence of the observations described hereabove. In particular, this architecture leverages the benefits of the skip network while addressing its potential limitations.

Taking multiple intermediate features at different resolutions and combining them seems to be a sensible approach to specifically address the localization/recognition trade-off, as done with skip networks. In such a scheme, the high-resolution features have a small receptive field, while the low-resolution ones have a wider receptive field. Combining them constitutes indeed an efficient use of resources, since we do not actually *need* the high-resolution filters to have a wide receptive field, following the principle of Fig. 5.3.

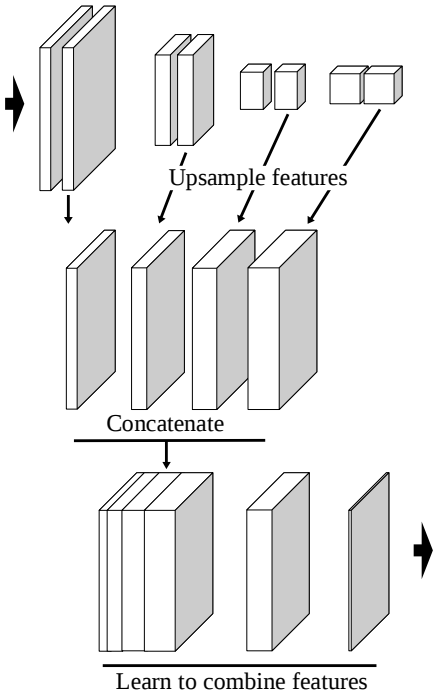


Figure 5.8: MLP network: intermediate CNN features are concatenated, to create a pool of features. Another network learns how to combine them to produce the final classification.

The skip network combines *predictions* derived from the different resolutions, i.e., score maps for each of the classes. For example, we try to refine the "blobby" building outputted by the coarse classifier, via a higher-resolution classification. However, it is unclear how effectively the higher-resolution classifier detects such building, considering its reduced receptive field and shallow reasoning.

We thus argue that a more relevant way of performing fine semantic labeling is to combine features, not classification maps. For example, to refine the boundaries of a coarse building, we would use high-resolution edge detectors and not high-resolution building detectors.

In our proposed scheme, intermediate features are extracted from the network and treated equally, creating a pool of features that emanate from different resolutions. A neural network then learns how to combine these features to give the final classification verdict. This adds flexibility to learn more complex relations between the different resolutions and generalizes the element-wise addition of the skip architecture.

The overall process is depicted in Fig. 5.8. First, a subset of intermediate features are extracted from the network. These are naively upsampled to match the resolution of the highest-resolution features. They are then concatenated to create the pool of features. Notice that while the spatial dimensions of the feature maps are all the same, they originally come from different resolutions. This way, the variation of the feature responses across space will be smoother in certain maps and sharper in others. Note that while it is practical to store in memory the upsampled responses, this is not intrinsically necessary. For example, we could imagine a system that answers to a high-resolution query by outputting the nearest neighbor in the coarser map or by interpolating neighboring values on the fly.

From the pool of features, a neural network predicts the final classification map (we could certainly use other classifiers, but this lets us train the system end to end). We assume that all the spatial reasoning has been conveyed in the features computed by the initial CNN. This is why we operate on a pixel-by-pixel basis to combine the features. Any need to look at neighbors should be expressed in the spatial filters of the CNN. This way we conceptually and architecturally separate the extraction of spatial features from their combination.

We can think of the multi-layer perceptron (MLP) with one hidden layer and a non-linear activation function as a minimal system to learn how to combine the pool of features. Such MLPs can learn to approximate any function and, since we do not have any particular constraints, it seems an appropriate choice. In practice, this is implemented as a succession of convolutional layers with 1×1 kernels, since we want the same MLP to be

applied at every location. By introducing the MLP and executing it at a fine resolution, we must expect an overhead in processing time compared to the skip network.

The proposed technique is intended to learn how to combine information at different resolutions, not how to upsample a low-resolution classification. An example of the type of relation conveyed by this scheme is as follows: "label a pixel as *building* if it is red and belongs to a larger red rectangular structure, which is surrounded by areas of green vegetation and near a road".

Finally, let us discuss the CNN from which features are extracted (the topmost part of Fig. 5.8). The different features are extracted from intermediate layers of a single CNN. This assumes that the higher-level features can be derived from the lower-level ones. It is basically a part-based model [51], where we consider that an object can be detected by its parts, and we are using those same parts as the higher-resolution features inputted to the MLP. This seems to be a sensible assumption, yet we must mention that we could eventually think of separate networks to detect features at different resolutions instead of extracting intermediate representations of a single network (as, e.g., in [47]). While we adopt the model of Fig. 5.8 in this work, the alternative could be also considered. It would be certainly interesting to study to which extent it is redundant to learn the features in separate networks and, conversely, how results could be eventually improved by doing it.

5.1.4 Experiments on Potsdam and Vaihingen datasets

We evaluate the aforementioned architectures on two benchmarks of aerial image labeling: Vaihingen and Potsdam, provided by Commission III of the ISPRS [85]. The Vaihingen dataset is composed of 33 image tiles (of average size 2494×2064), out of which 16 are fully annotated with class labels. The spatial resolution is 9 cm. Near infrared (NIR), red (R) and green (G) bands are provided, as well as a digital surface model (DSM), normalized and distributed by [60]. We select 5 images for validation (IDs: 11, 15, 28, 30, 34) and the remaining 11 images for training, following [163, 198, 146].

Potsdam dataset consists of 38 tiles of size 6000×6000 at a spatial resolution of 5 cm, out of which 24 are annotated. It provides an additional blue channel (which we here exclude for simplicity) and the normalized DSM. We select the same 7 validation tiles as in [163] (IDs: 2_11, 2_12, 4_10, 5_11, 6_7, 7_8 7_10) and the remaining 17 tiles for training. Both datasets are labeled into the following six classes: *impervious surface*, *building*, *low vegetation*, *tree*, *car* and *clutter/background*.

In order to account for labeling mistakes, another version of the ground truth with

Table 5.1: Architecture of our base FCN.

Layer	Filter size	Number of filters	Stride	Padding
Conv-1_1	5	32	2	2
Conv-1_2	3	32	1	1
Pool-1	2		2	
Conv-2_1	3	64	1	1
Conv-2_2	3	64	1	1
Pool-2	2		2	
Conv-3_1	3	96	1	1
Conv-3_2	3	96	1	1
Pool-3	2		2	
Conv-4_1	3	128	1	1
Conv-4_2	3	128	1	1
Pool_4	2		2	
Conv-Score	1	5	1	

eroded boundaries is provided, on which accuracy is measured. To evaluate the overall performance, overall accuracy is used, i.e., the percentage of correctly classified pixels. To evaluate class-specific performance, the F1-score is used, computed as the harmonic mean between precision and recall [33]. We also include the mean F1 measure among classes, since overall accuracy tends to be too insensitive to minority classes in imbalanced datasets.

Network architectures

To conduct our experiments we depart from a base fully convolutional network (FCN) and derive other architectures from it. Table 5.1 summarizes our base FCN for the Vaihingen dataset. The architecture is borrowed from [91], except for the fact that we increased the size of the filters from 3 to 5 in the first layer, since it is a common practice to use larger filters if there is a stride. Every convolutional layer (except the last one) is followed by a batch normalization layer [83] and a ReLU activation. We did not optimize the architecture of the base FCN. Padding refers to the amount of zero-valued pixels added around the input to a convolutional layer, so as to preserve the dimension of the feature maps (otherwise the convolution is not applied near the border and the maps become slightly smaller).

The total downsampling factor is 16, out of which 8 is the result of the max pooling layers and 2 of the stride in the first layer. The conversion of the last set of features to classification maps (the "score" layer) is performed by a 1×1 convolution. To produce a

dense pixel labeling we must add a deconvolutional layer to upsample the predictions by a factor of 16, thus bringing them back to the original resolution.

To implement a *skip* network, we extract the features of layers *Conv-*_2*, i.e., produced by the last convolution in each resolution and before max pooling. Additional scoring layers are added to produce classification maps from the intermediate features. The resulting score maps are then combined as explained in Section 5.1.2. Our MLP network was implemented by extracting the same set of features. As no intermediate scores are needed, we remove layer 'Conv-Score' from the base FCN. The features are combined as explained in Section 5.1.3. The added multi-layer perceptron contains one hidden layer with 256 neurons.

We also created a deconvolution network that exactly reflects the base FCN (as in [144]). This is straightforward, with deconvolutional and unpooling layers associated to every convolutional and pooling layer. The only particularity is that the last layer outputs as many maps as required classes and not as input channels. We here call it *unpooling* network, to differentiate it in the experiments from another method that uses a stack of deconvolutions but without unpooling [198], which we simply refer to as *deconvolution* network. To cover the last family of architectures of Sec. 5.1.2, the *dilation* network, we incorporate the results recently presented by Sherrah [163].

In both datasets we use the same four input channels: DSM, NIR, R and G. Notice that we simply add the DSM as an extra band. In the case of Vaihingen we predict five classes, ignoring the clutter class, due to the lack of training data for that class. In the case of Potsdam we predict all six classes.

Considering the difference in resolution in both datasets, in the case of Potsdam we downsample the input and linearly upsample the output by a factor of 2 (following [163]). We use the same architecture as for Vaihingen (besides the different number of output classes) between the downsampling and upsampling layers. This is to cover a roughly similar receptive field in terms of meters (and not pixels) for both datasets.

Training

The networks are trained by stochastic gradient descent [14]. In every iteration a group of patches is fed to the network for backpropagation. We sample random patches from the images, performing random flips (vertically, horizontally or both) and transpositions, augmenting the data 8 times. At every iteration we group five patches in the mini-batch, of size 256×256 for Vaihingen dataset and 512×512 for Potsdam (to roughly cover the same geographical area, considering the difference in resolution). In all cases, gradient

Table 5.2: Numerical evaluation of architectures derived from our base FCN on the Vaihingen validation set.

	Imp. surf.	Building	Low veg.	Tree	Car	Mean F1	Overall acc.
Base FCN	91.46	94.88	79.19	87.89	72.25	85.14	88.61
Unpooling	91.17	95.16	79.06	87.78	69.49	84.54	88.55
Skip	91.66	95.02	79.13	88.11	77.96	86.38	88.80
MLP	**91.69**	**95.24**	**79.44**	**88.12**	**78.42**	**86.58**	**88.92**

Table 5.3: Numerical evaluation of architectures derived from our base FCN on the Potsdam validation set.

	Imp. surf.	Building	Low veg.	Tree	Car	Clutter	Mean F1	Acc.
Base FCN	88.33	93.97	84.11	80.30	86.13	75.35	84.70	86.20
Unpooling	87.00	92.86	82.93	78.04	84.85	72.47	83.03	84.67
Skip	89.27	94.21	84.73	**81.23**	93.47	75.18	86.35	86.89
MLP	**89.31**	**94.37**	**84.83**	81.10	**93.56**	**76.54**	**86.62**	**87.02**

descent is run with a momentum of 0.9, and an L2 penalty on the network's parameters of 0.0005. Weights are initialized following [80] and, since we use batch normalization layers before ReLUs, there is no need to normalize the input channels.

We start from a base learning rate of 0.1 and anneal it with an exponential decay. The decay rate is set so that the learning rate is divided by ten every 10,000 iterations in the case of Vaihingen and every 20,000 iterations in Potsdam. We decrease the learning rate more slowly in the case of Potsdam because the total surface covered by the dataset is larger, thus we assume it must take longer to explore. Training is stopped after 45,000 iterations in the first dataset and 90,000 in the second one, when the error stagnates on the validation set.

To train the unpooling, skip and MLP networks we initialize the weights with the pretrained base FCN, and jointly retrain the entire architecture. We start this second training phase with a learning rate of 0.01, and stop after 30,000 and 65,000 iterations for Vaihingen and Potsdam datasets respectively. We verified that the initialization with the pretrained weights is indeed beneficial compared to training from scratch.

Numerical results

In this section I first present how the base FCN network compares to its derived architectures: unpooling, skip and MLP. I then position MLP with respect to other results reported in the literature, including a dilation network, thus completing the evaluation over all four families of techniques. I finally discuss our submission to the ISPRS contest.

Comparison of a base FCN to its derived unpooling, skip and MLP networks The classification performances on the validation sets are included in Tables 5.2 and 5.3, for Vaihingen and Potsdam datasets, respectively. The MLP network exhibits the best performance in almost every case. The skip network effectively enhances the results compared with the base network, yet it does not outperform MLP. Let us remark that the unpooling strategy does not necessarily improve the base FCN. This might be a result of the increased training difficulty due to the depth of the network and the sparsity of the unpooled maps. Our attempts to modify the training scheme did not conduce to improving its performance.

Overall, the numerical results show that the incorporation of lower-resolution features significantly improves the classification accuracy. MLP is the most competitive method, boosting the performance by learning how to combine these features.

Comparison with other methods Tables 5.4 and 5.5 (for Vaihingen and Potsdam datasets respectively) incorporate the numerical results reported by other authors using the same training and validation sets. Since not every method was applied to both datasets, the tables do not display exactly the same techniques. The MLP approach also outperforms the dilation strategy, in both datasets, thus positioning it as the most competitive category among those presented in Sections 5.1.2, 5.1.3 (dilation, unpooling, skip, MLP).

In the case of Vaihingen dataset, we also report the results of the *deconvolution* network [198], commented in Sec. 5.1.2, which performs upsampling by using a series of deconvolutional layers. Contrary to the *unpooling* network, the decoder does not exactly reflect the encoder and no unpooling operations are used. Additionally, we include the performance of other methods recently presented in the literature: the CNN+RF approach [146], which combines a CNN with a random forest classifier; the CNN+RF+CRF approch, which adds CRF post-processing to CNN+RF; and Dilation+CRF [163], which adds CRF post-processing to the dilation network. As depicted in the table, the MLP approach outperforms these other methods too.

For Potsdam dataset, Table 5.5 reports the performance of two other methods, presented in [163]. In both cases, a pretrained network based on VGG [167] is applied to the IR-R-G channels of the image, and another FCN is applied to the DSM, resulting in a huge hybrid architecture. An ordinary version (with upsampling at the end) and a *dilation* version are considered ('VGG pretr.' and 'VGG+Dilation' in Table 5.5, respectively). In the latter version, the dilation strategy could only be applied partially as it is too memory intensive. While MLP outperforms the non-pretrained simpler dilation

Table 5.4: Comparison of MLP with other methods on the Vaihingen validation set.

	Imp. surf.	Build.	Low veg.	Tree	Car	F1	Acc.
CNN+RF [146]	88.58	94.23	76.58	86.29	67.58	82.65	86.52
CNN+RF+CRF [146]	89.10	94.30	77.36	86.25	71.91	83.78	86.89
Deconvolution [198]						83.58	87.83
Dilation [163]	90.19	94.49	77.69	87.24	76.77	85.28	87.70
Dilation + CRF [163]	90.41	94.73	78.25	87.25	75.57	85.24	87.90
MLP	**91.69**	**95.24**	**79.44**	**88.12**	**78.42**	**86.58**	**88.92**

Table 5.5: Comparison of MLP with other methods on the Potsdam validation set.

	Imp. surf.	Build.	Low veg.	Tree	Car	Clutter	F1	Acc.
Dilation [163]	86.52	90.78	83.01	78.41	90.42	68.67	82.94	84.14
VGG pretr. [163]	89.84	93.80	85.43	83.61	88.00	74.48	85.86	87.42
VGG+Dilation [163]	**89.95**	93.73	**85.91**	**83.86**	**94.31**	74.62	**87.06**	**87.69**
MLP	89.31	**94.37**	84.83	81.10	93.56	**76.54**	86.62	87.02

network, the *VGG+Dilation* variants exhibits the best overall performance (though not on all of the individual classes). This suggests that the VGG component might be adding a competitive edge, though the authors stated that this is not the case on the Vaihingen dataset.

Overall, MLP provides better accuracies than most techniques presented in the literature, including dilation networks, ensemble approaches and CRF post-processing.

Submission to the ISPRS challenge We submitted the result of executing MLP on the Vaihingen test set to the ISPRS server (ID: 'INR'), which can be accessed online [85]. Our method scored second out of 29 methods, with an overall accuracy of 89.5%. Note that our MLP technique is very simple compared to other methods in the leaderboard, yet it scored better than them. For example, an ensemble of two *skip* CNNs was pretrained on large natural image databases [133], with over 20 convolutional layers and separate paths for the image and the DSM. Despite being simpler, our MLP network outperforms it in the benchmark.

