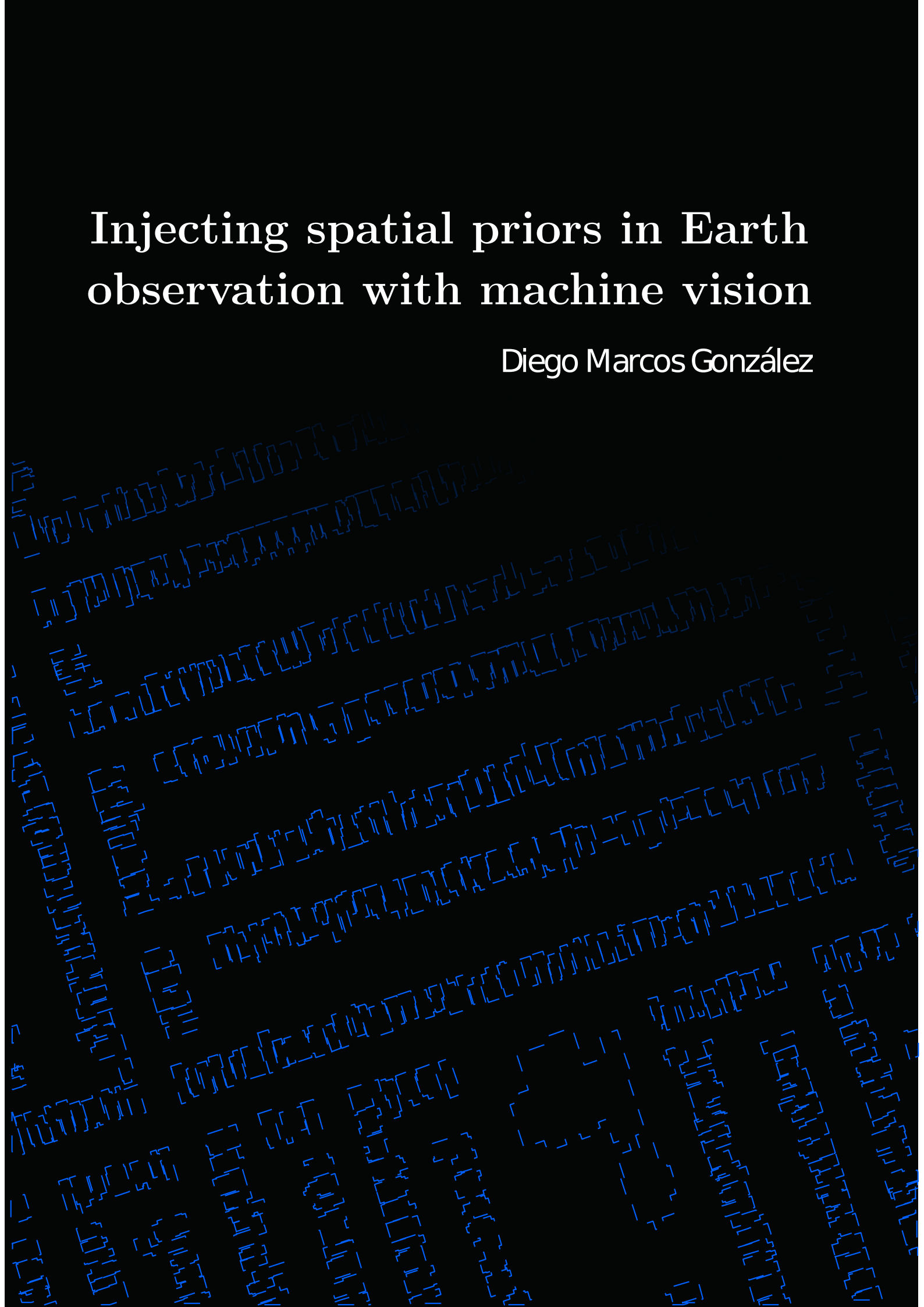
Injecting spatial priors in Earth observation with machine vision

Diego Marcos González



Propositions

1. In data-driven models, data should only be used to learn what is not known a priori.
   (this thesis)

2. The possibility to inject prior knowledge in Deep Learning is tightly related to interpretability.
   (this thesis)

3. Submission deadlines induce scientific short-sightedness.

4. Academic publishers should be non-profit, since most of their workforce is made up of volunteers.

5. It is unlikely to change someone's opinion with facts alone.

6. The price of food should reflect its environmental impact more than its production cost.

7. In a world with limited resources, ownership comes with responsibilities: taking what is about to rot might not be stealing.

Propositions belonging to the thesis entitled:

“Injecting spatial priors in Earth observation with machine vision”

Diego Marcos González
Wageningen, 11 February 2019

Injecting spatial priors in Earth observation with machine vision

Diego Marcos González

Thesis committee

Promotor
Prof. Dr D. Tuia
Professor of Geo-information Science
Wageningen University & Research

Other members
Prof. Dr E. van Henten, Wageningen University & Research
Prof. Dr R. Jenssen, The Arctic University of Norway (UiT),
Tromsø, Norway
Prof. Dr V. Lepetit, Université de Bordeaux, France
Prof. Dr K. Schindler, ETH Zurich, Switzerland

This research was conducted under the auspices of the C.T. de Wit Graduate School of Production Ecology & Resource Conservation (PE&RC)

Injecting spatial priors in Earth observation with machine vision

Diego Marcos González

Thesis
submitted in fulfilment of the requirements for the degree of doctor
at Wageningen University
by the authority of the Rector Magnificus
Prof. Dr A.P.J. Mol,
in the presence of the
Thesis Committee appointed by the Academic Board
to be defended in public
on Monday 11 February 2019
at 4 p.m. in the Aula.

Diego Marcos González
Injecting spatial priors in Earth observation with machine vision
131 pages.

PhD thesis, Wageningen University, Wageningen, NL (2019)
With references, with summary in English

ISBN 978-94-6343-560-4
DOI https://doi.org/10.18174/466814

Summary

Remote Sensing (RS) imagery with submeter resolution, or Very High Resolution (VHR), is becoming ubiquitous. Be it from satellites, aerial campaigns or Unmanned Aerial Vehicles (UAVs), this spatial resolution allows to recognize individual objects and their parts from above. This has driven, during the last few years, a big interest in the RS community on Computer Vision (CV) methods developed for the automated understanding of natural images. A central element to the success of CV is the use of prior information about the image generation process and the objects these images contain: neighboring pixels are likely to belong to the same object; objects of the same nature tend to look similar with independence of their location in the image; certain objects tend to occur in particular geometric configurations; *etc.* When using RS imagery, additional prior knowledge exists on how the images were formed, since we know roughly the geographical location of the objects, the *geospatial prior*, and the direction they were observed from, the *overhead-view prior*. This thesis explores ways of encoding these priors in CV models to improve their performance on RS imagery, with a focus on land-cover and land-use mapping.

Chapter 3 explores the use of the geospatial prior to extract features from RS images taken with different sensors that make them comparable across modalities, allowing to perform change aware multi-modal registration and thus transfer land-cover labels from one image to another. The results show an improvement over baseline invariant features both in terms of change detection and label transfer.

The overhead-view prior means that images are more isotropic than natural images, resulting in more predictable behavior with respect to rotation and lower variability in terms of apparent object shapes. These constraints have been used to improve the performance of Deep Learning (DL)-based segmentation methods for RS land-cover mapping.

Chapter 4 presents a method to allow for more control over the behavior of Convolutional Neural Networks (CNNs) with respect to rotation, allowing to obtain better results in problems with a specific expected behavior with respect to a rotation of the input image. Chapter 5 explores the application of this method to VHR aerial imagery, where it can exploit its isotropic nature and reduce the number of parameters of the model by an order of magnitude without any loss of performance.

Chapter 6 focuses on building segmentation, or building footprint extraction, using the hypothesis that building footprints are often composed of simple polygons made up of straight lines and corners. A CNN is trained, not to directly generate the segmentation map, but the energy function terms for an Active Contour Model (ACM). The experiments show that the CNN learns where in the image the polygon has to be pushed and where corners should be allowed or prevented, resulting in a substantial improvement over state-of-the-art models.

This thesis confirms that the use of domain-specific prior knowledge, in this case to Earth Observation (EO), in data driven methods can result in smaller and better performing models. In particular in the context of DL, where the complexity and "black-box" nature of the models often limit the interpretability required for the injection of prior knowledge and hamper the trustworthiness of the final results, the conclusions of this thesis point at the design of models that explicitly allow for the injection of priors as a very promising field.