Visual results

Fig. 5.9 shows visual comparisons on closeups of classified images of both datasets. As expected, the base FCN tends to output "blobby" objects, while the other methods provide sharper results. This is particularly noticeable for the cars of Rows 2, 5 and 6, and for the thin road at the lower left corner of Row 4. We also observe that the incorporation of reasoning at lower resolutions allows the derived networks to discover small objects that are otherwise lost. This is particularly noticeable in the 4th row, where

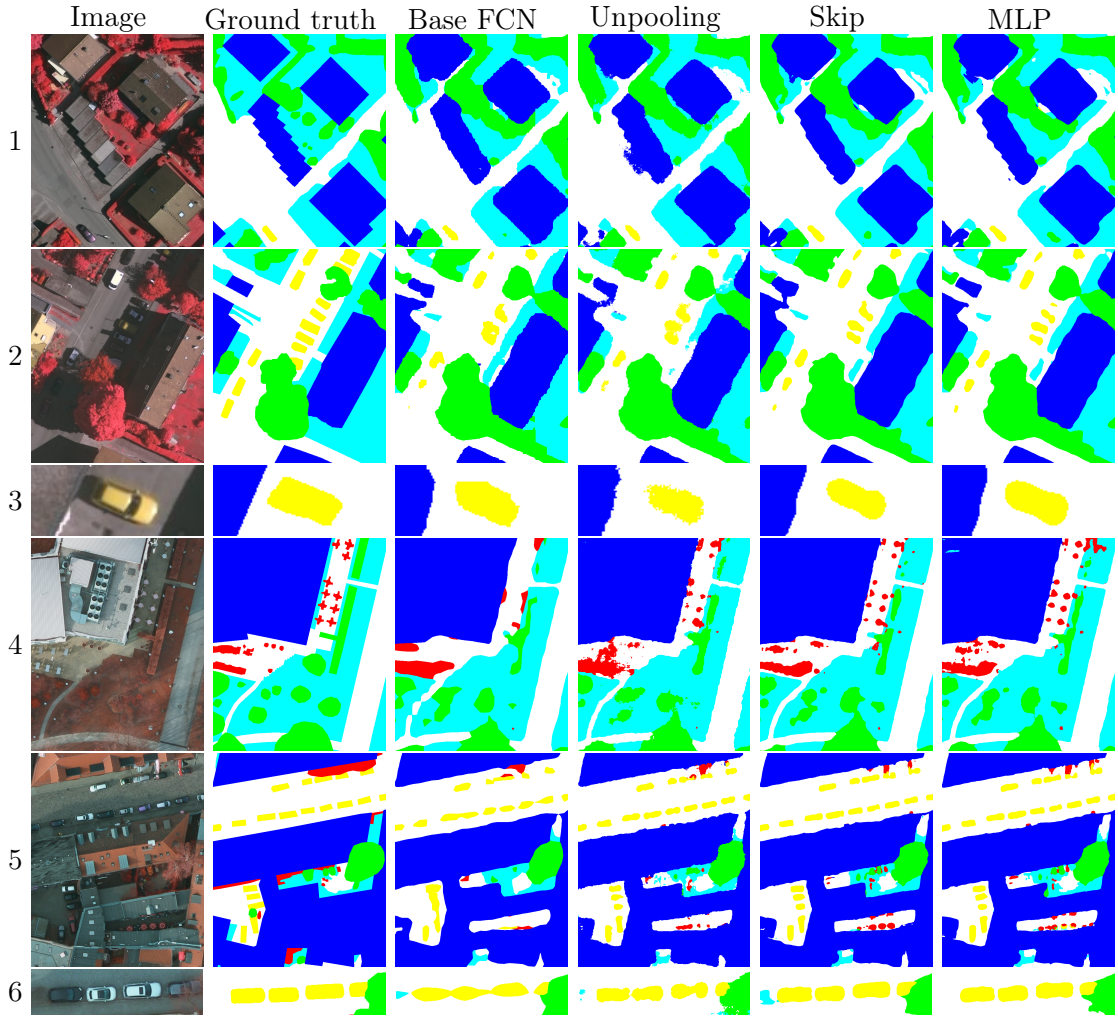


Figure 5.9: Classification of closeups of Vahingen (1–3) and Potsdam (4–6) validation sets. Classes: Impervious surface (white), Building (blue), Low veget. (cyan), Tree (green), Car (yellow), Clutter (red).

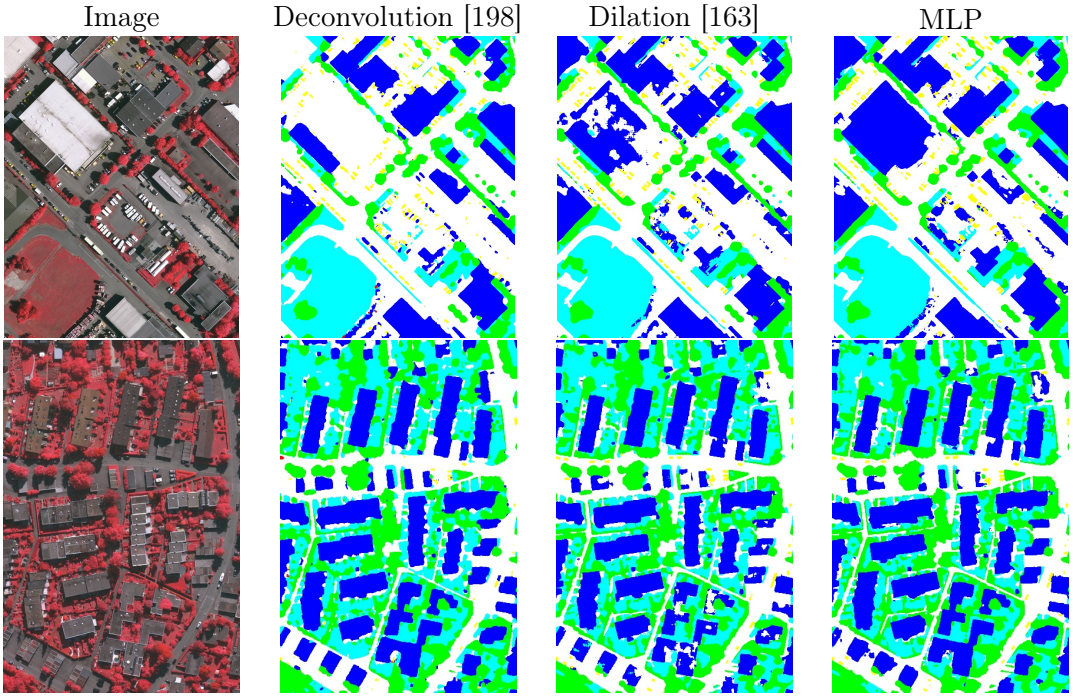


Figure 5.10: Classification of entire tiles of the Vaihingen test set.

there is a set of small round/cross-shaped objects of the *clutter* class (in red) that are omitted or grouped together by the base FCN.

The unpooling technique seems to be prone to outputting artifacts. These are often very small in size, even isolated pixels. This is well observed for example for the car of Row 3. This effect could be a natural consequence of the max unpooling mechanism, as depicted in Fig. 5.6, which upsamples into sparse matrices and delegates the task of reconstructing a smoother output to the deconvolutional layers.

At first sight it is more challenging to visually assess why MLP outperforms the skip network in almost every case in the numerical evaluation. Taking a closer look we can however observe that boundaries tend to be more accurate at a fine level in the case of MLP. For example, the "staircase" shape of one of the buildings in Row 1 is noticeably better outlined by the MLP network.

We can also observe that the ground truth itself is often not very precise. For example, the car in Row 3 does not seem to be labeled accurately, hence it is difficult to imagine that a network would learn to finely label that class. In Row 5, an entire lightwell between buildings has apparently been omitted in the ground truth (labeled as part of the building), yet recognized as an impervious surface by the CNNs.

Table 5.6: Execution times.

	Train [s]		Test [s/ha]	
	Vaih.	Pots.	Vaih.	Pots.
Base FCN	3.9	9.8	0.81	1.44
Unpooling	8.4	21.0	1.38	1.84
Skip	6.6	16.9	0.81	1.48
MLP	10.0	24.5	1.70	2.0
Dilation*	62	400	4.81	17.2

*As reported in [163] (see details in Sec. 5.1.4.)

The general recognition capabilities of CNNs can also be well appreciated in these fragments. For example, in Row 4, while there are tiny round objects both on the roof of the building and outside the building, CNNs correctly label as *building* the ones on the roof and as *clutter* the other ones.

In Fig. 5.10 the classification of entire tiles of the Vaihingen set are shown, obtained from the test set submissions. We include the *deconvolution* [198] and *dilation* [163] network results, together with our MLP. We can see, for instance, that a large white building in the first image is recognized by MLP but misclassified or only partially recovered by the other methods. In the second tile, the Dilation method outputs some holes in the buildings which are not present in the MLP results. A better combination of the information coming from different resolutions might explain why MLP successfully recognizes that these entire surfaces do belong to the same object.

Running times

Table 5.6 reports the running times for training and testing on both datasets. The training time of the architectures derived from the base FCN comprises the time to pretrain the base FCN first and the time to then train the whole system altogether (see details in Sec. 5.1.4). The architectures were implemented using Caffe [89] and run on an Intel I7 CPU @ 2.7Ghz with a Quadro K3100M GPU (4 GB RAM). We also add for comparison the results reported by the author of the Dilation network [163], run on a larger 12 GB RAM GPU. To classify large images we crop them into tiles with as much overlap as the amount of padding in the network, to avoid tile border effects.

As reported in the table, the unpooling, skip and MLP networks introduce an overhead to the base FCN. MLP is the slowest of the derived networks, followed by the unpooling and skip networks. MLP, which provides the highest accuracy, classifies the entire Vaihingen validation set in about 30 seconds and the Postdam validation set in 2 minutes. This is substantially faster than the dilation network. Incorporating the principle of Fig. 5.3 allows us to better allocate computational resources, not spending too

much time and space in conducting a high-resolution analysis where it is not needed, boosting accuracy and performance.

5.1.5 Can classification methods generalize to any city?

Over the last few years, there has been a growing interest in processing remote sensing imagery at a global scale, often the entire Earth at once [135]. New perspectives in remote sensing have particularly highlighted this interest, such as the use of aerial imagery for autonomous driving [135]. The improvements in the algorithms, and the use of clusters and GPUs have made the processing time less of a constraint. One of the current challenges is to design methods that generalize to different areas of the earth, considering the important intra-class variability encountered over large geographic extents.

The standard way of evaluating and comparing classification methods is to split the labeled data into two sets: one used for training and the other one for testing. For example, in the hyperspectral literature it is particularly common to randomly extract certain pixels from the labeled data and use them for training (ranging from as little as 50 pixels [49] to as much as 20% of all the labeled data [191]), while the rest is used for testing. The Pavia and Indian pines datasets [49] have become the standard benchmarks in the hyperspectral literature. They are mostly geared at distinguishing materials (e.g., *bitumen building* and *bricks*), thus leveraging the properties of hyperspectral imagery. However, those images cover limited geographic areas and the evaluation procedure does not assess how the methods generalize to different contexts or more abstract semantic classes.

With the goal of comparing classification methods over large areas, Mnih [140] created building and road classification datasets over Massachusetts, covering 340 km^2 and 2600 km^2 respectively. For testing, several randomly selected tiles were removed from the reference data. The training set thus covers a geographic surface with "holes", which are used for testing. This situation is analogous to the procedure used for the aforementioned hyperspectral datasets, though taken to a larger scale. While the Massachusetts datasets indeed cover a large surface with significant intra-class variability, the image tiles tend to be self-similar and with uniform color histograms. As shown in [140], a CNN trained on the Massachusetts dataset generalizes poorly to images over Buffalo, and a fine-tuning of the CNN to the new dataset is required.

In the context of high-resolution image classification, the Vaihingen and Potsdam datasets [198] have gained increasing attention over the last year, as discussed previously in this chapter. While they provide exhaustive reference data with multiple object classes,

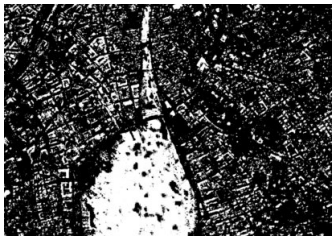


Figure 5.11: A CNN trained on a different dataset misclassifies most of Lake Zurich as a building.

the area covered is limited (roughly 1.5 km^2 and 3.5 km^2 respectively). The Bavaria and Aerial KITTI datasets [135], used for road labeling, also cover small surfaces (5 km^2 and 6 km^2, respectively).

In our experience, and in accordance to [140], training a classifier with images over a particular region and illumination conditions tends to generalize poorly to other images. For example, Fig. 5.11 depicts a classification map over Zurich into the *building/not building* classes, created by using the FCN trained over Forez, France. We can observe Lake Zurich being mostly classified as *building*. Even though there were buildings and body waters in the French imagery, the CNN seems to have learned what a building looks like in that particular images and not simply what a building looks like.

Our goal is to provide a common framework to evaluate classification techniques and, in particular, their generalization capabilities. We created a benchmark database of labeled imagery that covers varied urban landscapes, ranging from highly dense metropolitan financial districts to alpine resorts. The data, referred to as the *Inria Aerial Image Labeling Dataset*[2], includes urban settlements over the United States and Austria, and is labeled into *building* and *not building* classes. Contrary to all previous datasets, the training and test sets are split by city instead of excluding random pixels or tiles. This way, a system trained, for example, on Chicago, is expected to classify imagery over San Francisco (with a significantly different appearance). The test set reference data is not publicly released, and a contest has been launched for researchers to submit their results. In the following, we describe the dataset and then assess the performance of the networks presented previously in this chapter.

[2]project.inria.fr/aerialimagelabeling

The dataset

One of the first key points to decide when creating the dataset was which geographic areas to include and which semantic classes to consider. The criteria were as follows:

- Recent orthorectified imagery available;
- Recent official cadastral records available;
- Precise registration between the cadastral records and the orthorectified imagery;
- Open-access data, both for the images and the cadaster (free to access and distribute);
- Cover varied urban landscapes and illumination.

Let us first highlight the fact that we can only focus on regions where both the images and the reference data are available. In addition, we require the data to be open access in order to freely share our derived dataset with the community. After extensive research, we found that certain US and Austrian areas satisfy those requirements. In the case of the US, public domain orthoimages have been released by USGS through the National Map service (nationalmap.gov) in most urban areas of the country. Vectorial cadastral records have been released through certain local or statewide geographic information system (GIS) websites. We must focus on the zones where such reference data are available in addition to the images.

In the case of Austria, the different provinces have shared images through their respective GIS agencies. We focus, in particular, on Tyrol and Vienna provinces, since open vectorial cadastral data are also on hand. We obtained the images through the WMS services provided by the GIS departments[3] as well as the associated reference shapefiles.

The original US imagery is provided at either 15 or 30 cm resolution with three or four spectral bands (RGB/RGB-Infrared), depending on the area, and Vienna imagery contains three bands (RGB) at a resolution of 10 or 20 cm. We took out the common factor and built our dataset with 30 cm images (average resampling if needed) and using the three color bands.

We consider two semantic classes: *building* and *not building*. For this we must extract the so-called *building footprints* from the cadaster. While there are other classes present in some areas (e.g., trees and roads), the building class is the only one that is consistent across different areas. Roads, for example, are often represented with a line, but it is very often not located at the center of the road and its width is usually not specified.

[3] https://gis.tirol.gv.at/arcgis/services/Service_Public/orthofoto/MapServer/WMSServer; http://maps.wien.gv.at/wmts/1.0.0/WMTSCapabilities.xml

Train	Tiles*	Total area	**Test**	Tiles*	Total area
Austin, TX	36	81 km^2	Bellingham, WA	36	81 km^2
Chicago, IL	36	81 km^2	San Francisco, CA	36	81 km^2
Kitsap County, WA	36	81 km^2	Bloomington, IN	36	81 km^2
Vienna, Austria	36	81 km^2	Innsbruck, Austria	36	81 km^2
West Tyrol, Austria	36	81 km^2	East Tyrol, Austria	36	81 km^2
Total	180	405 km^2	Total	180	405 km^2

Table 5.7: Dataset statistics. *Tile size: 1500^2 px. (0.3 m resolution).

This makes it difficult to derive a pixelwise semantic labeling for roads, and is an active research problem itself [135].

Once we selected a number of candidate areas for the dataset, we visually inspected them to assess whether the cadaster is properly aligned with the images. In some regions, there are irregular shifts that led us to exclude them (e.g., Seattle and Spokane cities). This may be the result of errors or imprecision in the terrain model used to orthorectify the images, or in the digitization of the cadaster. Note that we have only considered official image and cadaster data sources, ignoring, e.g., OpenStreetMap (OSM) data. Curiously enough, while we find the official San Francisco building footprints to be perfectly aligned with the USGS imagery, a team of OSM collaborators manually modified 150,000 buildings from these footprints prior to their inclusion in OSM[4], arguing that they were inaccurate. We now observe them to be misaligned with the imagery. A possible explanation for this is that, at the time of the edit, the Bing images used as the base layer of the OSM editor may have not been geographically precise.

The regions included in the dataset and their distribution into training and test subsets are depicted in Table 5.7. Note first that the amount of data in each of the subsets is the same. This stresses our goal of properly assessing classification methods that generalize to different areas and images. The regions were split in such a way that each of the subsets contains both European and American landscapes, as well as high-density (e.g., Chicago/San Francisco and Vienna/Innsbruck) and low-density (e.g., Kistap/Bloomington, West/East Tyrol) urban settlements. While aerial images over Tyrol are present in both subsets, they have been obtained at different flights over the country, thus exhibiting different illumination characteristics. We have also selected dissimilar images inside some of the groups (e.g., Kitsap County contains tiles from two different flights with very dissimilar characteristics). The reference data was created by rasterizing the shapefiles with GDAL. Fig. 5.12 shows closeups of the images in the dataset.

[4]https://www.mapbox.com/blog/status-san-francisco-complete/

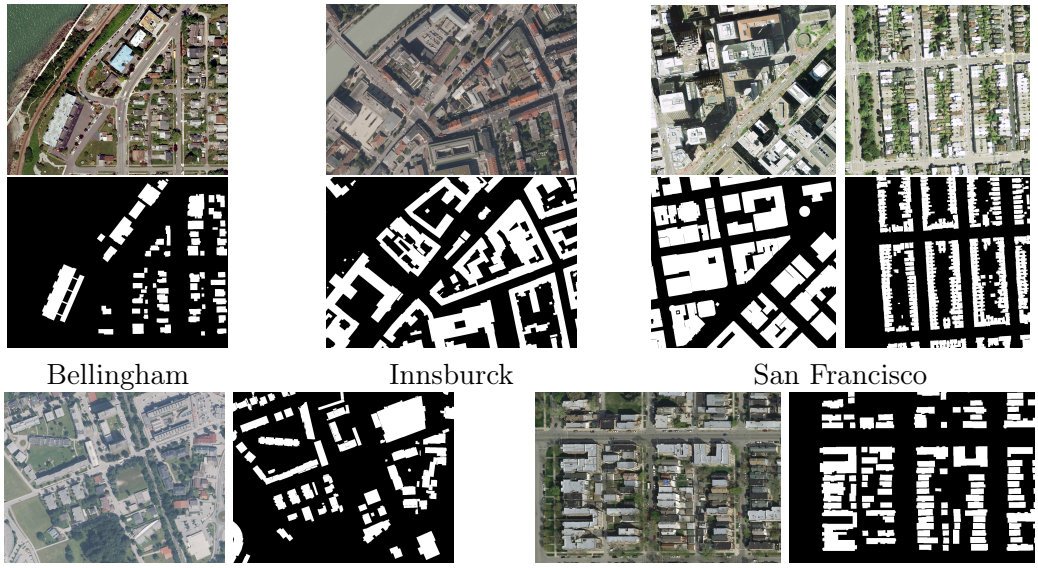


Figure 5.12: Close-ups of the dataset images and their corresponding reference data.

We consider two evaluation measures to assess the performance of different methods on the dataset: the accuracy and the intersection over union (IoU) of the positive (*building*) class. We compute accuracy and IoU on the overall dataset and for every region independently (e.g., San Francisco).

Experiments

We experimented with convolutional neural networks on the dataset. We created a validation set by excluding the first five tiles of each area from the training set (e.g., Austin{1-5}). We first trained the base fully convolutional network (FCN) proposed from Potsdam dataset in Section 5.1.4, for 120,000 iterations on randomly sampled patches of our dataset (momentum is set to 0.9, the L2 penalty to 0.0005 and the learning rate to 0.001). To provide a finer classification, we derived an MLP network on top of the base FCN, as explained in Section 5.1.3 and illustrated in Fig. 5.8. The pretrained FCN was used to initialize the corresponding parameters in the MLP network, and then the overall system was trained for an extra 250,000 iterations, which took 50 hours on a single GPU. We started with a learning rate of 0.0001, multiplying it by 0.1 every 50k iterations.

The numerical results are summarized in Tables 5.8 and 5.9, for the validation and test sets, respectively. We also include the performance of a *skip* network as an alternative way of combining features to refine the predictions of the coarse base FCN (see Section 5.1.2).

Table 5.8: Numerical evaluation on small validation set.

		Austin	Chicago	Kitsap Co.	West Tyrol	Vienna	Overall
FCN	IoU	47.66	53.62	33.70	46.86	60.60	53.82
	Acc.	92.22	88.59	98.58	95.83	88.72	92.79
Skip	IoU	57.87	61.13	46.43	54.91	70.51	62.97
	Acc.	93.85	90.54	98.84	96.47	91.48	94.24
MLP	IoU	61.20	61.30	51.50	57.95	72.13	64.67
	Acc.	94.20	90.43	98.92	96.66	91.87	94.42

Table 5.9: Numerical evaluation on test set.

		Bellingham	Bloomington	Innsbruck	S. Francisco	East Tyrol	Overall
FCN	IoU	44.83	35.38	36.50	44.92	43.69	42.19
	Acc.	94.48	94.07	92.97	82.60	95.14	91.85
Skip	IoU	52.91	46.08	58.12	57.84	59.03	55.82
	Acc.	95.14	94.95	95.16	86.05	96.40	93.54
MLP	IoU	56.11	50.40	61.03	61.38	62.51	59.31
	Acc.	95.37	95.27	95.37	87.00	96.61	93.93

Fig. 5.13 depicts close-ups of the classification on the test set, i.e., on regions never "seen" by the neural network at training time. While the FCN produces fuzzy results, it successfully identifies buildings in varied images. The MLP network provides finer outputs, as confirmed both numerically and visually.

The MLP network reaches about 60% IoU on the entire test set. This means that the output objects overlap the real ones by 60%, as assessed over a significant amount of test data. While there is certainly room for improvement, these values suggest that the current network does generalize well to different cities.

5.1.6 Concluding remarks

In this section, I have presented an overview of different families of dense classification convolutional neural network (CNN) prototypes. We have studied which relevant constraints can be imposed in the architecture by construction, reducing the number of parameters and improving the optimization. We observed that existing networks often spend efforts in learning invariances that could be otherwise guaranteed, and reason at a high resolution even when it is not needed. While previous methods are already competitive, we can devise more optimal approaches.

We derived a model in which spatial features are learned at multiple resolutions (and thus different levels of detail) and a specific CNN module learns how to combine them. In our experiments on aerial imagery, such a model proved to be more effective than the other approaches to conduct high-resolution labeling. It provides a better

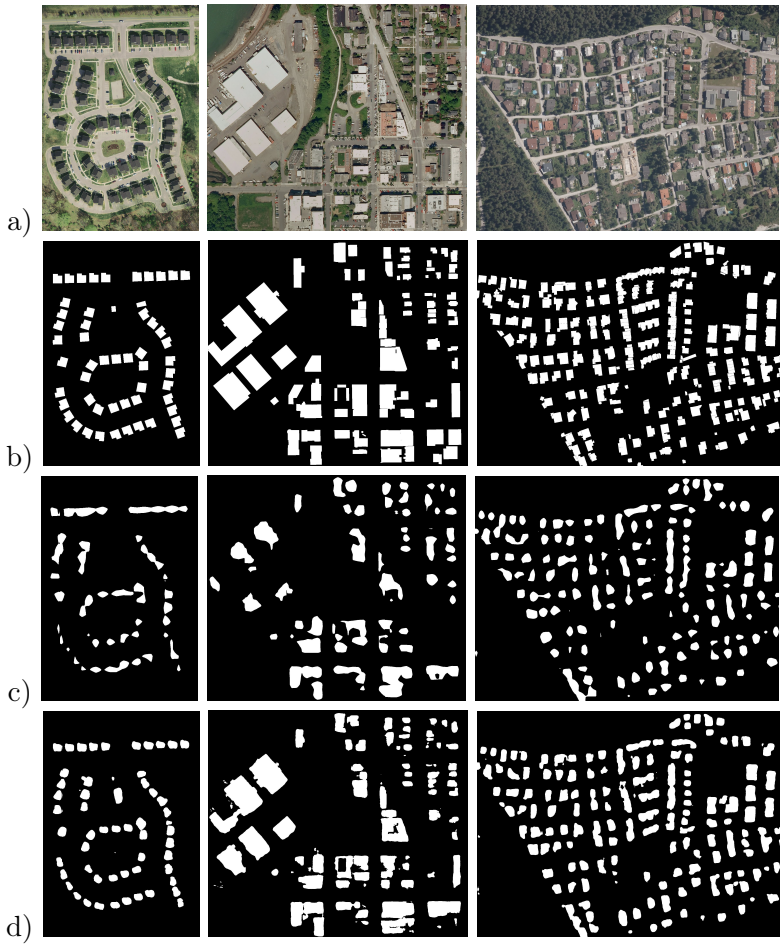


Figure 5.13: Visual close-ups on test set. (a) Color input. (b) Reference data. (c) FCN results. (d) MLP results.

accuracy with low computational requirements, leading to a win-win situation. Some of the outperformed methods are in fact significantly more complex than our approach, proving once again that striving for simplicity is often the way to go when using CNN architectures.

We also created a dataset for the classification of aerial images. This dataset highlights the need for methods that generalize to the dissimilar appearance of urban settlements around the Earth. Contrary to previous work, the testing is not performed over excluded areas of the training surface, but over entirely different cities instead. We cover a wide range of urban densities, on both European and American cities.

Our experiments with deep neural networks show their satisfactory generalization capability. However, there is still a large room for improvement, as observed in the numerical results. In the following, I present a method, designed with the goal to improve classification results, when there is a significant lack of spatially accurate reference data to train the classification system.

5.2 Recurrent neural networks to correct classification maps

In this section, we explore incorporating image information a posteriori in an enhancement module that sharpens the coarse classification maps around objects. The pioneering approach in this direction, called Deeplab [27], uses a fully connected conditional random field (CRF) to perform the enhancement. Zheng et al. [214] recently reformulated the fully connected CRF of Deeplab as an RNN, and Chen et al. [26] designed an RNN that emulates the domain transform filter [58]. Such a filter is used to sharpen the classification maps around image edges, which are themselves detected with a CNN. In these methods the refinement algorithm is designed beforehand and only few parameters that rule the algorithm are learned as part of the network's parameters. The innovating aspect of these approaches is that both steps (coarse classification and enhancement) can be seen as a single end-to-end network and optimized simultaneously.

Instead of predefining the algorithmic details as in previous works, we formulate a general iterative refinement algorithm through an RNN and let the network learn the specific algorithm. To our knowledge, little work has explored the idea of learning an iterative algorithm. In the context of image restoration, the preliminary work by Liu et al. [122, 123] proposed to optimize the coefficients of a linear combination of predefined terms. Chen et al. [28] later modeled this problem as a diffusion process and used an RNN to learn the linear filters involved as well as the coefficients of a parametrized non-linear function. Our problem is however different, in that we use the image as guidance

(a) Color image (b) CNN heat map (c) Ground truth

Figure 5.14: Sample classification of buildings with a CNN.

to update a classification map, and not to restore the image itself. Besides, while we drew inspiration on diffusion processes, we are also interested in imitating other iterative processes like active contours, thus we do not restrict our system to diffusions but consider all PDEs.

5.2.1 Learning iterative processes to enhance classification

Let us assume we are given a set of score (or "heat") maps u_k, one for each possible class $k \in \Omega$, in a pixelwise labeling problem. The score of a pixel reflects the likelihood of belonging to a class, according to the classifier's predictions. The final class assigned to every pixel is the one with maximal value u_k. Alternatively, a softmax function can be used to interpret the results as probability scores: $P(k) = e^{u_k} / \sum_{j \in \Omega} e^{u_j}$, as usually done in CNN literature. Fig. 5.14 shows a sample of the type of fuzzy heat map outputted by a CNN for the class 'building'.

Our goal is to combine the score maps u_k with information derived from the input image (e.g., edge features) to sharpen the scores near the real objects in order to enhance the classification. One way to perform such a task is to progressively enhance the score maps by using partial differential equations (PDEs). We first describe different types of PDEs we could certainly imagine to design in order to solve our problem. Instead of discussing which one is the best, we later propose a generic iterative process to enhance the classification maps without specific constraints on the algorithm rationale. Finally, we show how this equation can be expressed and trained as a recurrent neural network (RNN).

Partial differential equations (PDEs)

We can formulate a variety of diffusion processes applied to the maps u_k as partial differential equations. For example, the heat flow is described as:

$$\frac{\partial u_k(x)}{\partial t} = \mathrm{div}(\nabla u_k(x)), \tag{5.5}$$

where $\mathrm{div}(\cdot)$ denotes the divergence operator in the spatial domain of x. Applying such a diffusion process in our context would smooth out the heat maps. Instead, our goal is to design an image-dependent smoothing process that aligns the heat maps to the image features. A natural way of achieving this is to modulate the gradient in Eq. 5.5 by a scalar function $g(x, I)$ that depends on the input image I:

$$\frac{\partial u_k(x)}{\partial t} = \mathrm{div}(g(I, x)\nabla u_k(x)). \tag{5.6}$$

Eq. 5.6 is similar to the Perona-Malik diffusion [151] with the exception that Perona-Malik uses the smoothed function itself to guide the diffusion. $g(I, x)$ denotes an edge-stopping function that takes low values near borders of $I(x)$ in order to slow down the smoothing process there.

Another possibility would be to consider a more general variant in which $g(I, x)$ is replaced by a matrix $D(I, x)$, acting as a diffusion tensor that redirects the flow based on image properties instead of just slowing it down near edges:

$$\frac{\partial u_k(x)}{\partial t} = \mathrm{div}(D(I, x)\nabla u_k(x)). \tag{5.7}$$

This formulation is related to the so-called anisotropic diffusion process [201].

Alternatively, one can draw inspiration from the level set framework. For example, the geodesic active contours technique formulated as level sets translates into:

$$\frac{\partial u_k(x)}{\partial t} = |\nabla u_k(x)|\mathrm{div}\left(g(I, x)\frac{\nabla u_k(x)}{|\nabla u_k(x)|}\right). \tag{5.8}$$

Such a formulation favors the zero level set to align with minima of $g(I, x)$ [22]. Schemes based on Eq. 5.8 could then be used to improve heat maps u_k, provided they are scaled so that segmentation boundaries match zero levels.

As shown above, many different PDE approaches can be devised to enhance classification maps. However, several choices must be made to select the appropriate PDE and tailor it to our problem. For example, one must choose the edge-stopping function $g(I, x)$ in Eqs. 5.6, 5.8. Common choices are exponential or rational functions on the image gradient [151], which in turn requires to set an edge-sensitivity parameter. Extensions to the original Perona-Malik approach could also be considered, such as a popular regularized variant that computes the gradient on a Gaussian-smoothed version of the input image

[201]. In the case of opting for anisotropic diffusion, one must design $D(I, x)$.

Instead of using trial and error to perform such design, our goal is to let a machine learning system discover by itself a useful iterative process for our task.

Generic classification enhancement

PDEs are usually discretized in space by using finite differences, which represent derivatives as discrete convolution filters. We build upon this scheme to write a generic discrete formulation of an iterative enhancement process.

Let us consider that we take as input a score map u_k (for class k) and, in the most general case, an arbitrary number of feature maps $\{g_1, ..., g_p\}$ derived from image I. In order to perform differential operations, of the type $\{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial^2}{\partial x \partial y}, \frac{\partial^2}{\partial x^2}, ...\}$, we consider convolution kernels $\{M_1, M_2, ...\}$ and $\{N_1^j, N_2^j, ...\}$ to be applied to the heat map u_k and to the features g_j derived from image I, respectively. While we could certainly directly provide a bank of filters M_i and N_i^j in the form of Sobel operators, Laplacian operators, etc., we may simply let the system learn the required filters. We group all the feature maps that result from applying these convolutions, in a single set:

$$\Phi(u_k, I) = \Big\{ M_i * u_k, \;\; N_l^j * g_j(I) \; ; \;\; \forall i, j, l \Big\}. \tag{5.9}$$

Let us now define a generic discretized scheme as:

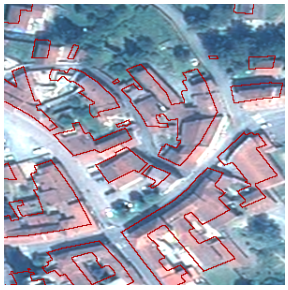
$$\frac{\partial u_k(x)}{\partial t} = f_k\big(\; \Phi(u_k, I)(x) \; \big), \tag{5.10}$$

where f_k is a function that takes as input the values of all the features in $\Phi(u_k, I)$ at an image point x, and combines them. While convolutions M_i and N_i^j convey the "spatial" reasoning, e.g., gradients, f_k captures the combination of these elements, such as the products in Eqs. 5.6 and 5.8.

Instead of deriving an arbitrary number of possibly complex features $N_i^j * g_j(I)$ from image I, we can think of a simplified scheme in which we directly operate on I, by considering only convolutions: $N_i * I$. The list of functionals considered in Eq. 5.10 is then

$$\Phi(u_k, I) = \Big\{ M_i * u_k, \;\; N_j * I \; ; \;\; \forall i, j \Big\} \tag{5.11}$$

and consists only of convolutional kernels directly applied to the heat maps u_k and to the image I. From now on, we here stick to this simpler formulation, yet we acknowledge that it may be eventually useful to work on a higher-level representation rather than on the input image itself. Note that if one restricts functions f_k in Eq. 5.10 to be linear,



(a) OpenStreetMap

(b) Manual labeling

Figure 5.15: Samples of reference data for the *building* class. Imprecise OpenStreetMap data vs manually labeled data.

we still obtain the set of all linear PDEs. We consider *any* function f_k, thus introducing non-linearities.