Contents

Acronyms

ACM Active Contour Model.

CNN Convolutional Neural Network.

CRF Conditional Random Field.

CV Computer Vision.

DL Deep Learning.

DNN Deep Neural Network.

DSM Digital Surface Model.

DWT Deep Watershed Transform.

EO Earth Observation.

GPU Graphics Processing Unit.

GT Ground Truth.

HoG Histogram of Gradients.

ICM Iterated Conditional Modes.

IoU Intersection over Union.

K-NN K-Nearest Neighbors.

LBP Local Binary Pattern.

MAP Maximum a Posteriori.

ML Machine Learning.

MRF Markov Random Field.

nDSM Normalized Digital Surface Model.

NIR Near Infrared.

ReLU Rectifier Linear Unit.

RMSE Root Mean Square Error.

RotEqNet Rotation Equivariant Vector Field Networks.

RS Remote Sensing.

SDSN Spatial Distribution of Spectral Neighbors.

SIFT Scale Invariant Feature Transform.

SSVM Structured Support Vector Machine.

SVM Support Vector Machine.

UAV Unmanned Aerial Vehicle.

VHR Very High Resolution.

Notation

$\mathbf{a} \in \mathbb{R}^d$	A d dimensional vector
a_i	i-th element of $\mathbf{a}$
a	A scalar
$\mathcal{A}$	A vector space
$\mathbb{A}$	A set of vectors
$(\mathbf{x}^i, \mathbf{y}^i)$	Input/output pair in the training set
$\mathbb{X}$	Training set
$\mathbf{x}_i$	The i-th pixel of input image $\mathbf{x}$
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$\mathbf{z}$	Feature vector
$\mathbf{w}$	Parameter vector or tensor
$\mathbf{q}_i = (p_i, q_i)$	Cartesian coordinates of the i-th element or pixel
$\mathbf{u}_i = (u_i, v_i)$	Value of 2D vector field at the i-th element or pixel
g	Transformation
G	Group of transformations
$T(g)$	Algebraic representation of g

Chapter 1

Introduction

1.1 Introduction

The capacity of monitoring the Earth's surface through Earth Observation (EO) is of vital importance to take decisions in a multitude of critical applications, such as disaster response, conservation or understanding the evolution of agrarian and urban spaces. The use of aerial imagery for surveying and assisting in the creation of maps predates modern aviation and was initially based on the manual interpretation of photographs obtained from kites and balloons. Nowadays the availability of Remote Sensing (RS) imagery has increased to an extent that it can not all be manually interpreted: satellites provide a continuous flow of global multi-spectral imagery, often multiple times a month for a given location, at spatial resolutions down to a few meters per pixels; large scale aerial campaigns are organized by many governments every few years, providing sub-meter resolution imagery, also known as Very High Resolution (VHR), over whole countries; and the plummeting costs of Unmanned Aerial Vehicles (UAVs) mean they are being increasingly deployed, typically providing each time a few square kilometers of imagery at decimeter resolution. Although advances in photogrammetry and image processing have allowed to assist the photo-interpreters in coping with this data deluge by providing them with orthorectified and radiometrically corrected imagery, the semantic interpretation of the content of the images is still heavily dependent on human expertise to interpret VHR imagery.

Automatized Machine Learning (ML) methods have been used to scale some manually interpreted samples up to large regions by interpolating to unobserved areas based on lower resolution satellite imagery (Hansen et al., 2013; Pengra et al., 2015). This can be done in cases where the relationship between the input data and the output is not too complex, such as using multi-spectral Landsat imagery, with 30 meter resolution, to classify pixels into simple land-cover classes, as can be vegetation, bare soil and water (see Figure 1.1). At such resolution, one pixel often contains the mixed spectral responses of multiple objects, with the effects of shadows, occlusions and geometrical configurations averaged out. This invariance of the data to geometric details is helpful when the aim is to classify into those simple classes that are highly correlated with the spectral response.

When higher spatial resolution is required, or when dealing with semantically more complex classes, for instance vehicle, road and building, it is often required to use VHR images. Due to trade-offs in sensor design, this imagery tends to be of lower spectral resolution, such as Red-Green-Blue (RGB). This simultaneous decrease in spectral information and increase in the task complexity make it often unfeasible to assign pixels to the correct class by merely using the color, or spectral, information, as shown in Figure 1.2.

The limited information provided by each individual pixel is compensated by the fact that we often have multiple pixels from each object. This richness allows to analyze the geometry of the object and how it relates to neighboring objects. Humans can easily perform

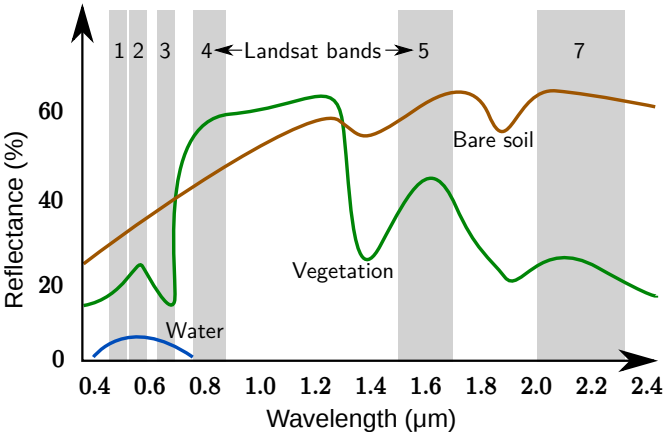


Figure 1.1: Healthy vegetation tends to have a much higher reflectance in the Near Infrared (NIR) region of the spectrum (Landsat band 4) than in the red region (band 3). This is often sufficient to distinguish a forested pixel from a bare soil one.

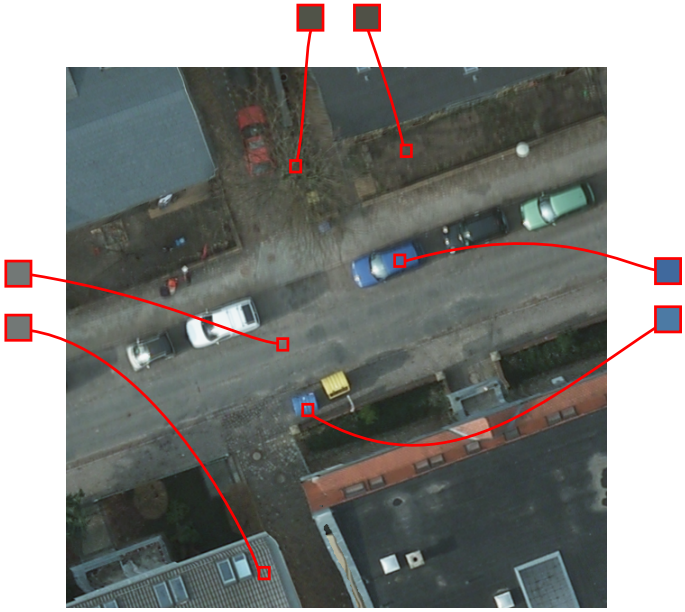


Figure 1.2: This VHR aerial image corresponds to one Landsat pixel, or 30×30 m. At that resolution, it would look like a generic built-up or bare soil pixel. On the other hand, the 5 cm resolution of the VHR aerial imagery allows to see the complexity of the scene. At the same time, the color information from the three RGB bands would be insufficient to distinguish any of the marked pairs of pixels even though they belong to different classes of objects. To solve this, we need to use the geometry and the context.

this task and understand to which category belong the pixels marked in Figure 1.2. For instance, we might not be sure whether the blue object is a trash container or not, but the presence of a yellow object, probably also a container, can help us with that decision. Depending on our experience, we might go a step further and infer that it is a paper recycling container. Even though this might seem like an easy task, our brain makes use of a phenomenal amount of prior knowledge, or priors, about the reality and how images are captured: nearby pixels are likely to belong to the same object, mostly if they look similar; a change in the image due to a change in the camera does not mean something has changed in the scene; objects can cast shadows and occlude other objects; cars tend to be on roads; buildings tend to have polygonal footprints; *etc.*

Computer Vision (CV) aims to achieve automatic image understanding and, as in human vision, the mentioned prior knowledge of how images are generated and how objects relate to each other is an essential part of the field. There is a variety of tasks studied in this field, such as object detection and tracking, image classification, pose estimation, 3D

reconstruction, and many more.

In this thesis the focus is on the task of image *semantic segmentation* (see Section 2.1), which consists of assigning a class label to each pixel in the image, producing a class map as a result. In the EO field of RS this is often referred to as *land-cover mapping*, when the classes relate to the materials on the surface, or as *land-use mapping*, when they relate to which human activities are associated to that location.

1.2 Objectives

This thesis explores the use of additional EO-specific prior knowledge, tailored for CV tasks on overhead imagery, that is based on the particular geometric constraints associated with this kind of data. In particular, the aim is to use these priors to improve the performance on semantic segmentation, either by reducing the amount of hand-labeled examples (or Ground Truth (GT)) required, reducing the computational load of the segmentation methods or improving the geometry of the resulting maps. Section 1.3.3 presents the intuitions behind the priors stemming from the nature of RS imagery and Section 1.4 presents the publications that have arisen from studying the injection of these priors in the context of this thesis.

1.3 Prior knowledge

This section introduces the concept of prior knowledge in the general context of ML and in the more specific contexts of CV and EO.

1.3.1 Machine Learning: the learned and the given (prior knowledge)

ML refers to a family of algorithms in which the ability to fulfill their task is learned from data. When enough training data is available, ML methods often obtain substantially better performances than manually coded methods. Nonetheless, ML algorithms are never fully data driven; instead, a scaffolding needs to be manually designed, a step usually referred to as *model design*. This scaffolding contains the learnable parameters, whose values are determined during the training process using the data, and defines how they relate to each other. In contrast to the model's learnable parameters, the elements involved in the model design consist of imposed constraints to the solution that define the family of models to use and other model design decisions that are taken before even looking at the data. For instance, if we want to model the distribution of the size of some animal population, we might decide to fit a unimodal, bimodal or multi-modal normal

distribution, depending on our knowledge about sexual dimorphism and age distribution. The particular values of the mean and standard deviations of the distributions are then set by using the data. Note that, in the absence of such knowledge, we could still model all possible hypotheses. Nevertheless, this would come at the cost of requiring more computational resources, exponentially many with the number of dimensions in the hypothesis space. Well chosen design decisions based on prior knowledge can substantially improve the trade-off between performance and cost.

1.3.2 Computer Vision and image-related prior knowledge

The field of CV deals with the automatic high-level understanding of images, videos and, more in general, spatially arranged data, such as medical scanner data or 3D point clouds. Many CV approaches can be framed as ML problems in which the use of priors is particularly important due to the very high dimensionality of each data point, since a single image can consists of millions of pixels, and the strong assumptions that can be made on the process of image formation and the observed objects: nearby pixels are more likely to belong to the same object, inducing spatial autocorrelation and thus allowing for sparse representations; camera position does not affect the nature of imaged objects, making most problems invariant or equivariant to translation; as well as other more specific priors originating from the spatial nature of images, such as expected geometries and relative positions of objects. The exploitation of the first two of these priors can be seen in most recent CV models in the form of a convolutional or patch based feature extraction step.

1.3.3 Prior knowledge in Earth Observation

Recent advances in CV, mostly driven by a series of breakthroughs in Convolutional Neural Networks (CNNs), have opened up the possibility of automatizing the processes of interpreting high resolution (sub-meter) EO imagery (Mnih, 2013; Volpi and Tuia, 2017; Maggiori et al., 2017). These models extract a hierarchy of features (see Section 2.3.3) using convolution operators, which heavily rely on the priors mentioned in the previous section. Although exploiting these priors in such way has been indeed very successful in EO, even stronger assumptions can be made than in the general CV setting. This is due to the additional knowledge we hold on how the images were acquired:

1. **Geospatial prior**: knowing the pose of the sensor at the moment of the acquisition allows to assign a location on the Earth's surface to each pixel in the image, generally with a precision of a few meters. This prior, illustrated in Figure 1.3, will be referred to as *geospatial correspondence*.

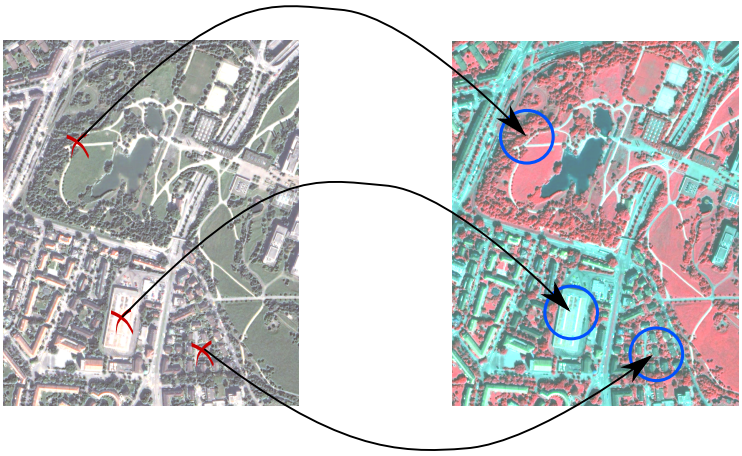


Figure 1.3: Even before we perform any analysis on these two images, obtained over the same area with different sensors and years apart, we know that the locations marked with crosses on the left correspond to some point within the circles on the right.

2. **Overhead-view prior**: observing the Earth's surface from a overhead-view perspective makes the images much more isotropic than natural images, which often display different statistics in the vertical and horizontal directions. This naturally calls for models that are *invariant or equivariant* (see Section 2.4.4) to changes in orientation of the input image. In addition to that, objects are often seen in less diverse poses, since many, such as buildings and vehicles, almost always show the roof to the sensor, meaning that only rotations around the vertical axis are typically observed. This is an invitation to inject *priors on the expected shapes* of the observed objects, eased thanks to the lower shape complexity compared to what can be expected in natural images (see Figure 1.4).

Figure 1.4: Top row: cars seen from a ground level perspective. There is a large variety in terms of perceived poses, scales and occlusions. Cars appear to always have the top pointing upwards and the wheels underneath, resulting in a very clearly dominant absolute orientation. **Bottom row:** cars seen from an overhead-view in an aerial image dataset. In this case, the *apparent size and shape of cars is much less diverse*, making it easier to use assumptions on their apparent shape. Their *absolute orientation is arbitrary*, with no dominant direction, which means that a CV model should be able interpret these images in the same way with independence of their absolute orientation.

1.4 Contributions

The contributions of this thesis are aimed at providing better CV models for the semantic segmentation of VHR overhead imagery. This is done using prior knowledge on how these images are generated. Publications produced in the context of this thesis are introduced in this section according to the nature of the prior knowledge they exploit:

Geospatial correspondence: Geo-referencing allows for time-related priors to be applied, even across multiple sensors and long time spans. Combining the prior that two objects that are similar in one domain (certain sensor and acquisition conditions) are also likely to be similar in another domain, together with the assumption that the relationship between unchanged areas is more consistent that between changed areas, it is possible to extract domain invariant features. These features can then be used to register images from different domains (known as heterogeneous image registration, which consists of finding which locations in different images correspond to the same objects), detect changes and transfer class labels from one domain to the other. This is exposed in Chapter 3 and based on:

> (Marcos et al., 2016a) Marcos, D., Hamid, R., and Tuia, D. (2016a). Geospatial correspondences for multimodal registration. In *Proceedings of the CVF/IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rotation equivariance: The known absence of any dominant absolute orientation in RS images (see Figure 1.4) makes rotation invariance or equivariance (see Section 2.4.4) more desirable than in natural images. The injection of translation equivariance by using convolution operators has been acknowledged as an element of vital importance on many CV systems and additional advantages can be expected from injecting also rotation equivariance. The close relationship between the arrangement of pixel data and translation renders the exploitation of translation equivariance much more straightforward than in the case of rotation, making it more challenging to obtain models that can profit from the latter. Chaper 4 explores a method for building deep CNNs based on a novel and efficient rotation equivariant operator. The chapter in based on:

> (Marcos et al., 2017) Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. (2017). Rotation equivariant vector field networks. In *Proceedings of the CVF/IEEE International Conference on Computer Vision (ICCV)*.

A study using only shallow CNN architectures was presented in:

> (Marcos et al., 2016c) Marcos, D., Volpi, M., and Tuia, D. (2016c). Learning rotation invariant convolutional filters for texture classification. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*.

An application of this method to land-cover mapping based on VHR imagery is presented

in Chapter 5, also to be found in:

> (Marcos et al., 2018c) Marcos, D., Volpi, M., Kellenberger, B., and Tuia, D. (2018c). Land cover mapping at very high resolution with rotation equivariant cnns: Towards small yet accurate models. *ISPRS Journal of Photogrammetry and Remote Sensing*.