PDEs are usually discretized in time, taking the form:

$$u_{k,t+1}(x) = u_{k,t}(x) + \delta u_{k,t}(x), \tag{5.12}$$

where $\delta u_{k,t}$ denotes the overall update of $u_{k,t}$ at time t.

Note that the convolution filters in Eqs. 5.9 and 5.11 are class-agnostic: M_i, N_j and N_l^j do not depend on k, while f_k may be a different function for each class k. Function f_k thus determines the contribution of each feature to the equation, contemplating the case in which a different evolution might be optimal for each of the classes, even if just in terms of a time-step factor. In the following, we detail a way to learn the update functions $\delta u_{k,t}$ from training data.

Iterative processes as RNNs

We now show that the generic iterative process can be implemented as an RNN, and thus trained from labeled data. This stage requires to provide the system with a piece of accurately labeled ground truth (see e.g., Fig. 5.15).

Let us first show that one iteration, as defined in Eqs. 5.10-5.12, can be expressed in terms of common neural network layers. Let us focus on a single pixel for a specific class, simplifying the notation from $u_{k,t}(x)$ to u_t. Fig. 5.16 illustrates the proposed network architecture. Each iteration takes as input the image I and a given heat map u_t to enhance at time t. In the first iteration, u_t is the initial coarse heat map to be improved, outputted by another pre-trained neural network in our case. From the heat map u_t we derive a series of filter responses, which correspond to $M_i * u_t$ in Eq. 5.11. These

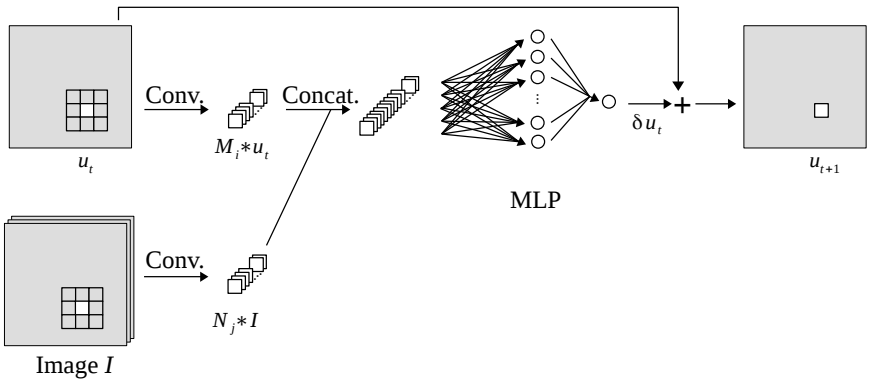


Figure 5.16: One enhancement iteration represented as common neural network layers.

responses are found by computing the dot product between a set of filters M_i and the values of $u_{k,t}(\cdot)$ in a spatial neighborhood of a given point. Analogously, a set of filter responses are computed at the same spatial location on the input image, corresponding to the different $N_j * I$ of Eq. 5.11. These operations are convolutions when performed densely in space, $N_j * I$ and $M_i * u_t$ being feature maps of the filter responses.

These filters are then "concatenated", forming the pool of features Φ coming from both the input image and the heat map, as in Eq. 5.11, and inputted to f_k in Eq. 5.10. We must now learn the function δu_t that describes how the heat map u_t is updated at iteration t (see Eq. 5.12), based on these features.

Eq. 5.10 does not introduce specifics about function f_k. In (5.5)-(5.8), for example, it includes products between different terms, but we could certainly imagine other functions. We therefore model δu_t through a multilayer perceptron (MLP), because it can approximate any function within a bounded error [14]. We include one hidden layer with nonlinear activation functions followed by an output neuron with a linear activation (a typical configuration for regression problems), although other MLP architectures could be used. Applying this MLP densely is equivalent to performing convolutions with 1×1 kernels at every layer. The implementation to densely label entire images is then straightforward.

The value of δu_t is then added to u_t in order to generate the updated map u_{t+1} (see Fig. 5.16). This addition is performed pixel by pixel in the case of a dense input. Note that although we could have removed this addition and let the MLP directly output the updated map u_{t+1}, we opted for this architecture since it is more closely related to the equations and better conveys the intention of a progressive refinement of the classification map. Moreover, learning δu_t instead of u_{t+1} has a significant advantage at

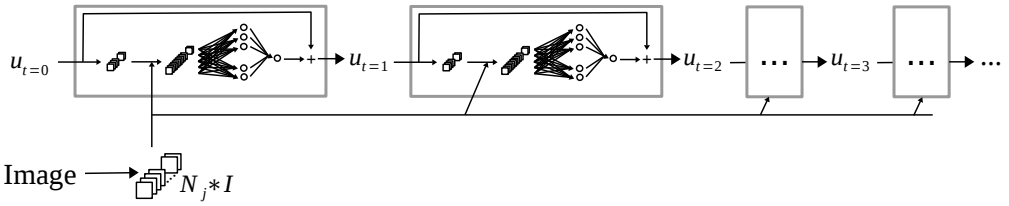


Figure 5.17: Modules of Fig. 5.16 are stacked (while sharing parameters) to implement an RNN.

training time: a random initialization of the networks' parameters centered around zero means that the initial RNN represents an iterative process close to the identity (with some noise). Training uses the asymmetry induced by this noise to progressively move from the identity to a more useful iterative process.

The overall iterative process is implemented by unrolling a finite number of iterations, as illustrated in Fig. 5.17, under the constraint that the parameters are shared among all iterations. Such a sharing is enforced at training time by a simple modification to the back-propagation training algorithm where the derivatives of every instance of a weight at different iterations are averaged [203]. Note that the spatial features are shared across the classes, while a different MLP is learned for each of them, following Eq. 5.10. As depicted by Fig. 5.17 and conveyed in the equations, the features extracted from the input image are independent of the iteration.

The RNN of Fig. 5.17 represents a dynamical system that iteratively improves the class heat maps. Training such an RNN amounts to finding the optimal dynamical system for our enhancement task.

5.2.2 Experimental results

Implementation details

We first describe the CNN used to produce the coarse predictions, then detail the proposed RNN. The employed CNN is based on a previous remote sensing network presented by Mnih [140]. We have designed a fully convolutional [124] version of Mnih's network, since recent remote sensing work has shown the theoretical and practical advantages of this type of architecture [92, 130]. The CNN takes 3-band color image patches at 1m^2 resolution and produces as many heat maps as classes considered. The resulting four-layer architecture is as follows: 64 conv. filters (12×12, stride 4) $\rightarrow$ 128 conv. filters (3×3) $\rightarrow$ 128 conv. filters (3×3) $\rightarrow$ 3 conv. filters (9×9). Since the first convolution is performed with a stride of 4, the resulting feature maps have a quarter of the input

resolution. Therefore, a deconvolutional layer [124] is added on top to upsample the classification maps to the original resolution. The activation functions used in the hidden layers are rectified linear units. This network is trained on patches randomly selected from the training dataset. We group 64 patches with classification maps of size 64×64 into mini-batches (following [140]) to estimate the gradient of the network's parameters and back-propagate them. Our loss function is the cross-entropy between the target and predicted class probabilities. Stochastic gradient descent is used for optimization, with learning rate 0.01, momentum 0.9 and an L2 weight regularization of 0.0002. We did not however optimized these parameters nor the networks' architectures.

We now detail the implementation of the RNN described in Sec. 5.2.1. Let us remark that at this stage we fix the weights of the initial coarse CNN and the manually labeled tile is used to train the RNN only. Our RNN learns 32 M_i and 32 N_j filters, both of spatial dimensions 5×5. An independent MLP is learned for every class, using 32 hidden neurons each and with rectified linear activations, while M_i and N_j filters are shared across the different classes (in accordance to Equations 5.10 and 5.11). This highlights the fact that M_i and N_j capture low-level features while the MLPs convey class-specific behavior. We unroll five RNN iterations, which enables us to significantly improve the classification maps without exhausting our GPU's memory. Training is performed on random patches and with the cross-entropy loss function, as done with the coarse CNN. The employed gradient descent algorithm is AdaGrad [44], which exhibits a faster convergence in our case, using a base learning rate of 0.01 (higher values make the loss diverge). All weights are initialized randomly by sampling from a distribution that depends on the number of neuron inputs [64]. We trained the RNN for 50,000 iterations, until observing convergence of the training loss, which took around four hours on a single GPU.

Dataset and results

We perform our experiments on images acquired by a Pléiades satellite over the area of Forez, France. An RGB color image is used, obtained by pansharpening [200] the satellite data, which provides a spatial resolution of 0.5 m^2. Since the networks described hereabove are designed for 1 m^2 resolution images, we downsample the Pléiades images before feeding them to our networks and bilinearly upsample the outputs.

From this image we selected an area with OpenStreetMap (OSM) [75] coverage to create a 22.5 km^2 training dataset for the classes *building*, *road* and *background*. The reference data was obtained by rasterizing the raw OSM maps. Misregistrations and omissions are present all over the dataset (see, e.g., Fig. 5.15(a)). Buildings tend to be

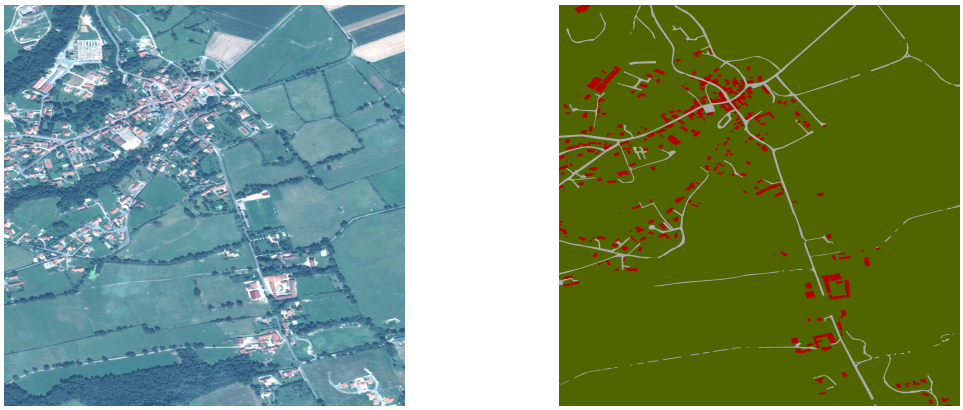


Figure 5.18: Manually labeled tile used to train the RNN for the classification enhancement task.

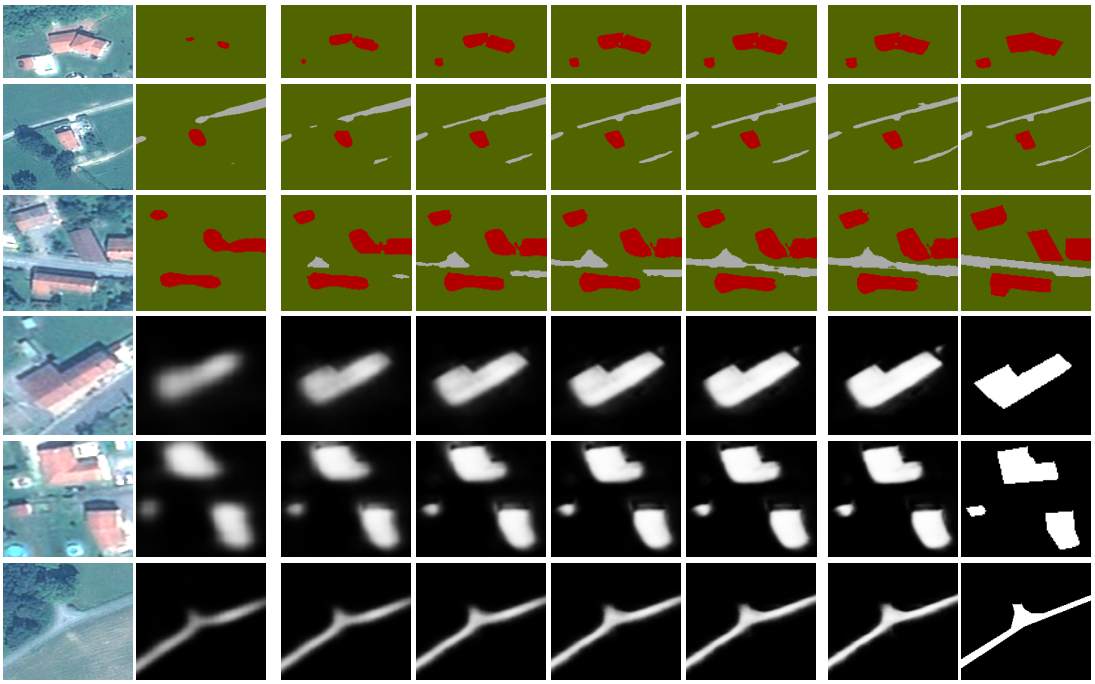


Figure 5.19: Evolution of fragments of classification maps (top rows) and single-class fuzzy scores (bottom rows) through RNN iterations. The classification maps are progressively sharpened around object's edges.

misaligned or omitted, while many roads in the ground truth are not visible in the image (or the other way around). Moreover, since OSM's roads are represented by polylines, we set a fixed road width of 7 m to rasterize this class (following [140]), which makes their classification particularly challenging. This dataset is used to train the initial coarse CNNs.

We manually labeled two 2.25 km^2 tiles to train and test the RNN at enhancing the predictions of the coarse network. We denote them by enhancement and test sets, respectively. Note that our RNN system must discover an algorithm to refine an existing classification map, and not to conduct the classification itself, hence a smaller training set should be sufficient for this stage. The enhancement set is depicted in Fig. 5.18 while the test set is shown in Figs. 5.22(a)/(f).

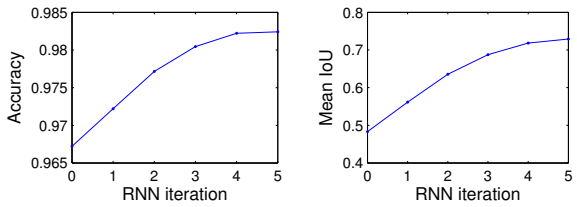
In the following, we report the results obtained by using the proposed method on the Pléiades dataset. Fig. 5.19 provides closeups of results on different fragments of the test dataset. The initial and final maps (before and after the RNN enhancement) are depicted, as well as the intermediate results through the RNN iterations. We show both a set of final classification maps and some single-class fuzzy probability maps. We can observe that as the RNN iterations go by, the classification maps are refined and the objects better align to image edges. The fuzzy probabilities become more confident, sharpening object boundaries. To quantitatively assess this improvement we compute two measures on the test set: the overall accuracy (proportion of correctly classified pixels) and the intersection over union (IoU) [124]. Mean IoU has become the standard in semantic segmentation since it is more reliable in the presence of imbalanced classes (such as *background* class, which is included to compute the mean) [34]. As summarized in the table of Fig. 5.20(a), the performance of the original coarse CNN (denoted by CNN) is significantly improved by attaching our RNN (CNN+RNN). Both measures increase monotonously along the intermediate RNN iterations, as depicted in Fig. 5.20(b).

The initial classification of roads has an overlap of less than 10% with the roads in the ground truth, as shown by its individual IoU. The RNN makes them emerge from the background class, now overlapping the ground truth roads by over 50%. Buildings also become better aligned to the real boundaries, going from less than 40% to over 70% overlap with the ground truth buildings. This constitutes a multiplication of the IoU by a factor of 5 for roads and 2 for buildings, which indicates a significant improvement at outlining and not just detecting objects.

Additional visual fragments before and after the RNN refinement are shown in Fig. 5.21. We can observe in the last row how the iterative process learned by the RNN both thickens and narrows the roads depending on the location.