An additional study applying this concept to scale can be found in:

> (Marcos et al., 2018a) Marcos, D., Kellenberger, B., Lobry, S., and Tuia, D. (2018a). Scale equivariance in cnns with vector fields. In *FAIM/ICML Workshop on Towards learning with limited labels: Equivariance, Invariance, and Beyond*.

Simple shape priors: The overhead-view setting reduces the complexity of possible object appearances, since objects are arranged on a practically two dimensional space and are less prone to complex occlusions and poses. For instance, building rooftops are often formed by straight lines and corners, helping in the process of inferring their footprint, which can be described in terms of closed polygonal shapes. Therefore, when training a model to perform automatic building footprint extraction, it can be of help to restrict the solution space to the family of closed polygons and define where there should be corners and where straight lines. This is explored in Chapter 6 and published in:

> (Marcos et al., 2018b) Marcos, D., Tuia, D., Kellenberger, B., Zhang, L., Bai, M., Liao, R., and Urtasun, R. (2018b). Learning deep structured active contours end-to-end. In *Proceedings of the CVF/IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

A preliminary study exploiting the relative simplicity of observed geometries from an overhead-view perspective by using ground truth patches as jigsaw puzzle pieces to perform land-cover mapping was presented as:

> (Marcos et al., 2016d) Marcos, D., Volpi, M., and Tuia, D. (2016d). Solving structured segmentation of aerial images as puzzles. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*.

Chapter 2

Background

2.1 Semantic segmentation of VHR imagery

The main motivation behind this thesis is the improvement of *semantic segmentation* methods using RS images by harnessing constraints that are specific to this kind of imagery.

In CV the expression *semantic segmentation*, also called *semantic labeling* in the RS community, refers to the partition of an image $\mathbf{x} \in \mathbb{R}^{M \times N \times d}$, with $d = 3$ in the case of RGB images, into semantically coherent regions to form a segmentation map $\mathbf{y} \in \mathbb{R}^{M \times N}$, meaning that one class label is assigned to each image pixel. In contrast to low-level image segmentation, which consists of finding visually coherent regions in an unsupervised way, semantic segmentation aims at grouping the pixels of the image that belong to the same semantic class, generally in a supervised way by using GT segmentation examples in the form of image-segmentation pairs $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{X}$, where $\mathbf{x}_i$ is an image in the GT and $\mathbf{y}_i$ the associated segmentation example.

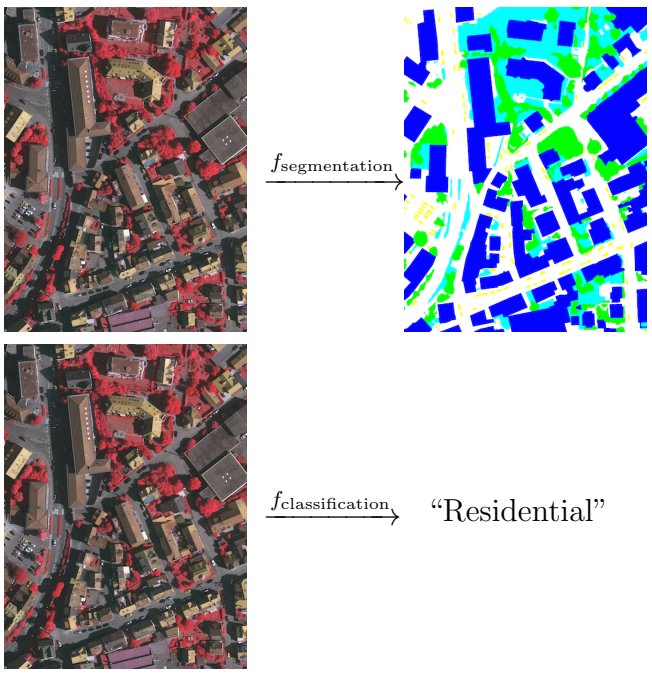


Figure 2.1: The CV task of semantic segmentation consists of obtaining a function f that, given an image, produces an output map that assigns a semantic class, among a predifined set of classes, to each pixel in the image. In the context of RS this can be used for tasks such as land-cover and land-use mapping. **Top:** In this example, an aerial image has been segmented into the following classes: building (blue), road (white), car (yellow), grass (cyan) and trees (green). Each pixel in the image is assigned a class label. **Bottom:** The related task of image classification takes as input an image and produces a single class label, or class probability, that is assigned to the image as a whole, forgoing the spatial component in the output.

When the size of the objects of interest is similar or smaller than the size of the image pixels, such as when observing cities and forest at a resolution of multiple tens of meters, it is common to classify each pixel individually. This would remove the spatial component in the input and make the process of semantic segmentation equivalent to performing pixel classification (Section 2.3.2) multiple times. In fact, the expressions *image classification* and *semantic labeling* are often used as synonyms of semantic segmentation in RS, while

in CV image classification implies a single label per image and segmentation, a label per pixel (see Figure 2.1).

As introduced in Section 1.1, when the resolution of the imagery increases in such way that each object encompasses multiple pixels, as is typically the case in VHR imagery, it becomes important to use the properties of the pixel's neighborhood to assign it to a semantic class. This has been exploited by using textural descriptors (Regniers et al., 2016), Bag of Visual Words (BoVW) (Gueguen, 2015), random fields to encourage consistency in the output map (Moser et al., 2013) or object-based image analysis (Blaschke, 2010), which consists of presegmenting the image in an unsupervised way into superpixels and then using the properties of the superpixels, such as shape, texture and other statistics, for classification. These methods, based on the classical feature extraction and classification paradigm mentioned in Sections 2.3.1 and 2.3.2, have been giving way to deep CNN based methods (Kampffmeyer et al., 2016; Volpi and Tuia, 2017; Maggiori et al., 2017), in which the feature extractor and classifier are jointly learned (see Section 2.3.3).

2.2 Machine Learning for Computer Vision

ML refers to algorithms in which part of the behavior is determined by a dataset, called *training set*. A ML algorithm implements a function $\mathbf{y} = f(\mathbf{x})$ that takes an input vector $\mathbf{x}$ to produce an output $\mathbf{y}$. Given some input $\mathbf{x} \in \mathcal{X}$, where $\mathbf{x}$ is some measurement instance, generally a vector, and $\mathcal{X}$ the space of all possible inputs, we want to infer some aspects of the state of the reality $\mathbf{y} \in \mathcal{Y}$ that generated that data. One way of approaching this is to design a function $f : \mathcal{X} \to \mathcal{Y}$ that provides the desired output given the input data.

Since the measurement of $\mathbf{x}$ can be noisy and multiple states could, in principle, result in the same measurement, these methods often aim at obtaining the *posterior probability distribution* $p(\mathbf{y}|\mathbf{x}; \mathbf{w})$ over the possible outputs for the given data, where $\mathbf{w}$ are the parameters that describe the probability function. Because we are often only interested in the most likely output, or Maximum a Posteriori (MAP) solution, it is often more tractable to bypass computing the full distribution $p(\mathbf{y}|\mathbf{x}; \mathbf{w})$ and obtain directly the MAP solution: $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}|\mathbf{x}; \mathbf{w})$.

CV aims at building methods for automatic image understanding, gathering useful information about the world from images. Therefore, $\mathbf{x}$ is typically an image and $\mathbf{y}$ can be, for instance, the type of objects present in the image, their relative pose or the material they are made of.

Most recent CV methods are based on ML and the process to obtain the result is often split in the following steps:

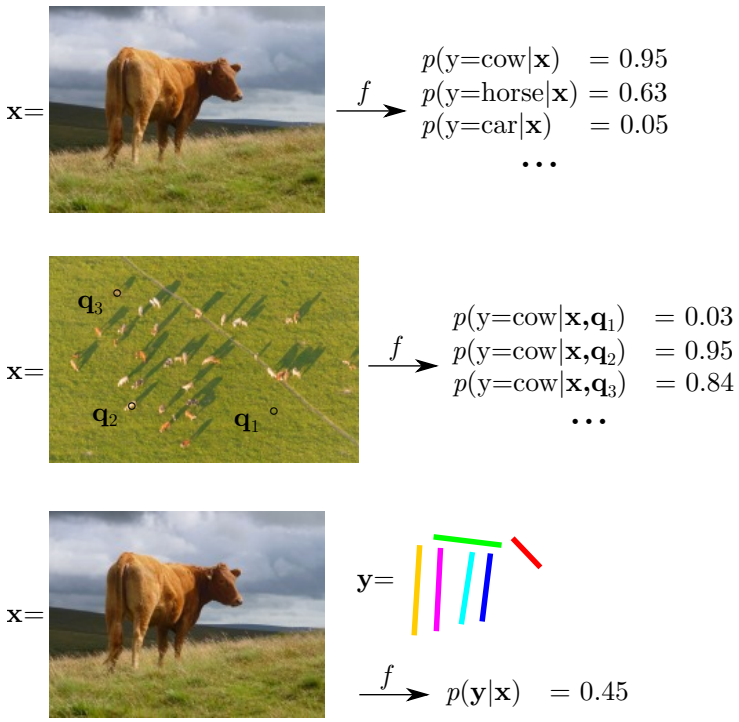


Figure 2.2: Given some image $\mathbf{x}$ and some output hypothesis $\mathbf{y}$, many CV problems can be posed as estimating the probability of all possible $\mathbf{y} \in \mathcal{Y}$ conditioned on the input $\mathbf{x}$. The example on the top corresponds to *image classification*, where the possible outputs consist of a list of considered classes. In the middle, in a problem called *object detection*, the location of the object is also accounted for. The space of possible outputs can be more complex than a list of classes. The bottom figure illustrates the example of *pose estimation*, where the output is formed by multiple interconnected components, and Figure 2.1 shows how the output can be a map conditioned on the image.

- *Model design:* The selection of the family of possible functions $f \in \mathcal{F}$, parametrized by $\mathbf{w} \in \mathcal{W}$, from $\mathcal{X}$ to $\mathcal{Y}$, allows us to encode our prior knowledge of the problem. This traditionally includes the choice of a feature representation, discussed in Section 2.3.1; a final task model, such as a classifier, a few of which are discussed in Section 2.3.2; and some information about preferred configurations of the output $\hat{\mathbf{y}}$, seen in Section 2.4. Although these three elements are clearly distinct in many traditional CV pipelines, the boundaries between them tend to be more blurred under the Deep Learning (DL) paradigm, introduced in Section 2.3.3. These elements may include learnable parameters, which we will group into the parameter vector $\mathbf{w}$.

- *Learning:* Also called training. In *supervised learning* the data consists of input/output pairs, or GT, the parameters $\mathbf{w}$ are learned given a training GT dataset $(\mathbf{x}^i, \mathbf{y}^i) \in \mathbb{X}, i = [1, \ldots, n]$ such that f performs the task well. This corresponds to finding the $\mathbf{w}$ that maximizes the probability of the observed output given the observed data. This step therefore consists of solving the inverse problem:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \sum_{i}^{n} p(\mathbf{y}^i | \mathbf{x}^i; \mathbf{w}). \tag{2.1}$$

  The definition of what performing the task well means is generally captured in the form of a loss function, or cost. In a supervised learning setting, the loss function

$l(\hat{\mathbf{y}}^i, \mathbf{y}^i)$ outputs a low value when the $\hat{\mathbf{y}}^i = f(\mathbf{x}^i)$ is close to the desired value $\mathbf{y}^i$, and thus the learning step consists of solving the optimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{i}^{n} l(\hat{\mathbf{y}}^i, \mathbf{y}^i). \tag{2.2}$$

In the case of *unsupervised learning* (which is not the focus of this thesis), the loss function does not depend on a GT output and is typically of the form $l(\hat{\mathbf{y}}^i)$.

- *Inference:* Once the model parameters $\mathbf{w}$ are chosen it becomes possible to obtain estimates of the posterior distribution or to draw a MAP sample from $\mathcal{Y}$ given a new input image $\mathbf{x}$, a process called inference or prediction:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}|\mathbf{x}; \mathbf{w}). \tag{2.3}$$

Ideally, the result of running inference on a new $\mathbf{x}$ will provide a useful result $\hat{\mathbf{y}}$. In this case, we say the model *generalizes* well to unseen data. If the model is only able to predict well the samples in the training set, but does not generalize to new samples, we say that the model *overfits* to the training set.

2.3 Classification with Machine Learning

Given some image data and a final ML task (this section focuses on classification, but could be others, such as object detection or semantic segmentation), many traditional CV methods use a two-step approach: first, a function f_1 is designed (sometimes involving unsupervised learning), to extract features from the data. Then a second function f_2 takes these features as input and uses them to perform the final task, also called downstream task (see Figure 2.3). Section 2.3.1 introduces the concept of feature extraction and some feature descriptors used in this thesis. The classification step is introduced in Section 2.3.2, and 2.3.3 presents the idea behind DL, in which f_1 and f_2 are jointly learned as a single function.

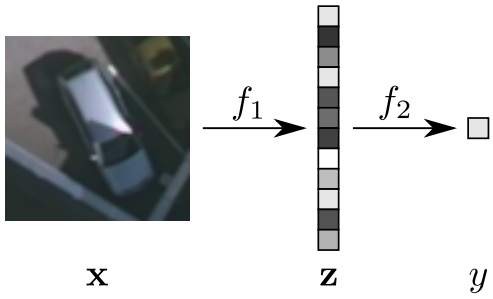


Figure 2.3: In traditional CV pipelines, a function f_1 is designed to extract a feature vector $\mathbf{z} = f_1(\mathbf{x})$, where each gray square represents a scalar, from the input $\mathbf{x}$. A second function f_2 is then designed and trained on a training dataset to produce the final output y, which can be, for instance, a classification score for the class "car".

2.3.1 Feature representation

The first step in most pipelines is *feature extraction*, also called descriptor extraction. The original feature representation, *e.g.* pixel values, is often not well suited for the downstream task. For instance shadows and acquisition conditions can greatly influence the pixel values of an image, but are often irrelevant for the task. Feature extraction techniques are reared towards building a function f_1: $\mathbf{z} = f_1(\mathbf{x})$, such that $\mathbf{z}$ is more useful, which could mean being invariant to shadows and other irrelevant variations in the image $\mathbf{x}$ while capturing the factors of variation that are relevant to the final task.

Handcrafted features

The traditional approach to building f_1 is to carefully design it to capture our prior knowledge about the data and the problem. This section presents a few examples of features mentioned in the thesis.

Local Binary Patterns (LBPs) (Ojala et al., 1996) were developed for texture classification and exploit the rotation and brightness invariance associated to textures. At a given location on an image, a LBP is extracted by checking the image intensity at a number of points arranged in a circle around the location. If the value is higher than at the center, a one will be written into an intermediate feature vector, and a zero otherwise. The final feature is usually a histogram of occurrence of each possible LBP in each local image window.

Histogram of Gradients (HoG) (Dalal and Triggs, 2005), originally designed also for texture classification, then adapted to object detection, is based on counting local image gradient directions, therefore being invariant to brightness and small translations.

The Scale Invariant Feature Transform (SIFT) descriptor (Lowe, 2004) is similar in nature to HoG, but follows a more complex pipeline, including multiscale extraction and orientation estimation, allowing for representations that are rotation and scale invariant.

Another approach for obtaining invariance to appearance while keeping information on shape and spatial arrangement is the Local Self-Similarity (LSS) descriptor (Shechtman and Irani, 2007). For each image location this descriptor is built by computing the similarity between a patch centered in the location and multiple patches around it. This provides a local spatial pattern of similarly looking patches which is independent on the actual appearance of the patch. A descriptor following this logic, but making use of geospatial correspondences, is presented in Chapter 3.

Data-driven features

Unlike handcrafted features, data-driven features undergo a preliminary training step to learn the parameters that define the way in which the features are extracted. These features are thus adapted to the data distribution, potentially making them more compact and useful for the final task.

A simple way of building data driven features is to find representative examples in the training set and verify whether a similar pattern occurs in a test sample. This is the idea behind features based on Textons (Cula and Dana, 2001; Leung and Malik, 2001) and BoVW (Yang et al., 2007), which rely on templates computed as the centroids of a clustering algorithm partition of image patches. Dictionary learning methods (Kreutz-Delgado et al., 2003) can be used to learn an overcomplete basis from the data that allows to represent it as a sparse vector that can approximately reconstruct the original signal and be useful for other downstream tasks, such as classification (Mairal et al., 2009).

2.3.2 Classification

In traditional CV pipelines, the features obtained in the feature extraction step are used in a final task such as classification: $\mathbf{y} = f_2(\mathbf{z})$.