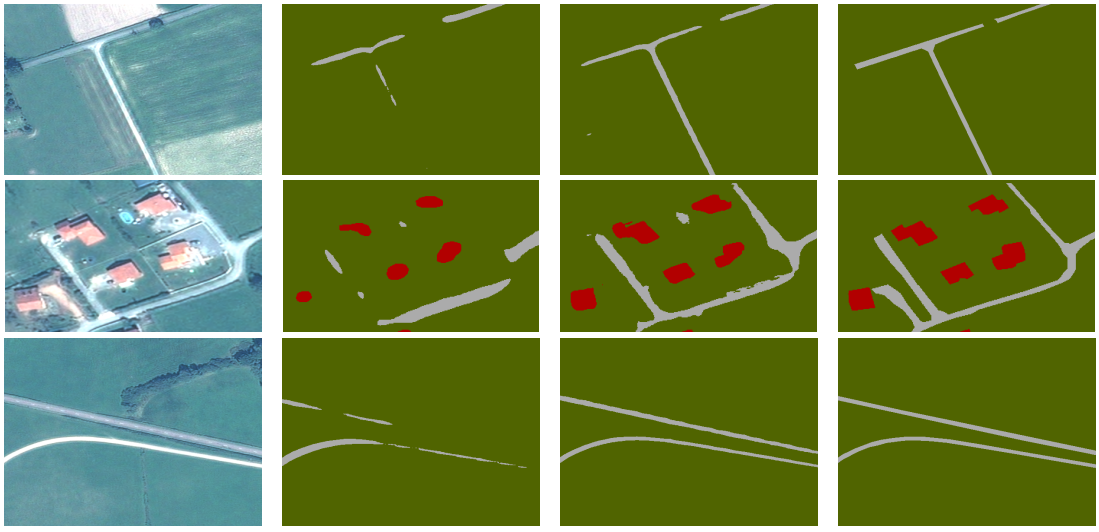
Method	Overall accuracy	Mean IoU	Class-specific IoU Build.	Road	Backg.
CNN	96.72	48.32	38.92	9.34	96.69
CNN+CRF	96.96	44.15	29.05	6.62	96.78
CNN+RNN$^{=}$	97.78	65.30	59.12	39.03	97.74
CNN+RNN	**98.24**	**72.90**	**69.16**	**51.32**	**98.20**

(a) Numerical comparison (in %)



(b) Evolution through RNN iterations

Figure 5.20: Quantitative evaluation on Pléaiades images test set over Forez, France.



Color image | Coarse CNN | RNN output | Ground truth

Figure 5.21: Initial coarse classifications and the enhanced maps by using RNNs.

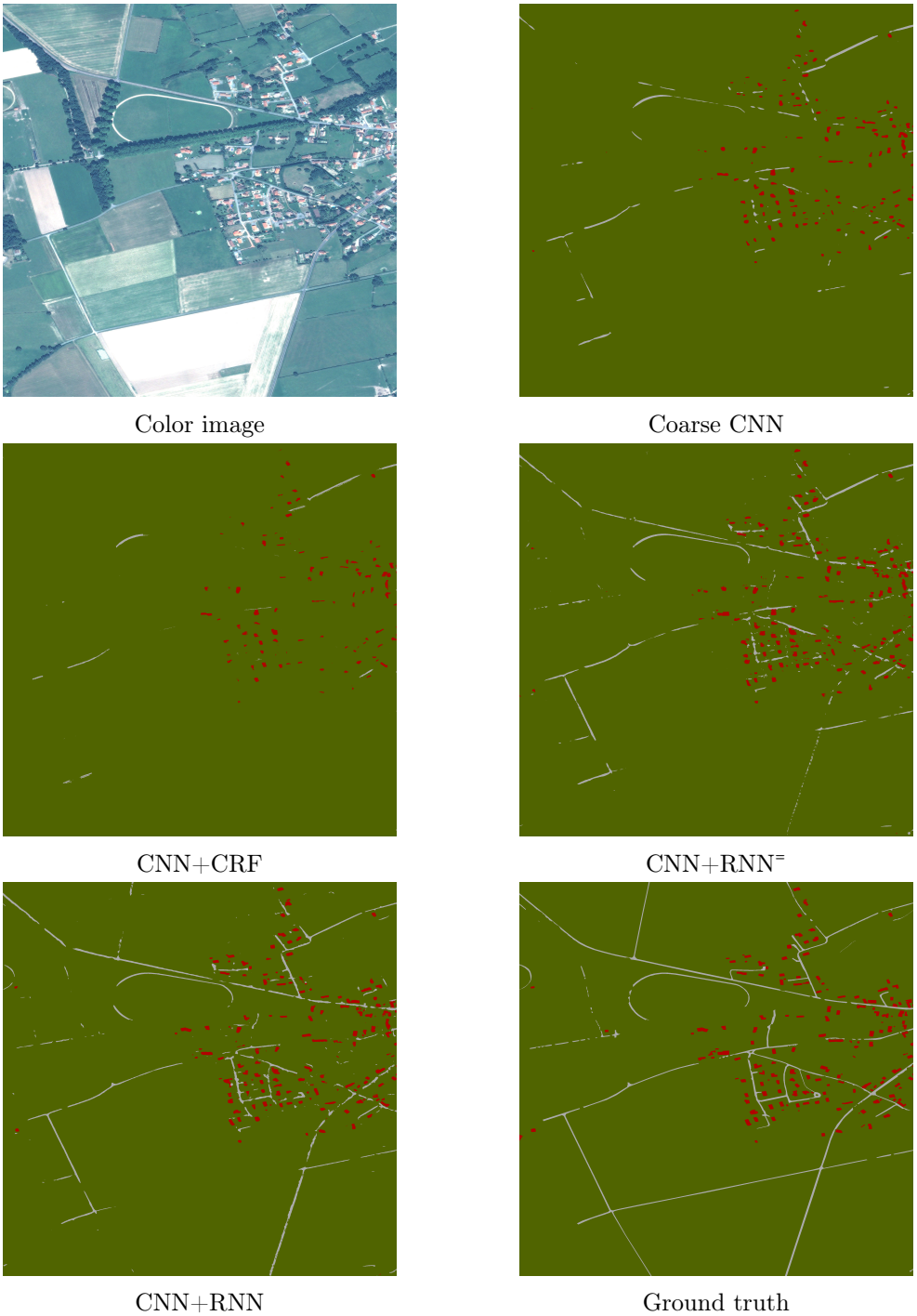


Figure 5.22: Visual comparison on a Pléiades satellite image tile of size 3000×3000 covering 2.25 km^2.

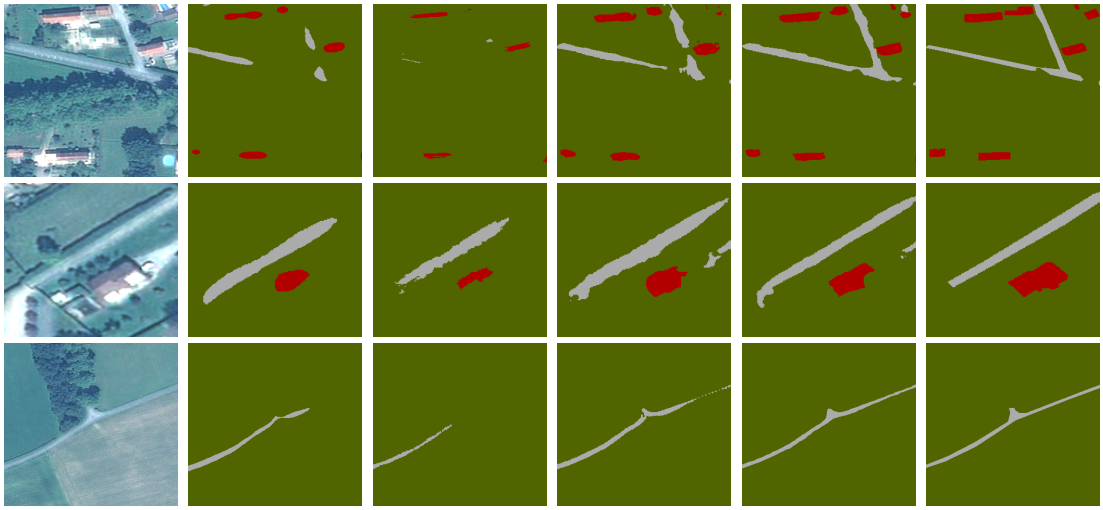


Figure 5.23: Visual comparison on closeups of the Pléiades dataset.

We also compare our RNN to the approach in [27] (here denoted by CNN+CRF), where a fully-connected CRF is coupled both to the input image and the coarse CNN output, in order to refine the predictions. This is the idea behind the so-called Deeplab network, which constitutes one of the most important current baselines in the semantic segmentation community. While the CRF itself could also be implemented as an RNN [214], we here stick to the original formulation because the CRF as RNN idea is only interesting if we want to train the system end to end (i.e., together with the coarse prediction network). In our case we wish to leave the coarse network as is, otherwise we risk overfitting it to this much smaller set. We thus simply use the CRF as in [27] and tune the energy parameters by performing a grid search using the enhancement set as a reference. Five iterations of inference on the fully-connected CRF were preformed in every case.

To further analyze our method, we also consider an alternative enhancement RNN in which the weights of the MLP are shared across the different classes (which we denote by "class-agnostic CNN+RNN"). This forces the system to learn the same function to update all the classes, instead of a class-specific function.

Numerical results are included in the table of Fig. 5.20(a) and the classification maps are shown in in Fig. 5.22. Close-ups of these maps are included in Fig. 5.23 to facilitate comparison. The CNN+CRF approach does sharpen the maps but this often occurs around the wrong edges. It also makes small objects disappear in favor of larger objects (usually the background class) when edges are not well marked, which explains the

mild increase in overall accuracy but the decrease in mean IoU. While the class-agnostic CNN+RNN outperforms the CRF, both quantitative and visual results are beaten by the CNN+RNN, supporting the importance of learning a class-specific enhancement function.

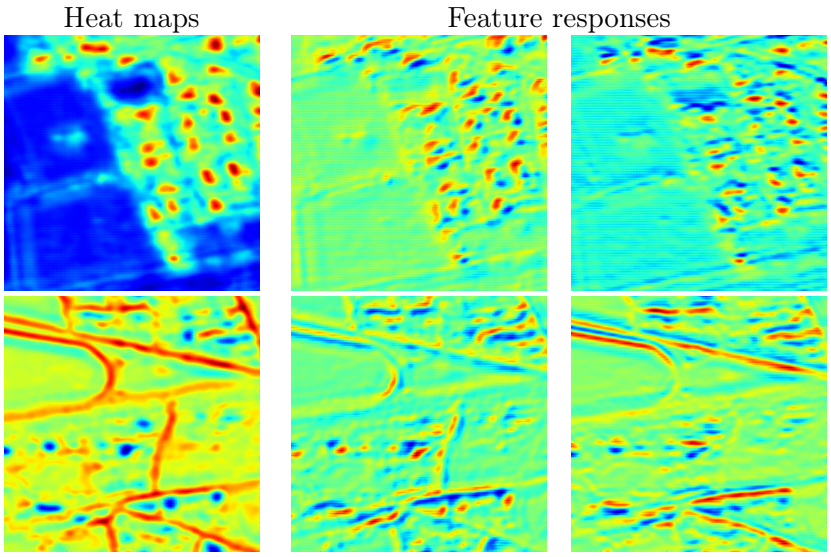
To validate the importance of using a recurrent architecture, and following Zheng et al. [214], we retrained our system considering every iteration of the RNN as an independent step with its own parameters. After training for the same number of iterations, it yields a lower performance on the test set compared to the RNN and a higher performance on the training set. If we keep on training, the non-recurrent network still enhances its training accuracy while performing poorly on the test set, implying a significant degree of overfitting with this variant of the architecture. This provides evidence that constraining our network to learn an iterative enhancement process is crucial for its success.

Feature visualization

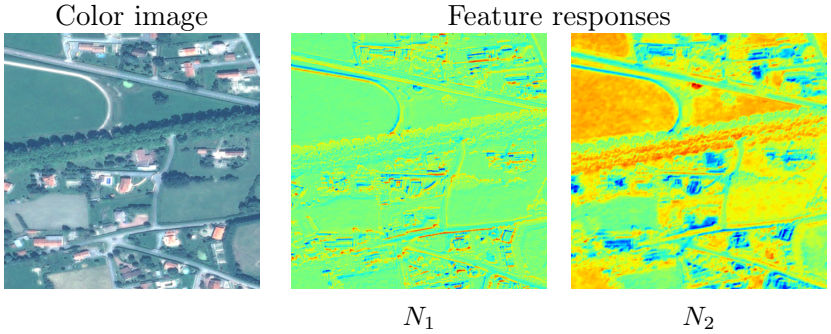
Though it is difficult to interpret the overall function learned by the RNN, especially the part of the multi-layer perceptron, there are some things we expect to find if we analyze the spatial filters M_i and N_j learned by the RNN (see Eq. 5.11). Carrying out this analysis is a way of validating the behavior of the network.

The iterative process learned by the RNN should combine information from both the heat maps and the image at every iteration (since the heat maps constitute the prior on where the objects are located, and the image guides the enhancement of these heat maps). A logical way of enhancing the classification is to align the high-gradient areas of the heat maps with the object boundaries. We expect then to find derivative operators among the filters N_j applied to the heat maps. Concerning the image filters N_j, we expect to find data-dependent filters (e.g., image edge detectors) that help identify the location of object boundaries.

To interpret the meaning of the filters learned by the RNN we plot the map of responses of a sample input to the different filters. Fig. 5.24(a) illustrates fragments of heat maps of the *building* and *road* classes, and the responses to two of the filters M_i learned by the RNN. When analyzing these responses we can observe that they act as gradients in different directions, confirming the expected behavior. Fig. 5.24(b) illustrates a fragment of the color image and its response to two filters N_j. One of them acts as a gradient operator an the other one highlights green vegetation, suggesting that this information is used to enhance the classification maps.



(a) Filter M_1 acts like a gradient operator in the South-East direction and M_2 in the North direction (top: building, bottom: road).



(b) N_1 acts like a gradient operator in the North direction and N_2 highlights green vegetation.

Figure 5.24: Feature responses (red: high, blue: low) to selected M_i and N_j filters, applied to the heat maps and input image respectively (see Eq. 5.11).

5.2.3 Concluding remarks

I have described a recurrent neural network (RNN) that learns how to refine the coarse output of another neural network, in the context of pixelwise image labeling. The inputs are both the coarse classification maps to be corrected and the original color image. The output at every RNN iteration is an update to the classification map of the previous iteration, using the color image for guidance.

Little human intervention is required, since the specifics of the refinement algorithm are not provided by the user but learned by the network itself. For this, we analyzed different iterative alternatives and devised a general formulation that can be interpreted as a stack of common neuron layers. At training time, the RNN discovers the relevant features to be taken both from the classification map and from the input image, as well as the function that combines them.

The experiments on satellite imagery show that the classification maps are improved significantly, increasing the overlap of the foreground classes with the ground truth. To conclude, we demonstrated that RNNs succeed in learning iterative processes for classification enhancement tasks.

Chapter 6

Conclusion and Future Work

I have presented my research on spectral-spatial image classification, mainly applied for automatic interpretation of remote sensing data. I have explored different strategies for this purpose, varying from the use of strong shape priors for object discrimination, regularization of classification probabilities on the image graphs, and up to the use of deep learning approaches which are capable of learning complex shape features from training data.

I can conclude that for specific applications, it is often advantageous to incorporate and enforce the known shape priors (e.g., as we have seen in Chapter 4, if we know that a shape of the object will only grow in time series, enforcing this shape constraint within the classification system naturally boosts its performance). However, if we aim at designing methods to automatically classify remote sensing images on a world-scale, the developed learning approaches must be generic and highly scalable. Within this context, deep learning techniques, in particular convolutional neural networks have gained significant attention, since they have demonstrated their ability to learn expressive multi-scale contextual features and have shown a remarkable computational performance. We have shown in Chapter 5 that the modern learning methods succeed in classifying new areas on the Earth belonging to the cities the classifier has never seen before. However, there is still significant work to do for designing systems which would be able to automatically create and update maps (e.g., geographic information system (GIS) maps) on a world-scale, using remote sensing images.

Future Work

To explore these ideas further, I have proposed the project **EPITOME** (**Efficient reP-resentatIon TO structure large-scale satellite iMagEs**), which was funded for the period of 2017-2021 by the French National Research Agency ANR. The goal of the EPITOME project is to devise novel effective representations for large-scale satellite images, which would represent and structure the essential content (which we call *epitome*) of input images, thus enabling novel possibilities for the exploitation of remote sensing datasets, including novel efficient navigation modes through space, scale, and perhaps time, or more powerful indexation mechanisms. Such representations must be highly generic and suited to massive image data, to be applicable for images from all over the world and for a wide range of applications. At the same time, they should be structure-preserving, i.e. best represent the meaningful objects in the image scenes, with the possibility to be enriched by semantic or other kinds of information.

The vector-based representation is well-known to provide several important advantages over raster images, the three most salient ones being compactness, scalability and easiness of updating [120]. These are the main reasons the GIS maps are stored as the database of vector primitives. Thus, I opt to devise a multi-resolution vector-based representation, together with the methods for its efficient generation, manipulation and continuous navigation between scales. Each level (scale) of this representation would convey different level of details from the image. Here I describe a non-exhaustive list of milestones I have identified to address the presented goal:

1. I plan to investigate the use of a large source of free-access online map data, such as OpenStreetMap [74], to automatically derive weakly labeled training data for learning about both geometric and semantic structures, their relations in the image scene and their accurate scale alignment. In particular, my current work deals with the development of a data alignment framework, by exploring CNN architectures to learn how to align maps with satellite images.