Classification is the problem of assigning a data point $\mathbf{x}^i \in \mathcal{X}$ (the input data are often vectorial, $\mathcal{X} = \mathbb{R}^d$) to a class $y^i \in \mathcal{Y} = \{1, 2, \ldots, C\}$ in a predefined set of classes $\mathcal{Y}$, being the total number of classes $C = |\mathcal{Y}|$. It is a supervised learning task, meaning that we have access to a training dataset $\mathbb{X} = \{(\mathbf{x}^i, y^i)\}_{i=1}^n$ formed of n pairs of data points $\mathbf{x}^i$ and labels y^i. The training set $\mathbb{X}$ is used to divide the input space $\mathcal{X}$ into decision regions, separated by decision boundaries. These boundaries can then be used during inference to assign a class label to any newly observed $\mathbf{x} \in \mathcal{X}$.

In the following, three commonly used classification algorithms, all of them mentioned throughout this thesis, are presented:

K-Nearest Neighbors

The K-Nearest Neighbors (K-NN) classifier, the epitome of non-parametric models, works by storing the whole training set $\mathbb{X}$ in memory and assigning to a new data point $\mathbf{x}$ the most common class among the k nearest points $N_k(\mathbf{x}) \in \mathbb{X}$. Section 2.4 explains how prior knowledge on class abundance can be used by splitting this process in two steps: 1) inference, which consist in obtaining a score vector $\mathbf{s} \in \mathbb{R}^C$ by assigning the score $s_c = \sum_{i \in N_k} (y^i = c), \forall c \in \mathcal{Y}$ and 2) decision, generally selecting the class with the highest score, $y = \arg\max \mathbf{s}$.

Decision Trees and Random Forest

Instead of keeping the whole training set in memory, a decision tree can be used to find a hierarchical set of rules, similar to the dichotomous keys used in taxonomy. The rules are built greedily and in a recurrent fashion. Starting with the whole dataset (root), one or more dimensions are explored in the search for a good split, where goodness is usually defined as some class purity measure. This divides the dataset in two (leaves) and the same procedure is applied to each sub-dataset. This is repeated until some stopping criterion is met, such as the maximum tree height or minimum number of data points per leaf. Decision trees can approximate any decision boundary, given enough data, and don't require the data to be scaled or normalized in any particular way. On the down side, they easily overfit the data and often do not perform very well in practice. Many ensemble methods based on decision trees have been proposed. The most popular method is Random Forest (RF) (Breiman, 2001), which consists of learning multiple decision trees, each on a different bootstrapped subset of the training set, to make sure that each tree will learn a different decision boundary, and averaging the results of all the trees together to get the final result. These methods are still efficient to train and often provide satisfactory results, making them a popular choice for classification, together with Support Vector Machines (SVMs), described next.

Support Vector Machine

Linear binary classifiers, with $\mathcal{Y} = \{-1, 1\}$, are aimed at finding a hyperplane (the generalization of the line in 2D space and the plane in 3D space to spaces of higher dimensionality) that separates the training samples of both classes, thus restricting the possible decision boundaries to the family of hyperplanes in the input feature space. The SVM is aimed at finding the separating hyperplane that provides the largest margin. This means that it finds the data points that are closest to the decision boundary, also called support vectors, and then finds the hyperplane that maximizes the distance between the decision boundary and the support vectors. This has been shown to ensure good generalization to unseen samples (Cortes and Vapnik, 1995).

A hyperplane in $\mathbb{R}^d$ can be parametrized by a vector $\mathbf{w}$ orthogonal to the hyperplane and a bias w_0, such that it consists of the points that satisfy $\mathbf{w}^\top \mathbf{x} + w_0 = 0$. If the dataset is separable, the objective of binary classification is to find the hyperplane that perfectly separates the training set. In addition, we would want to leave a margin between the positive and the negative samples such that they don't lie right next to the hyperplane. This can be translated as finding $\mathbf{w}$ and w_0 such that, for each training sample $(\mathbf{x}^i, y^i)$, we observe:

$$y^i = \begin{cases} 1, & \text{if } \mathbf{w}^\top \mathbf{x}^i + w_0 \geq 1 \\ -1, & \text{if } \mathbf{w}^\top \mathbf{x}^i + w_0 \leq -1. \end{cases} \tag{2.4}$$

The distance between the two hyperplanes defining the margin in Equation (2.4) can be shown to be $\frac{2}{\|\mathbf{w}\|}$, meaning that finding the max-margin hyperplane consist of solving $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{w}\|^2$ subject to Equation (2.4).

In order to relax the requirement that the data be linearly separable, the hard constraint in Equation (2.4) can be relaxed using the *hinge loss*. This loss is zero if the constraint is satisfied and proportional to the violation if the constrained is violated. This leaves us with the problem:

$$(\hat{\mathbf{w}}, \hat{w}_0) = \arg\min_{\hat{\mathbf{w}}, \hat{w}_0} \Big[\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \max(0, 1 - y^i(\mathbf{w}^\top \mathbf{x}^i + w_0)) \Big], \tag{2.5}$$

where C is a constant weighting the two terms.

Research on how to extend this formulation for building footprint extraction is presented in Chapter 6.

2.3.3 Deep Learning: joint feature extraction and classification

Deep Neural Networks (DNNs) are used to learn the parameters of sets of hierarchical features *together* with the final task, something referred to as *end-to-end* learning. In contrast to the two step pipeline common to methods reviewed so far in this section, where the feature representation is designed or learned first and then the final task uses these features, DNNs use *backpropagation* (Rumelhart et al., 1986) to optimize the whole pipeline jointly to solve the final task. Backpropagation allows to learn a deep and complex hierarchy of features that can solve tasks requiring to fit high-dimensional non-linear functions. DNNs are built by composing a set of simple functions (Rumelhart et al., 1986), often linear mappings, with learnable parameters, and simple nonlinearities, such the sigmoid or the Rectifier Linear Unit (ReLU). A feed forward DNN, with input $\mathbf{x}$ and output $\mathbf{y}$, which can be for instance a classification score or a regression prediction, can be expressed as a composition of L functions $\mathbf{z}_l = f_l(\mathbf{z}_{l-1}; \theta_l), \quad l \in 1 \dots L$, also called layers, where $\mathbf{z}_{l-1}$ and θ_l are respectively the input and parameters of the function f_l, while $\mathbf{z}_l$ is its output. To simplify the notation, this will be written $\mathbf{z}_l = f_{\theta_l}(\mathbf{z}_{l-1})$. A DNN model can therefore be expressed as:

$$\mathbf{y} = f_{\theta_L}(f_{\theta_{L-1}}(\dots f_{\theta_1}(\mathbf{x}) \dots)) \tag{2.6}$$

In addition to this, DNNs often enjoy better generalization to unseen samples than other families of models, even if their capacity would in principle allow them to overfit to very large training sets (Zhang et al., 2016a).

Convolutional Neural Networks

Although DNNs enjoyed some success in multiple fields during the 1990s, they did not become popular methods in Computer Vision until the early 2010s, when a few methodological innovations, such as the efficient ReLU non-linearity (Glorot et al., 2011), and the adaptation of Graphics Processing Units (GPUs) to scientific computing allowed models based on deep CNNs to substantially improve the state-of-the-art in image classification (Krizhevsky et al., 2012) and even surpass human performance (Cireşan et al., 2012). Nowadays, CNNs are the workhorse of deep learning based CV. They are particularly suited to image data by making use of the 2D arrangement of image pixels. This in done by extracting features in a convolutional manner, meaning that the same local feature is extracted at every location in the image or input feature map to create an output feature map with the same 2D arrangement. This allows for a major reduction in the number of required learnable parameters through weight sharing and encodes, to some extent, a translation equivariant behavior in the model, as will be seen in Section 2.4.4.

For more details on the most common building blocks of CNNs, please refer to Section 5.2.1.

2.4 Prior knowledge in Computer Vision

This thesis focuses on the use of prior knowledge that stems from the nature of EO data in CV models. This section introduces the concepts and methods related to the injection of priors in CV relevant to this thesis.

Prior knowledge, or just priors, is some information held about the object of a ML problem in addition to the data itself. Encoding these priors in the models before learning has been shown to be vital in enabling a good performance in many cases, since many ML problems would be ill-posed without them. Some examples are the regularization term used in SVMs or the translation equivariance encoded in CNNs.

2.4.1 Priors in the Bayesian sense

The score vector $\mathbf{s} = f(\mathbf{x})$ provided by a classifier gives us some information that can be thought of as the trustworthiness of the classifier's outcome; if the scores of more than one class are similarly high, we might not want to trust the classification decision as much as when only one class has a non-zero score. In order to quantify this trustworthiness it can be useful to interpret this scores as probabilities. Depending on how they are normalized we could interpret them in two ways:

- The posterior probability $p(y_c|\mathbf{x})$, estimated as $\frac{s_c}{\sum_i s_i}$. Since $\mathbf{s}$ is non-negative, and this normalization makes sure that its C components sum to one, this can be interpreted as the probability of $\mathbf{x}$ belonging to class y_c.
- The likelihood $p(\mathbf{x}|y_c)$, estimated as $\frac{s_c}{Z}$, with $Z = \int_{\mathbf{x}\in\mathcal{X}} f_c(\mathbf{x})$ being the partition function. This can be interpreted as the probability of observing $\mathbf{x}$ given that the class is y_c. Note than computing Z would require to evaluate the classifier in every point of the input space $\mathcal{X}$, making it unfeasible to compute in many cases. As we will see in Section 2.4.2, computing Z is not required in the case where we want to compare the likelihood of two different labelings, since Z depends only on the data and not on the labeling itself.

Interpreting classification scores directly as a posterior probability makes the classification decision trivially simple, since $\arg\max_c p(y_c|\mathbf{x}) = \arg\max_c s_c$. However, this does not allow to inject additional information we might have about the problem, apart from the scores. Let us use a K-NN classifier as an illustrating example. As we have seen in Section 2.3.2, K-NN is generally used to build a set of scores that are interpreted as an estimation of the posterior probability $p(y_c|\mathbf{x})$:

$$p(y_c|\mathbf{x}) = \frac{K_c}{K}, \tag{2.7}$$

which is the proportion of neighboring training points that belong to class y_c.

Instead, we can estimate the class-conditioned likelihood:

$$p(\mathbf{x}|y_c) = \frac{K_c}{N_c V}, \tag{2.8}$$

where V is the volume of the minimal sphere centered in $\mathbf{x}$ and containing all of its K nearest neighbors and N_c is the total number of training samples with class y_c. This is an estimation of the local class-conditioned density. If we also compute the unconditional likelihood of $\mathbf{x}$, or local density:

$$p(\mathbf{x}) = \frac{K}{NV}, \tag{2.9}$$

we can use Bayes' theorem to connect the estimates in Equation (2.7) and Equation (2.8), and obtain the posterior using the likelihood:

$$p(y_c|\mathbf{x}) = \frac{p(\mathbf{x}|y_c)p(y_c)}{p(\mathbf{x})}. \tag{2.10}$$

Note that the same posterior as in Equation (2.7) is obtained when we plug in the class proportion priors, estimated as the class proportions in the training set:

$$p(\mathbf{y_c}) = \frac{N_c}{N}. \tag{2.11}$$

Now, one may ask what is the interest of estimating likelihoods when we can directly estimate the posterior, which is the final result we want. Let us imagine that we know the class proportions to be different in the test set and in the training set, maybe because we have created a class-balanced training set. By applying Equation (2.7) we are implicitly assuming the same class proportions in the train and test datasets. If we instead use Equation (2.8), we are then free to use a new class prior by applying Bayes' theorem (2.10).

In the simple case of K-NN classification the class prior can just convey information about known class proportions. But if we consider a richer output space, with not just a single class label but a collection of labels organized in a graph structure, much more informative class priors can be used. This is explored in the following section and exploited in Chapters 3 and 6.

2.4.2 Image structure and Graphical models

When the input $\mathbf{x} \in \mathbb{R}^{M \times N \times d}$ is an image, we consider it to be formed by multiple elements with predefined neighborhood relationships: the pixels. Each pixel is represented by its site index $i \in \mathcal{S} = [1, \ldots, MN]$, which is associated to its 2D location on the image, and its pixel value $\mathbf{x}_i \in \mathbb{R}^d$. The output $\mathbf{y}$ is accordingly formed by multiple elements with the same neighborhood structure, such as a label y_i for each image pixel x_i. That often means that we can apply some additional prior information on this structure of the output. We might know that certain classes are more likely to co-occur in the same image than others, that objects of some class tend to appear together, or that certain shapes are more likely. These priors can help obtaining better classification performance when the result of a classifier applied to each pixel contains some level of noise, which in practice is always the case, since it can help correct classification mistakes when they are unlikely according to the priors.

The name *graphical models* stems from being based on data structures called *graphs*. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$, also called sites, and a set of edges $\mathcal{E}$ that encode the neighborhood relationship between nodes. Two nodes $i, j \in \mathcal{V}$ are said to be connected if the corresponding edge ϵ_{ij} exists in $\mathcal{E}$. The neighborhood $\mathcal{N}_i$ of node i is the set of nodes connected to i.

A prior on the output $p(\mathbf{y})$ now provides information about the probability of observing some particular configuration of $\mathbf{y}$. This is almost always intractable, since the number of configurations tends to grow exponentially with the number of elements. The complexity can be reduced to linear with respect to the number of elements by making use of some locality constraint, which factorizes the probability of the whole output configuration $p(\mathbf{y})$ to the product of probabilities of parts of the output. The intuition is that only nearby elements have a direct influence on each other, and thus using the probabilities of many

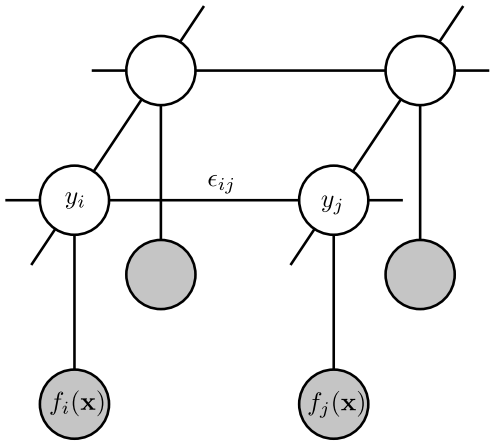


Figure 2.4: Graph of a Markov Random Field for image segmentation. The gray nodes represent the data dependent variables, while the white nodes are the output variables, *i.e.* the class labels. The outputs of pixels i and j are connected by the edge ϵ_{ij}.

local *cliques*, sets of elements fully connected to each other, can be equivalent to using the full $p(\mathbf{y})$. If a clique is denoted by $C \in \mathcal{C}$, then:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \Psi_C(C), \tag{2.12}$$

where $\Psi_C(C)$ is the clique potential. In the case of order 2 cliques, $C = (i, j)$, $\Psi_2(i, j) \geq 0$ is called *pairwise potential* and measures the compatibility of y_i and y_j and, optionally, $\mathbf{x}_i$ and $\mathbf{x}_j$. For order 1 cliques $C = (i)$ and $\Psi_1(i) = p(\mathbf{x}_i|y_i)$. Called *unary potential*, it is usually the likelihood predicted by some classifier. Z is the partition function that makes sure that the result is normalized as a probability distribution. If we separate the unary and pairwise factors in Equation (2.12), we obtain:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i \in \mathcal{C}_1} \Psi_1(\mathbf{x}_i, y_i) \prod_{(i,j) \in \mathcal{C}_2} \Psi_2(y_i, y_j, \mathbf{x}_i, \mathbf{x}_j), \tag{2.13}$$

a formulation called Conditional Random Field (CRF), since the output is generally treated as a field of random variables and all the factors are conditioned on the input. A simple and commonly used prior is contrast-sensitive label smoothness, meaning that we believe that similar and neighboring pixels are more likely to belong to the same class than to different classes. The pairwise potential can be $\Psi_2(i, j) = 1$ if $y_i = y_j$ and otherwise $\Psi_2(i, j) = \alpha(\|\mathbf{x}_i - \mathbf{x}_j\|)$, with $0 \leq \alpha < 1$. Figure 2.4 shows a diagram of the connectivity of a random field with pairwise connections in a 4-neighborhood and unary potential from a classifier $f(\mathbf{x})$. Note that the factors in this case coincide with the edges because only pairwise potentials are considered, but this does not always have to be the case.