2. I envision to study how to adapt and extend the state-of-the-art image vectorization methods, for instance the algorithm based on iterative simplification of triangular mesh [207], to my task of representing satellite data. This implies enforcing the preservation of geometric structures and semantic labels discovered by applying a machine learning algorithm. In our initial work, we have proposed a novel polygonization algorithm, which vectorizes the input classification maps and is based on the approximation of a triangular mesh to the considered map [128].

3. It would certainly be interesting to design neural networks that output directly geometric primitives instead of raster classification maps. This objective is inspired by a few recent works on using neural networks to output geometric objects. For example, a recurrent neural network has been used to generate the convex hull of a point cloud [197] and in [46] the coordinates of object bounding boxes are regressed by a CNN. I envision to further bridge the gap between deep machine learning and geometric modeling tools, by devising novel methods which allow learning directly in a space of vector primitives. A possible approach consists in defining a discrete finite set of vector primitives within admitted discretization error bounds, and then designing an appropriate neural network architecture capable to learn how to output the desired representations in this space of primitives.

Appendix A

Proofs related to BPT optimization

I first elaborate the proofs that supplement the paragraphs that describe the space of moves in the optimization approach. I then add the proofs of the space and computational complexities of including convex hulls in Binary Partition Trees (BPTs).

A.1 Properties of *prune-and-paste* moves

Proposition 1. *Given a tree τ, suppose a node R_m is pasted at $\tau_i < \tau_1$ leading to a new tree φ. Let us consider an alternative move that pastes R_m at τ_j, with $\tau_i < \tau_j < \tau_1$, producing a tree ψ. In the cases where either $C(\varphi_1) - C(\tau_1) \leqslant 0$ or $C(R_m) \geqslant C(\varphi_1) - C(\tau_1)$, then $C(\psi_1) \geqslant C(\varphi_1)$.*

Proof. Let us abbreviate $\mathcal{E}(\tau_i)$ as e_i^τ and $C(\tau_i)$ as c_i^τ. Following (3.11) in Chapter 3,

$$c_1^\tau = min(e_1^\tau, \overline{c_2^\tau} + min(e_2^\tau, \overline{c_3^\tau} + min(...min(e_{i-2}^\tau, \overline{c_{i-1}^\tau} + min(e_{i-1}^\tau, \overline{c_i^\tau} + c_i^\tau))...))), \quad \text{(A.1)}$$

where $\overline{c_i^\tau}$ denotes the sibling of c_i^τ (see Fig. 3.4-b). This implies:

$$\underbrace{\underset{k=1}{\overset{i-1}{\forall}} \left(e_k^\tau \geqslant c_1^\tau - \Sigma_{j=2}^k \overline{c_j^\tau} \right)}_{\alpha} \text{ and } \left(c_i^\tau \geqslant c_1^\tau - \Sigma_{j=2}^i \overline{c_j^\tau} \right). \quad \text{(A.2)}$$

Let us now suppose that a node R_m is pasted at τ_i. Then

$$c_1^\varphi = min(e_1^\varphi, \overline{c_2^\tau} + min(e_2^\varphi, \overline{c_3^\tau} + min(...min(e_{i-2}^\varphi, \overline{c_{i-1}^\tau} + min(e_{i-1}^\varphi, \overline{c_i^\tau} + min(e_x^\varphi, c_i^\tau + c_m^R)))...))), \quad \text{(A.3)}$$

which implies:

$$\underbrace{\overset{i-1}{\underset{k=1}{\forall}}\left(e_k^{\varphi} \geqslant c_1^{\varphi} - \Sigma_{j=2}^{k}\overline{c_j^{\tau}}\right)}_{\beta} \text{ and } \left(e_x^{\varphi} \geqslant c_1^{\varphi} - \Sigma_{j=2}^{i}\overline{c_j^{\tau}}\right) \text{ and } \underbrace{\left(c_m^{R} \geqslant c_1^{\varphi} - \Sigma_{j=2}^{i}\overline{c_j^{\tau}} - c_i^{\tau}\right)}_{\gamma}. \tag{A.4}$$

Let us now paste R_m one position upper than before. We wish to check if it is possible that this move will be better that the previous one ($c_1^{\psi} < c_1^{\varphi}$):

$$\begin{aligned} c_1^{\psi} = & min(e_1^{\varphi}, \overline{c_2^{\tau}} + min(e_2^{\varphi}, \overline{c_3^{\tau}} + min(...min(e_{i-2}^{\varphi}, \overline{c_{i-1}^{\tau}} + min(e_y^{\psi}, \\ & c_m^{R} + min(e_{i-1}^{\tau}, \overline{c_i^{\tau}} + c_i^{\tau})))...)) < c_1^{\varphi}. \end{aligned} \tag{A.5}$$

This can be true if and only if:

$$\begin{aligned} & \underbrace{\overset{i-2}{\underset{k=1}{\exists}}\left(e_k^{\varphi} < c_1^{\varphi} - \Sigma_{j=2}^{k}\overline{c_j^{\tau}}\right)}_{I} \text{ or } \underbrace{\left(e_y^{\psi} < c_1^{\varphi} - \Sigma_{j=2}^{i-1}\overline{c_j^{\tau}}\right)}_{II} \\ & \text{or } \underbrace{\left(e_{i-1}^{\tau} < c_1^{\varphi} - \Sigma_{j=2}^{i-1}\overline{c_j^{\tau}} - c_m^{R}\right)}_{III} \text{ or } \underbrace{\left(c_m^{\tau} < c_1^{\varphi} - \Sigma_{j=2}^{i}\overline{c_j^{\tau}} - c_i^{\tau}\right)}_{IV}. \end{aligned} \tag{A.6}$$

In this expression, I contradicts β. Considering that $e_y^{\psi} = e_{i-1}^{\varphi}$ (see Fig. 3.4-b), the term II also contradicts β. The term IV contradicts γ. We must now analyze III. By combining III and α:

$$c_1^{\tau} - \Sigma_{j=2}^{i-1}\overline{c_j^{\tau}} \leqslant e_{i-1}^{\tau} < c_1^{\varphi} - \Sigma_{j=2}^{i-1} - c_m^{R} \Rightarrow c_m^{R} < c_1^{\varphi} - c_1^{\tau}. \tag{A.7}$$

If $c_1^{\varphi} - c_1^{\tau} \leqslant 0$, then c_m^{R} must be non-positive, which contradicts our hypothesis. If $c_1^{\varphi} - c_1^{\tau} > 0$: then it must be $c_m^{R} < c_1^{\varphi} - c_1^{\tau}$. As a conclusion, if the first move decreases C, III is contradicted, hence it must be $c_1^{\psi} \geqslant c_1^{\varphi}$. For a positive gain, III is contradicted unless $c_m^{R} < c_1^{\varphi} - c_1^{\tau}$. □

Proposition 2. *Let us consider a case where Prop. 1 hypotheses do not apply. There might then exist a higher paste place τ_α so that $C(\psi_1) < C(\varphi_1)$. Let us suppose that instead of pasting at τ_α we paste at τ_β, with $\tau_\alpha < \tau_\beta < \tau_1$, leading to a tree ρ. Then $C(\rho_1)$ would monotonously decrease as the paste place τ_β is located higher.*

Proof. If Prop. 1 hypotheses do not apply, then the term III in its proof must be true. This term implies that when pasting at τ_j, the cut on the tree will be located at or below

R_m. The cost c_1^ψ associated with this move will then be

$$c_1^\psi = \Sigma_{j=2}^{i} \overline{c_j^\tau} + c_i^\tau + c_m^R, \tag{A.8}$$

considering the location of the new cut. Analogously, the cost when pasting R_m k units up of i is:

$$c_1^\rho = \Sigma_{j=2}^{i-k} \overline{c_j^\tau} + c_{i-k}^\tau + c_m^R. \tag{A.9}$$

Notice in the previous expression that we consider that the cut is still as low as R_m. The cut could not be higher, because if it were the case, then it would have already been cut there before.

Let us now see if the cost c_1^ρ could increase as k advances:

$$\begin{aligned} &\Sigma_{j=2}^{i-k} \overline{c_j^\tau} + c_{i-k}^\tau + c_m^R - \left(\Sigma_{j=2}^{i-k+1} \overline{c_j^\tau} + c_{i-k+1}^\tau + c_m^R\right) \\ =& -\overline{c_{i-k+1}^\tau} + c_{i-k}^\tau - c_{i-k+1}^\tau > 0 \\ \Leftrightarrow& c_{i-k}^\tau > c_{i-k+1}^\tau + \overline{c_{i-k+1}^\tau}, \end{aligned} \tag{A.10}$$

contradicting the algorithm to compute the cuts, hence the cost at R_1 must monotonously decrease. □

Proposition 3. *Let us suppose we paste R_m at or over the initial cut of tree τ, leading to tree φ. Let us consider we paste higher instead, producing tree ψ. It must then be $c_1^\psi \geqslant c_1^\varphi$.*

Proof. Let us resume the proof of Proposition 2. It was shown that $c_1^\psi < c_1^\varphi$ if and only if III was false. At that point we could not contradict III but show that under certain conditions it would be contradicted. Now we will show that the fact that we know the first cut was at or below τ_i will contradict III.

After III and γ we have:

$$\begin{aligned} &e_{i-1}^\tau + c_1^\varphi - \Sigma_{j=2}^{i} \overline{c_j^\tau} - c_i^\tau \leqslant e_{i-1}^\tau + c_m^R < c_1^\varphi - \Sigma_{j=2}^{i-1} \overline{c_j^\tau} \\ \Leftrightarrow& e_{i-1}\tau - \overline{c_i^\tau} - c_i^\tau \leqslant e_{i-1}^\tau + c_m^R < 0. \end{aligned} \tag{A.11}$$

If we now add the knowledge about the cut being below τ_{i-1}, it must be $e_{i-1}^\tau > c_i^\tau + \overline{c_i^\tau}$, then

$$c_i^\tau + \overline{c_i^\tau} - \overline{c_i^\tau} - c_i^\tau < e_{i-1}^\tau - \overline{c_i}^\tau - c_i^\tau \leqslant e_{i-1}^\tau + c_m^R < 0 \Leftrightarrow 0 < 0. \tag{A.12}$$

Therefore, III cannot be true, which proves the proposition.

□

Corollary: Combining Proposition 2, where C monotonously decreases ($\leqslant$) as the paste location gets higher, and Proposition 3, where C cannot decrease ($\geqslant$), it becomes evident that pasting a node anywhere between the initial cut and the lowest common ancestor produces the same effect on the energy.

Proposition 4. *The amount of spatially adjacent regions in a balanced BPT is bounded by* $O(n\log(n))$.

Proof. Let us call $\mathcal{N}_R$ the number of neighbors of the region R. At the lowest scale and in a discrete environment we can suppose that the number of neighbors is equal to its boundary length (δR). We are interested in knowing the number of neighbors at all scales. In a balanced tree it can be assumed that the number of neighbors at every scale is half the number at the following one. As a result:

$$\mathcal{N}_R = \delta R + \frac{1}{2}\delta R + \frac{1}{2^2}\delta R + \frac{1}{2^3}\delta R + ... < 2\delta R. \tag{A.13}$$

In a discrete implementation (assuming 4-connectivity):

$$\mathcal{N}_R < 2\delta R \leqslant 2 \cdot 4|R| = 8|R|. \tag{A.14}$$

The summation of the neighbors of *all* regions in a tree $\mathcal{T}$ is then

$$\sum_{R_i \epsilon \mathcal{T}} \mathcal{N}_{R_i} < 8 \sum_{R_i \epsilon \mathcal{T}} |R_i|. \tag{A.15}$$

Following (A.18), the total number of neighbors (the possible cut/paste moves) is a factor of $n\log(n)$. □

A.2 Complexity of incorporating convex hulls

Proposition 5. *The storage space required to add the convex hull to every node of a balanced BPT in a discrete environment is bounded by* $O(n\log(n))$.

Proof. Let us call $CH(R)$ the convex hull of a region R. In the extreme case (the most compact region), $CH(R)$ can be as large as the perimeter δR of R which, in a discrete implementation (assuming 4-connectivity) does not contain more points than four times the area of the region:

$$CH(R) \leqslant \delta R \leqslant 4|R|. \tag{A.16}$$

As a consequence, the points of the convex hull of *all* regions in a tree $\mathcal{T}$ must be

$$\sum_{R_i \epsilon \mathcal{T}} |CH(R_i)| \leqslant 4 \sum_{R_i \epsilon \mathcal{T}} |R_i|. \tag{A.17}$$

If we observe that in a balanced tree

$$\sum_{R_i \epsilon \mathcal{T}} |R_i| = \sum_{l=1}^{\#levels} \sum_{R_j \epsilon l \epsilon \mathcal{T}} |R_j| = \sum_{l=1}^{\#levels} n = n \sum_{l=1}^{\#levels} 1 = n \cdot \#levels = n \log(n), \tag{A.18}$$

then (A.17) is bounded by a factor of $n \log(n)$.

□

Proposition 6. *The complexity of computing the convex hull of every region represented in a balanced BPT in a discrete environment, is bounded by* $O(n \log(n))$.

Proof. Let us call $CH(R)$ the convex hull of a region R. In the extreme case (the most compact region), $CH(R)$ can be as large as the perimeter δR of R which, in a discrete implementation (assuming 4-connectivity) does not contain more points than four times the area of the region:

$$CH(R) \leqslant \delta R \leqslant 4|R|. \tag{A.19}$$

The time to compute $CH(R_i)$ is linear on the number of points in the polygons of the children:

$$O(|\delta LeftChild(R_i)| + |\delta RightChild(R_i)|). \tag{A.20}$$

The time to compute the convex hull of every node in the tree is then bounded by a factor of:

$$\sum_{R_i \epsilon \mathcal{T}} (|\delta LeftChild(R_i)| + |\delta RightChild(R_i)|)$$
$$\leqslant 4 \sum_{R_i \epsilon \mathcal{T}} (|LeftChild(R_i)| + |RightChild(R_i)|) = 4 \sum_{R_i \epsilon \mathcal{T}} |R_i|. \tag{A.21}$$

Following (A.18), the execution time is then a factor of $n \log(n)$. □

Appendix B

Submodularity proof

$W_{i,j}$ can be written as a sum of terms over time t, which is, if $\tau_i \leqslant \tau_j$:

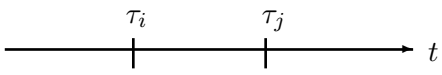
$$W_{i,j}(\tau_i,\tau_j) \;=\; \sum_t \begin{cases} W^t_{i,j}(0,0) & \text{if } t < \min(\tau_i,\tau_j) \quad (A) \\ W^t_{i,j}(1,0) & \text{if } \tau_i \leqslant t < \tau_j \quad (B) \\ W^t_{i,j}(1,1) & \text{if } t \geqslant \max(\tau_i,\tau_j) \quad (D) \end{cases}$$

and

$$W_{i,j}(\tau_i,\tau_j) \;=\; \sum_t \begin{cases} W^t_{i,j}(0,0) & \text{if } t < \min(\tau_i,\tau_j) \quad (A) \\ W^t_{i,j}(0,1) & \text{if } \tau_j \leqslant t < \tau_i \quad (C) \\ W^t_{i,j}(1,1) & \text{if } t \geqslant \max(\tau_i,\tau_j) \quad (D) \end{cases}$$

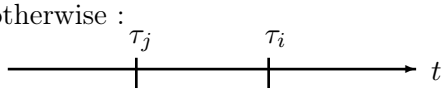
otherwise.

We can thus represent $W_{i,j}(\tau_i,\tau_j)$ by the sequence of cases A, B, C or D chosen for each t:



$W_{i,j}(\tau_i,\tau_j) = \sum_t$ AAAAA BBBBBBB DDDDDD

when $\tau_i \leqslant \tau_j$, and otherwise :



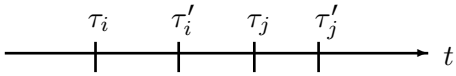
$W_{i,j}(\tau_i,\tau_j) = \sum_t$ AAAAA CCCCCCC DDDDDD

Now, to represent the submodularity condition, let $\tau_i, \tau_j, \tau'_i, \tau'_j$ be any times, satisfying $\tau_i \leqslant \tau'_i$ and $\tau_j \leqslant \tau'_j$. We will suppose moreover that $\tau_i \leqslant \tau_j$: otherwise consider $W_{j,i}(\tau_j,\tau_i)$

instead and reverse names i and j. Three cases are possible:

1. $\tau_i \leqslant \tau_i' \leqslant \tau_j \leqslant \tau_j'$
2. $\tau_i \leqslant \tau_j \leqslant \tau_i' \leqslant \tau_j'$
3. $\tau_i \leqslant \tau_j \leqslant \tau_j' \leqslant \tau_i'$

In case (1), the terms of the submodularity condition write:

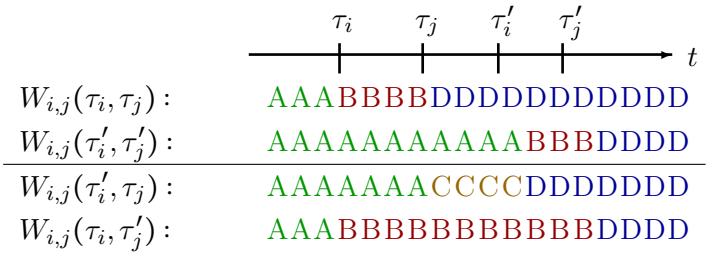


$W_{i,j}(\tau_i, \tau_j)$: AAABBBBBBBBDDDDDDD
$W_{i,j}(\tau_i', \tau_j')$: AAAAAAABBBBBBBDDDD

$W_{i,j}(\tau_i', \tau_j)$: AAAAAAABBBBDDDDDDD
$W_{i,j}(\tau_i, \tau_j')$: AAABBBBBBBBBBDDDD

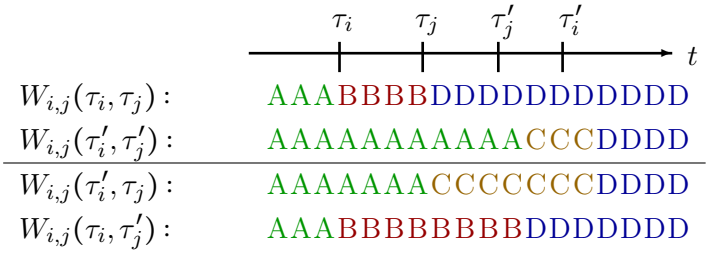
and the submodularity condition is satisfied if the sum of the two first rows is less than or equal to the sum of the two last rows. It turns out that in the particular case above, the sums are equal for each instant t, and consequently the submodularity condition is satisfied as being an equality.

In case (2), the terms of the submodularity condition write:



This time, the two sums, at any instant t, are equal if $t < \tau_j$ or $t \geqslant \tau'_i$. For other values of t, that is to say when $\tau_j \leqslant t < \tau'_i$, the sum $A + D$ is to be compared with $B + C$, *i.e.* the question is to compare $W^t_{i,j}(0,0) + W^t_{i,j}(1,1)$ with $W^t_{i,j}(0,1) + W^t_{i,j}(1,0)$. As E^t is binary submodular by hypothesis, and by definition of binary submodularity in equation (4.2), we have $A + D \leqslant B + C$, and thus the sum of the two first rows is less than or equal to the sum of the two last ones, which means precisely $W_{i,j}(\tau_i, \tau_j) + W_{i,j}(\tau'_i, \tau'_j) \leqslant W_{i,j}(\tau_i, \tau'_j) + W_{i,j}(\tau'_i, \tau_j)$, and thus the submodularity condition for E is checked.

Finally, in case (3), the terms of the submodularity condition write:



The two sums are equal for all instants $t < \tau_j$ or $t \geqslant \tau'_j$. When $\tau_j \leqslant t < \tau'_j$, the sum $A + D$ is to be compared with $B + C$, as previously, and consequently the submodularity condition for E is checked again.

In all cases (1), (2) and (3), E is proven to be submodular, and this concludes the proof.

Bibliography

[1] Abdullah Al-Dujaili, François Merciol, and Sébastien Lefèvre. Graphbpt: An efficient hierarchical data structure for image representation and probabilistic inference. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 301–312. Springer, 2015.

[2] Pablo Arbeáez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE TPAMI*, 33(5):898–916, May 2011.

[3] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[5] C. Becker, R. Rigamonti, V. Lepetit, and P. Fua. Supervised feature learning for curvilinear structure segmentation. In *MICCAI*, pages 526–533, September 2013.

[6] C. Becker, R. Rigamonti, V. Lepetit, and P. Fua. Supervised feature learning for curvilinear structure segmentation. In *MICCAI*, pages 526–533, September 2013.

[7] J. A. Benediktsson and P. H. Swain. *Statistical Methods and Neural Network Approaches for Classification of Data from Multiple Sources.* PhD thesis, Purdue Univ., School of Elect. Eng., West Lafayette, IN, 1990.

[8] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy. Conjugate gradient neural networks in classification of very high dimensional remote sensing data. *International Journal of Remote Sensing*, 14(15):2883 – 2903, 1993.

[9] J.A. Benediktsson, M. Pesaresi, and K. Arnason. Classification and feature extraction for remote sensing images from urban areas based on morphological transformations. *IEEE Trans. Geosci. Remote Sens.*, 41(9):1940–1949, Sept. 2003.

[10] K. Bernard, Y. Tarabalka, J. Angulo, J. Chanussot, and J. A. Benediktsson. Spectral-spatial classification of hyperspectral data based on a stochastic minimum spanning forest approach. *IEEE Transactions on Image Processing*, 21(4):2008–2021, April 2012.

[11] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. Kernelized structural SVM learning for supervised object segmentation. In *CVPR*, pages 2153–2160, June 2011.

[12] J. Besag. On the statistical analysis of dirty pictures. *Journal of Royal Statistical Society B*, 48(3):259–302, 1986.

[13] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. M. Nasrabadi, and J. Chanussot. Hyperspectral remote sensing data analysis and future challenges. *IEEE Geosci. and Remote Sens. Mag.*, 1(2):6–36, 2013.

[14] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[15] F. G. Blankenberg, R. L. Teplitz, W. Ellis, M. S. Salamat, B. H. Min, L. Hall, D. B. Boothroyd, I. M. Johnstone, and D. R. Enzmann. The influence of volumetric tumor doubling time, DNA ploidy, and histologic grade on the survival of patients with intracranial astrocytomas. *Am J Neuroradiol*, 16:1001–1012, 1995.

[16] M. Borassi, P. Crescenzi, M. Habib, W. A. Kosters, A. Marino, and F. W. Takes. Fast diameter and radius BFS-based computation in (weakly connected) real-world graphs: With an application to the six degrees of separation games. *Theor. Comput. Sci.*, 586:59–80, 2015.

[17] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, pages 111–118, 2010.

[18] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI*, 26(9):1124–1137, September 2004.

[19] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23(11):1222–1239, 2001.

[20] Felipe Calderero and Ferran Marques. Region merging techniques using information theory statistical measures. *IEEE Trans. Image Process.*, 19(6):1567–1586, 2010.

[21] G. Camps-Valls, L. Gomez-Chova, J. Muñoz-Marí, J. Vila-Francés, and J. Calpe-Maravilla. Composite kernels for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.*, 3(1):93–97, 2006.

[22] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, 1997.

[23] S. Chambon, C. Gourraud, J.-M. Moliard, and P. Nicolle. Road crack extraction with adapted filtering and Markov model-based segmentation. In *VISAPP(2)*, pages 81–90, May 2010.

[24] S. Chambon, C. Gourraud, J.-M. Moliard, and P. Nicolle. Road crack extraction with adapted filtering and Markov model-based segmentation. In *VISAPP(2)*, pages 81–90, May 2010.

[25] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. In *SIGGRAPH*, 2007.

[26] Liang-Chieh Chen, Jonathan T Barron, George Papandreou, Kevin Murphy, and Alan Yuille. Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform. *arXiv preprint arXiv:1511.03328*, 2015.

[27] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, May 2015.

[28] Yunjin Chen, Wei Yu, and Thomas Pock. On learning optimized reaction diffusion processes for effective image restoration. In *IEEE CVPR*, pages 5261–5269, 2015.

[29] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[30] D. G. Corneil, F. F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. *Discrete Applied Mathematics*, 113(2–3):143–166, 2001.

[31] Daniel Cremers, Stanley J Osher, and Stefano Soatto. Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. *IJCV*, 69(3):335–351, 2006.

[32] Daniel Cremers, Florian Tischhauser, Joachim Weickert, and Christoph Schnorr. Diffusion snakes: introducing statistical shape knowledge into the Mumford-Shah functional. *Int. Journal of Computer Vision*, 50(3):295–313, 2002.

[33] Gabriela Csurka, Diane Larlus, Florent Perronnin, and France Meylan. What is a good evaluation measure for semantic segmentation?. In *BMVC*, 2013.

[34] Gabriela Csurka, Diane Larlus, Florent Perronnin, and France Meylan. What is a good evaluation measure for semantic segmentation?. In *BMVC*, 2013.

[35] G. B. Dantzig and D. R. Fulkerson. On the max-flow min-cut theorem of networks. *Ann. Math. Studies*, (38), 1956.

[36] J. Darbon. Global optimization for first order Markov random fields with submodular priors. In *Int. Workshop on Combinatorial Image Analysis*, 2008.

[37] Piali Das, Olga Veksler, Vyacheslav Zavadsky, and Yuri Boykov. Semiautomatic segmentation with compact shape prior. *Image and Vision Computing*, 27(1):206–219, 2009.

[38] A. Delong and Y. Boykov. A scalable graph-cut algorithm for n-d grids. In *CVPR*, 2008.

[39] A. Delong and Y. Boykov. Globally optimal segmentation of multi-region objects. In *ICCV*, 2009.

[40] D. Dementhon. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Statistical Methods in Video Processing Workshop (SMVP)*, 2002.

[41] Stig Descamps, Xavier Descombes, Arnaud Béchet, and Josiane Zerubia. Automatic flamingo detection using a multiple birth and death process. In *IEEE ICASSP*, pages 1113–1116, 2008.

[42] Lee Dice. Measure of the amount of ecological association between species. *Ecology*, 26(3):297–302, 1945.

[43] Anastasia Dubrovina, Pavel Kisilev, Boris Ginsburg, Sharbell Hashoul, and Ron Kimmel. Computational mammography using deep neural networks. *Computer*

Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, pages 1–5, 2016.

[44] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[45] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2001.

[46] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *IEEE CVPR*, pages 2147–2154, 2014.

[47] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE TPAMI*, 35(8):1915–1929, 2013.

[48] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson. Spectral and spatial classification of hyperspectral data using svms and morphological profiles. *IEEE Trans. Geosci. Remote Sens.*, 46(11):3804–3814, Nov 2008.

[49] Mathieu Fauvel, Yuliya Tarabalka, Jon Atli Benediktsson, Jocelyn Chanussot, and James C Tilton. Advances in spectral-spatial classification of hyperspectral images. *Proc. of the IEEE*, 101(3):652–675, 2013.

[50] P. Felzenszwalb. *Representation and Detection of Shapes in Images*. PhD thesis, Massachusetts Institute of Technology, Sept. 2003.

[51] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE TPAMI*, 32(9):1627–1645, 2010.

[52] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. In *MICCAI*, pages 130–137, October 1998.

[53] Daniel Freedman and Tao Zhang. Interactive graph cut based segmentation with shape priors. In *IEEE CVPR*, pages 755–762, 2005.

[54] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE TPAMI*, 13(9):891–906, September 1991.

[55] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE TPAMI*, 13(9):891–906, September 1991.

[56] Gareth Funka-Lea, Yuri Boykov, Charles Florin, M-P Jolly, Romain Moreau-Gobard, Rana Ramaraj, and Daniel Rinck. Automatic heart isolation for ct coronary visualization using graph-cuts. In *IEEE Int. Symp. Biomedical Imaging: Nano to Macro*, pages 614–617, 2006.

[57] A. Gamal-Eldin, X. Descombes, G. Charpiat, and J. Zerubia. Multiple birth and cut algorithm for point process optimization. In *SITIS*, 2010.

[58] Eduardo S. L. Gastal and Manuel M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, 30(4):69:1–69:12, 2011.

[59] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the bayesian restoration of images. *IEEE TPAMI*, 6(6):721–741, November 1984.

[60] Markus Gerke. Use of the stair vision library within the ISPRS 2d semantic labeling benchmark (vaihingen). Technical report, University of Twente, 2015.

[61] P. Ghamisi, R. Souza, J. A. Benediktsson, X. X. Zhu, L. Rittner, and R. A. Lotufo. Extinction profiles for the classification of remote sensing data. *IEEE Trans. Geosci. Remote Sens.*, 54(10):5631–5645, Oct 2016.

[62] L. Giglio, T. Loboda, D. P. Roy, B. Quayle, and C. O. Justice. An active-fire based burned area mapping algorithm for the modis sensor. *Remote Sens. of Environment*, (113):408–420, 2009.

[63] Gonzalo Giribet. Efficient tree searches with available algorithms. *Evolutionary bioinformatics online*, 3:341, 2007.

[64] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

[65] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, (35(4)), 1988.

[66] G. González, E. Türetken, F. Fleuret, and P. Fua. Delineating trees in noisy 2D images and 3D image-stacks. In *CVPR*, pages 2799–2806, June 2010.

[67] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[68] Lena Gorelick, Frank R. Schmidt, and Yuri Boykov. Fast trust region for segmentation. In *IEEE CVPR*, pages 1714–1721, June 2013.

[69] Lena Gorelick, Olga Veksler, Yuri Boykov, and Claudia Nieuwenhuis. Convexity shape prior for segmentation. In *ECCV*, pages 675–690, 2014.

[70] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):771–732, 1995.

[71] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):771–732, 1995.

[72] P. J. Green and A. Mira. Delayed rejection in reversible jump Metropolis-Hastings. *Biometrika*, 88(4):1035–1053, 2001.

[73] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video segmentation. In *CVPR*, 2010.

[74] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, Oct 2008.

[75] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[76] J. Ham, Yangchi Chen, M. M. Crawford, and J. Ghosh. Investigation of the random forest framework for classification of hyperspectral data. *IEEE Trans. Geosci. Remote Sens.*, 43(3):492–501, March 2005.

[77] R. Hansch and O. Hellwich. When to fuse what? random forest based fusion of low-, mid-, and high-level information for land cover classification from optical and sar images. In *IEEE IGARSS*, pages 3587–3590, July 2016.

[78] Bharath Hariharan, Pablo Arbelaez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[79] D. Haverkamp and C. Tsatsoulis. Using temporal information in an automated classification of summer, marginal ice zone imagery. In *IGARSS'96*, pages 109–111, May 1996.

[80] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE CVPR*, pages 1026–1034, 2015.

[81] Rui Huang, Vladimir Pavlovic, and Dimitris N Metaxas. A graphical model framework for coupling MRFs and deformable models. In *IEEE CVPR*, pages 739–746, 2004.

[82] X. Huang and L. Zhang. A comparative study of spatial approaches for urban mapping using hyperspectral rosis images over pavia city, northern italy. *International Journal of Remote Sensing*, 30(12):3205–3221, 2009.

[83] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[84] H. Ishikawa. Exact optimization for Markov random fields with convex priors. *TPAMI*, 25, October 2003.

[85] ISPRS. 2D semantic labeling contest. http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html.

[86] M. Jacob and M. Unser. Design of steerable filters for feature detection using Canny-like criteria. *IEEE TPAMI*, 26(8):1007–1019, August 2004.

[87] M. Jacob and M. Unser. Design of steerable filters for feature detection using Canny-like criteria. *IEEE TPAMI*, 26(8):1007–1019, August 2004.

[88] S.-G. Jeong, Y. Tarabalka, and J. Zerubia. Marked point process model for curvilinear structures extraction. In *EMMCVPR 2015, LNCS 8932*, pages 436–449, January 2015.

[89] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[90] T. Joachims. Training linear SVMs in linear time. In *KDD*, pages 217–226, 2006.

[91] Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *IEEE CVPR Workshops*, 2016.

[92] Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *IEEE CVPR Workshops*, pages 1–9, 2016.

[93] Tina Kapur, W Eric L Grimson, William M Wells III, and Ron Kikinis. Segmentation of brain tissue from magnetic resonance images. *Medical Image Analysis*, 1(2):109–127, 1996.

[94] K. Karantzalos and N. Paragios. Recognition-driven two-dimensional competing priors toward automatic and accurate building detection. *IEEE TGRS*, 47(1):133–144, Jan 2009.

[95] Nirmal Keshava. A survey of spectral unmixing algorithms. *Lincoln Lab. J.*, 14:55–78, 2003.

[96] R. L. Kettig and D. A. Landgrebe. Classification of multispectral image data by extraction and classification of homogeneous objects. *IEEE Trans. Geoscience Electronics*, 14(1):19–26, Jan. 1976.

[97] S. Kim, C. D. Yoo, S. Nowozin, and P. Kohli. Image segmentation using higher-order correlation clustering. *IEEE TPAMI*, 36(9):1761–1774, September 2014.

[98] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[99] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1143, 1995.

[100] E. Konukoglu, O. Clatz, B. H. Menze, B. Stieltjes, M. Weber, E. Mandonnet, H. Delingette, and N. Ayache. Image guided personalization of reaction-diffusion type tumor growth models using modified anisotropic eikonal equations. *IEEE Trans Med Imaging*, 29:77–95, 2010.

[101] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[102] M Pawan Kumar, Philip H.S. Torr, and Andrew Zisserman. Obj cut. In *IEEE CVPR*, pages 18–25, 2005.

[103] S. Kumar and M. Hebert. Discriminative random fields. *IJCV*, 68(2):179–201, 2006.

[104] Camille Kurtz, Nicolas Passat, Pierre Gancarski, and Anne Puissant. Extraction of complex patterns from multiresolution remote sensing images: A hierarchical top-down methodology. *Pattern Recognition*, 45(2):685–706, 2012.

[105] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. In *SIGGRAPH*, pages 795–802, 2005.

[106] C. Lacoste, X. Descombes, and J. Zerubia. Point processes for unsupervised line network extraction in remote sensing. *IEEE TPAMI*, 27(10):1568–1579, October 2005.

[107] C. Lacoste, X. Descombes, and J. Zerubia. Point processes for unsupervised line network extraction in remote sensing. *IEEE TPAMI*, 27(10):1568–1579, October 2005.

[108] J. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmetaing and labeling sequence data. In *ICML*, pages 282–289, 2001.

[109] Pierre Lassalle, Jordi Inglada, Julien Michel, Manuel Grizonnet, and Julien Malik. A scalable tile-based framework for region-merging segmentation. *IEEE Tran. Geosci. Remote Sens.*, 53(10):5473–5485, 2015.

[110] M. W. Law and A. Chung. Three dimensional curvilinear structure detection using optimally oriented flux. In *ECCV*, pages 368–382, October 2008.

[111] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[112] Sebastien Lefevre, Laetitia Chapel, and François Merciol. Hyperspectral image classification from multiscale description with constrained connectivity and metric learning. In *6th International Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2014.

[113] Victor Lempitsky, Andrew Blake, and Carsten Rother. Image segmentation by branch-and-mincut. In *ECCV*, pages 15–29, 2008.

[114] Michael E Leventon, W Eric L Grimson, and Olivier Faugeras. Statistical shape influence in geodesic active contours. In *IEEE CVPR*, pages 316–323, 2000.

[115] J. B. Lewis. Fire mapping for managers in North Australian savanna; an object orientated classification model for MODIS imagery. In *Proc. of ARSPC'12*, pages 1–12, Fremantle, Western Australia, 2004.

[116] O. Lézoray and L. Grady. *Image Processing and Analysis with Graphs: Theory and Practice*. Digital Imaging and Computer Vision. CRC Press / Taylor and Francis, 2012.

[117] Fulu Li and Andrew Lippman. Random tree optimization for the construction of the most parsimonious phylogenetic trees. In *CISS*, pages 757–762, 2009.

[118] Kang Li, Xiaodong Wu, Danny Z. Chen, and Milan Sonka. Optimal surface segmentation in volumetric images - a graph-theoretic approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):119, January 2006.

[119] Wenwen Li, Michael F Goodchild, and Richard Church. An efficient measure of compactness for two-dimensional shapes and its application in regionalization problems. *Int. Journal of Geographical Information Science*, 27(6):1227–1250, 2013.

[120] Z. Liao, H. Hoppe, D. Forsyth, and Y. Yu. A subdivision-based representation for vector image editing. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1858–1867, Nov 2012.

[121] M. N. M. van Lieshout. *Markov point processes and their application*. Imperial College Press, 2000.

[122] Risheng Liu, Zhouchen Lin, Wei Zhang, and Zhixun Su. Learning PDEs for image restoration via optimal control. In *ECCV*, pages 115–128. Springer, 2010.

[123] Risheng Liu, Zhouchen Lin, Wei Zhang, Kewei Tang, and Zhixun Su. Toward designing intelligent pdes for computer vision: An optimal control approach. *Image and vision computing*, 31(1):43–56, 2013.

[124] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, 2015.

[125] Huihai Lu, John C Woods, and Mohammed Ghanbari. Binary partition tree for semantic object extraction and image segmentation. *IEEE Trans. Circuits and Systems for Video Technology*, 17(3):378–383, 2007.

[126] A. Lucchi, Y. Li, K. Smith, and P. Fua. Structured image segmentation using kernelized features. In *ECCV*, pages 400–413, October 2012.

[127] E. Maggiori, Y. Tarabalka, and G. Charpiat. Improved partition trees for multi-class segmentation of remote sensing images. In *IEEE IGARSS*, pages 1016–1019. IEEE, 2015.

[128] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. Polygonization of remote sensing classification maps by mesh approximation. In *IEEE ICIP*. IEEE, 2017.

[129] Emmanuel Maggiori, Yuliya Tarabalka, and Guillaume Charpiat. Optimizing partition trees for multi-object segmentation with shape prior. In *26th British Machine Vision Conference (BMVC)*, 2015.

[130] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Fully convolutional neural networks for remote sensing image classification. In *IEEE IGARSS*, 2016.

[131] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Trans. Geosci. Remote Sens.*, 55(2):645–657, 2017.

[132] Dimitrios Marmanis, Konrad Schindler, Jan Dirk Wegner, Mihai Datcu, and Uwe Stilla. Semantic segmentation of aerial images with explicit class-boundary modeling. In *IEEE IGARSS*, pages 5071–5074, 2017.

[133] Dimitrios Marmanis, Jan D Wegner, Silvano Gallianib, Konrad Schindler, Mihai Datcu, and Uwe Stilla. Semantic segmentation of aerial images with an ensemble of cnns. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 473–480, 2016.

[134] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE TPAMI*, 26(5):530–549, May 2004.

[135] Gellert Mattyus, Shenlong Wang, Sanja Fidler, and Raquel Urtasun. Enhancing road maps by parsing aerial images around the world. In *IEEE ICCV*, 2015.

[136] B. H. Menze, K. Van Leemput, D. Lashkari, M.-A. Weber, N. Ayache, and P. Golland. A generative model for brain tumor segmentation in multi-modal images. In *MICCAI*, pages 151–159, 2010.

[137] E. Merényi. Intelligent understanding of hyperspectral images through self-organizing neural maps. In *Proc. 2nd Int. Conf. Cybernetics and Information*

Technologies, Systems and Applications (CITSA 2005), pages 30–35, Orlando, FL, USA, 2005.

[138] Julien Michel, David Youssefi, and Manuel Grizonnet. Stable mean-shift algorithm and its application to the segmentation of arbitrarily large remote sensing images. *IEEE Tran. Geosci. Remote Sens.*, 53(2):952–964, 2015.

[139] A. Mittal, M. B. Blaschko, A. Zisserman, and P. H. S. Torr. Taxonomic multi-class prediction and person layout using efficient structured ranking. In *ECCV*, pages 245–258, October 2012.

[140] Volodymyr Mnih. *Machine learning for aerial image labeling.* PhD thesis, University of Toronto, 2013.

[141] Raul S Montero and Ernesto Bribiesca. State of the art of compactness and circularity measures. *Int. Mathematical Forum*, 4(27):1305–1335, 2009.

[142] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone. Morphological attribute profiles for the analysis of very high resolution images. *IEEE Trans. Geosci. Remote Sens.*, 48(10):3747–3762, Oct 2010.

[143] Claudia Nieuwenhuis, Eno Töppe, and Daniel Cremers. A survey and comparison of discrete and continuous multi-label optimization approaches for the Potts model. *IJCV*, 104(3):223–240, 2013.

[144] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *IEEE CVPR*, pages 1520–1528, 2015.

[145] Mathias Ortner, Xavier Descombes, and Josiane Zerubia. Building outline extraction from digital elevation models using marked point processes. *IJCV*, 72(2):107–132, 2007.

[146] Sakrapee Paisitkriangkrai, Jamie Sherrah, Pranam Janney, Van-Den Hengel, et al. Effective semantic pixel labelling with convolutional networks and conditional random fields. In *IEEE CVPR Workshops*, 2015.

[147] Guillem Palou and Philippe Salembier. Occlusion-based depth ordering on monocular images with binary partition tree. In *IEEE ICASSP*, pages 1093–1096, 2011.

[148] S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *ECCV*, 2008.

[149] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[150] P. Perona. Deformable kernels for early vision. *IEEE TPAMI*, 17(5):488–499, May 1995.

[151] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE TPAMI*, 12(7):629–639, 1990.

[152] R. Rigamonti and V Lepetit. Accurate and efficient linear structure segmentation by leveraging ad hoc features with learned filters. In *MICCAI*, pages 189–197, January 2012.

[153] T. Riklin-Raviv, K. V. Leemput, B. H. Menze, W. M. Wells, and P. Golland. Segmentation of image ensembles via latent atlases. *Med. image analys.*, 14(5):654–665, Oct. 2010.

[154] Mikael Rousson and Nikos Paragios. Shape priors for level set representations. In *In Proc. of ECCV*, pages 78–92. Springer, 2002.

[155] D. P. Roy, P. E. Lewis, and C. O. Justice. Burned area mapping using multi-temporal moderate spatial resolution data - a bi-directional reflectance model-based expectation approach. *Remote Sens. of Environment*, (83):263–286, 2002.

[156] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *ICCV*, pages 59–66, 1998.

[157] Philippe Salembier, Samuel Foucher, and Carlos López-Martínez. Low-level processing of PolSAR images with binary partition trees. In *IEEE IGARSS*, 2014.

[158] Philippe Salembier and Luis Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Trans. Image Process.*, 9(4):561–576, 2000.

[159] Philippe Salembier, Albert Oliveras, and Luis Garrido. Antiextensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, 7(4):555–570, 1998.

[160] T. Schoenemann and D. Cremers. Globally optimal image segmentation with an elastic shape prior. In *ICCV*, 2007.

[161] T. Schoenemann and D. Cremers. Globally optimal shape-based tracking in real-time. In *CVPR*, 2008.

[162] Thomas Schoenemann and Daniel Cremers. Globally optimal image segmentation with an elastic shape prior. In *ICCV*, pages 1–6, 2007.

[163] Jamie Sherrah. Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. *arXiv preprint arXiv:1606.02585*, 2016.

[164] Jianbo Shi and Jitendra Malik. Motion segmentation and tracking using normalized cuts. In *Proc. of ICCV'98*, ICCV '98, pages 1154–1160, Washington, DC, USA, 1998.

[165] Bernard W Silverman. *Density estimation for statistics and data analysis*. CRC press, 1986.

[166] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Trans. on Graphics*, 35(4):121, 2016.

[167] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[168] Ali Kemal Sinop and Leo Grady. Uninitialized, globally optimal, graph-based rectilinear shape segmentation-the opposing metrics method. In *ICCV*, pages 1–8, 2007.

[169] Amos Sironi, Vincent Lepetit, and Pascal Fua. Multiscale centerline detection by learning a scale-space distance transform. In *CVPR*, pages 2697–2704, June 2014.

[170] Greg Slabaugh and Gozde Unal. Graph cuts segmentation using an elliptical shape prior. In *IEEE ICIP*, pages 1222–1225, 2005.

[171] P. Soille. *Morphological Image Analysis*. Springer-Verlag, Heidelberg, 2nd edition, 2003.

[172] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[173] J. J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken. Ridge based vessel segmentation in color images of the retina. *IEEE TMI*, 23(4):501–509, April 2004.

[174] J. J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken. Ridge based vessel segmentation in color images of the retina. *IEEE TMI*, 23(4):501–509, April 2004.

[175] D. Stoyan, W. S. Kendall, and J. Mecke. *Stochastic geometry and its applications*. Wiley, 1987.

[176] H. Talbot and B. Appleton. Efficient complete and incomplete path openings and closings. *Image and Vision Computing*, 25(4):416–425, April 2007.

[177] H. Talbot and B. Appleton. Efficient complete and incomplete path openings and closings. *Image and Vision Computing*, 25(4):416–425, April 2007.

[178] Y. Tarabalka. *Classification of Hyperspectral Data Using Spectral-Spatial Approaches*. PhD thesis, University of Iceland and Grenoble Institute of Technology, 2010.

[179] Y. Tarabalka, L. Brucker, A. Ivanoff, and J. C. Tilton. Shape-constrained segmentation approach for arctic multiyear sea ice floe analysis. In *IGARSS*, 2012.

[180] Y. Tarabalka, J. Chanussot, and J. A. Benediktsson. Segmentation and classification of hyperspectral images using minimum spanning forest grown from automatically selected markers. *IEEE Trans. Systems, Man, and Cybernetics: Part B*, 40(5):1267–1279, Oct. 2010.

[181] Y. Tarabalka and A. Rana. Graph-cut-based model for spectral-spatial classification of hyperspectral images. In *igarss*, pages 3418–3421, July 2014.

[182] Y. Tarabalka and A. Rana. Graph-cut-based model for spectral-spatial classification of hyperspectral images. In *IEEE IGARSS*, pages 3418–3421. IEEE, 2014.

[183] Yuliya Tarabalka and James C Tilton. Improved hierarchical optimization-based classification of hyperspectral images using shape analysis. In *IEEE IGARSS*, pages 1409–1412, 2012.

[184] James Tilton and Edoardo Pasolli. Incorporating edge information into best merge region-growing segmentation. In *IEEE IGARSS*, 2014.

[185] Godfried T Toussaint. The rotating calipers: An efficient, multipurpose, computational tool. In *ICCTIM*, pages 215–225, 2014.

[186] George Trigeorgis, Patrick Snape, Mihalis A Nicolaou, Epameinondas Antonakos, and Stefanos Zafeiriou. Mnemonic descent method: A recurrent process applied for end-to-end face alignment. In *IEEE CVPR*, pages 4177–4187, 2016.

[187] F. Tsai, C.-K. Chang, and G.-R. Liu. Texture analysis for three dimension remote sensing data by 3D GLCM. In *Proc. of 27th Asian Conference on Remote Sensing*, pages 1–6, August 2006.

[188] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, September 2005.

[189] E. Türetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua. Reconstructing loopy curvilinear structures using integer programming. In *CVPR*, pages 1822–1829, June 2013.

[190] E. Türetken, G. González, C. Blum, and P. Fua. Automated reconstruction of dendritic and axonal tress by global optimization with geometric priors. *Neuroinformatics*, 9(2–3):279–302, 2011.

[191] S. Valero, P. Salembier, and J. Chanussot. Hyperspectral image representation and processing with binary partition trees. *IEEE Trans. Image Process.*, 22(4):1430–1443, 2013.

[192] Silvia Valero, Philippe Salembier, and Jocelyn Chanussot. Object recognition in urban hyperspectral images using binary partition tree representation. In *IGARSS*, pages 4098–4101, 2013.

[193] V. Vapnik. *Statistical Learning Theory*. New York: Wiley, 1998.

[194] Olga Veksler. Star shape prior for graph-cut image segmentation. In *ECCV*, pages 454–467, 2008.

[195] Y. Verdié and F. Lafarge. Detecting parametric objects in large scenes by Monte Carlo sampling. *IJCV*, 106(1):57–75, 2014.

[196] Veronica Vilaplana, Ferran Marques, and Philippe Salembier. Binary partition trees for object detection. *IEEE Trans. Image Process.*, 17(11):2201–2216, 2008.

[197] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NIPS*, pages 2692–2700, 2015.

[198] Michele Volpi and Devis Tuia. Dense semantic labeling of sub-decimeter resolution images with convolutional neural networks. *IEEE Tran. Geosci. Remote Sens.*, 2016.

[199] Yinghua Wang, Florence Tupin, and Chongzhao Han. Building detection from high-resolution polsar data at the rectangle level by combining region and edge information. *Pattern Recognition Letters*, 20, July 2010.

[200] Zhijun Wang, Djemel Ziou, Costas Armenakis, Deren Li, and Qingquan Li. A comparative analysis of image fusion methods. *IEEE Tran. Geosci. Remote Sens.*, 43(6):1391–1402, 2005.

[201] Joachim Weickert. *Anisotropic diffusion in image processing.* Teubner Stuttgart, 1998.

[202] Williams M Wells III, W Eric L Grimson, Ron Kikinis, and Ferenc A Jolesz. Adaptive segmentation of MRI data. *IEEE Transactions on Medical Imaging*, 15(4):429–442, 1996.

[203] Paul Werbos. Backpropagation through time: what it does and how to do it. *Proc. of the IEEE*, 78(10):1550–1560, 1990.

[204] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226, July 2006.

[205] R. Wolz, R. A. Heckemann, P. Aljabar, J. V. Hajnal, A. Hammers, J. Lotjonen, and D. Rueckert. Measurement of hippocampal atrophy using 4d graph-cut segmentation: Application to ADNI. *NeuroImage*, 52(1):109–118, 2010.

[206] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005, 2004.

[207] T. Xia, B. Liao, and Y. Yu. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphic*, 28(5):1–10, Dec 2009.

[208] J. A. Richards y X. Jia. Remote Sensing Digital Image Analysis. In *Springer-Verlag*, 1999.

[209] Jimei Yang, Brian Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang. Object contour detection with a fully convolutional encoder-decoder network. *arXiv preprint arXiv:1603.04530*, 2016.

[210] Mingqiang Yang, Kidiyo Kpalma, Joseph Ronsin, et al. A survey of shape feature extraction techniques. *Pattern recognition*, pages 43–90, 2008.

[211] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[212] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*. Springer, 2014.

[213] Wenzhi Zhao, Zhou Guo, Jun Yue, Xiuyuan Zhang, and Liqun Luo. On combining multiscale deep learning features for the classification of hyperspectral remote sensing imagery. *International Journal of Remote Sensing*, 36(13):3368–3379, 2015.

[214] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. Conditional random fields as recurrent neural networks. In *IEEE CVPR*, pages 1529–1537, 2015.