If the pairwise potential is not conditioned on the data, for instance if α depends only on the labeling, then the posterior can easily be factorized such that it corresponds to Bayes' theorem, Equation (2.10):

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i \in \mathcal{C}_1} \Psi_1(\mathbf{x}_i, y_i) \prod_{(i,j) \in \mathcal{C}_2} \Psi_2(y_i, y_j) = \frac{1}{p(\mathbf{x})} p(\mathbf{x}|\mathbf{y}) p(\mathbf{y}) \tag{2.14}$$

The per pixel likelihood provided by a classifier can be combined with this prior information using Bayes' theorem (2.10) in the same way that the knowledge of class proportions can be injected in the K-NN output. This special case of CRF is called a Markov Random Field (MRF), and was first introduced to image processing for the task of image restoration (Geman and Geman, 1984) but has since been used to pose many vision tasks as graph labeling problems (Li, 1994).

The aim is to find the $\mathbf{y}$ that maximizes the expression in Equation 2.13. Since Z does not depend on $\mathbf{y}$ it can conveniently be removed from the expression. In order to make this optimization problem easier, we can instead minimize the logarithm of the expression, referred to as energy function:

$$E(\mathbf{y}|\mathbf{x}) = -\sum_{i \in \mathcal{C}_1} log(\Psi_1(\mathbf{x}_i, y_i)) - \sum_{(i,j) \in \mathcal{C}_2} log(\Psi_2(y_i, y_j, \mathbf{x}_i, \mathbf{x}_j)). \tag{2.15}$$

A variety of optimization methods have been proposed to solve the minimization of Equation (2.13) with respect to $\mathbf{y}$, in general allowing for a trade-off between flexibility/simplicity and optimality/efficiency. On the latter end of the spectrum the most popular methods are based on graph-cuts (Boykov et al., 2001; Kolmogorov and Zabih, 2004), which are fast and reliable but are more strict in terms of the required properties of the pairwise potential. On the former, Iterated Conditional Modes (ICM) is a very simple algorithm that tends to be more sensitive to the initialization, typically returning results of lower quality than graph-cut based methods, but poses no additional requirement in the properties of the energy function. This thesis makes use of this formulation, optimized with ICM, in Chapters 3 and 6.

2.4.3 Active contours and shape priors

Models of the CRF family, as exposed in Section 2.4.2, are well adapted to encode positional priors, related to relative positions of output elements. Due to their local nature, required to allow for efficient inference through factoring, CRFs do not naturally encode high-level geometric constraints such as closedness, convexity or shape smoothness. Active Contour Models (ACMs) were introduced by Kass et al. (1988) for single object segmentation in images. ACMs naturally constrain the segmentation output to be closed, compact and smooth. To do this, the output is forced to be a polygon with a fixed number of edges L and encouraged to be smooth while sitting on the object boundaries found in the image. An ACM can be represented as a polygon $\mathbf{y}$ with L nodes, made up of 2D Cartesian coordinates $\mathbf{y}_s = (u_s, v_s) \in \mathbb{R}^2$, with $s \in 1 \dots L$, where each s represents one of the nodes that form the discretized contour. The polygon $\mathbf{y}$ is deformed such that the following energy function is minimized (see also Figure 2.5):

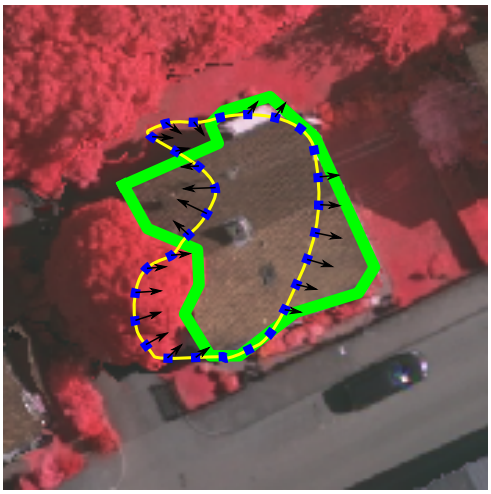


Figure 2.5: Explicit ACMs work by initializing a polygon $\mathbf{y}$ on the image, represented by the yellow line with blue nodes, where the nodes are explicitly describing the polygon. An energy function that depends on the image and the curvature and length of $\mathbf{y}$ is manually crafted such that the "forces" (black arrows) created by the minimization of the energy push the polygon towards the desired outline, in green.

$$E(\mathbf{y}, \mathbf{x}) = \sum_{s=1}^{L} \left[D\big(\mathbf{x}, (\mathbf{y}_s)\big) + \alpha\big(\mathbf{y}_s\big) \left| \frac{\partial \mathbf{y}}{\partial s} \right|^2 + \beta\big(\mathbf{y}_s\big) \left| \frac{\partial^2 \mathbf{y}}{\partial s^2} \right|^2 \right], \tag{2.16}$$

where $D\big(\mathbf{x}\big) \in \mathbb{R}^{M \times N \times 1}$ is the data term, with input image $\mathbf{x} \in \mathbb{R}^{M \times N \times d}$, evaluated at position $\mathbf{y}_s$, and $\alpha, \beta \in \mathbb{R}$ are the penalization terms, encouraging short and smooth polygons respectively. The notation $D\big(\mathbf{x}, (\mathbf{y}_s)\big)$ means the value in $D\big(\mathbf{x}\big)$ indexed by the position of $\mathbf{y}_s$. In the literature, $D\big(\mathbf{x}\big)$ is usually some predefined function on the image, such an edge detector, and α and β are hand tuned constants. Many variants of this method can be found in the literature which add other terms to Equation (2.16), such as size (Cohen, 1991) and shape priors (Leventon et al., 2002; Rousson and Paragios, 2008). The latter allow to use a collection of predefined shapes to encourage the result to converge towards one of them.

Chapter 6 presents a modified ACM that allows, using a deep CNN, to learn from a training set how much each term of the energy function has to be encouraged in each location of the image.

2.4.4 Equivariance to transformations

Let us start by explaining a related and more familiar property: *invariance*. We say that a function f is invariant to a transformation g if the output of f is not modified when transforming the input under g. To formalize this, it is often convenient to describe the set of possible transformations within the framework of Group Theory. A group G is a set with some operation $(\circ)$ such that the identity element exists, every element has an inverse and associativity and closure are satisfied. Note that no reference has been made to how this transformation is implemented. To help with this, another useful concept is the *group representation*. A mapping T from G to a family of linear operators is a

representation of G if, for $g, h \in G$:

$$T(g)T(h) = T(gh). \tag{2.17}$$

A representation T is then a linear operator that implements the transformation.

Let us take as example the group of 90°-step rotations G_{90}, as well as three functions with different behaviors with respect to rotation (see Figure 2.6):

- **Invariance:** A classifier f_1. A rotation by $\alpha°$, g_α, is represented by the corresponding reshuffling of the image pixels $T^{\mathcal{X}}(g_\alpha)$. The output, on the other hand, is expected to stay constant under rotations of the input, making f_1 rotation invariant:

$$f_1(T^{\mathcal{X}}(g)\mathbf{x}) = f_1(\mathbf{x}). \tag{2.18}$$

- **Covariance:** An orientation estimation function f_2. The output should behave predictably with when the input is rotated, although not with a rotation but with a scalar addition, meaning that the representation of the transformation is different at input and output but we can find two representations of G, $T^{\mathcal{X}}$ and $T^{\mathcal{Y}}$, such that:

$$f(T^{\mathcal{X}}(g)\mathbf{x}) = T^{\mathcal{Y}}(g)f(\mathbf{x}). \tag{2.19}$$

  In this case, to apply the rotation transformation to the output of the car angle regression value $y_2 \in \mathbb{R}$, the transformation g_α is represented by the addition $T^{\mathcal{Y}_2}(g_\alpha)y = y + \alpha$. In this thesis this will be referred to as *rotation covariance*. In the literature this more general case is called equivariance.

- **Equivariance:** A semantic segmentation function f_3. The output responds with a rotation to the rotation of the input, meaning that the representations are equal, $T^{\mathcal{Y}} = T^{\mathcal{X}}$. This will be referred to as rotation equivariance in this thesis:

$$f(T^{\mathcal{X}}(g)\mathbf{x}) = T^{\mathcal{X}}(g)f(\mathbf{x}). \tag{2.20}$$

In Figure 2.6, we can see the four possible states of an image when subject to the different elements in the group: g_0, g_{90}, g_{180} and g_{270}. We can verify that this set of transformations form a group: the identity element exists and can be called g_0 or g_{360}; associativity applies under composition, since actually all the elements can be generated by applying g_{90} zero, one or multiple times (for instance, $g_{180} = g_{90} \circ g_{90}$); every element has an inverse, as shown in the figure; and the group is closed, since the composition of any two 90°-step rotations is also a 90°-step rotation.

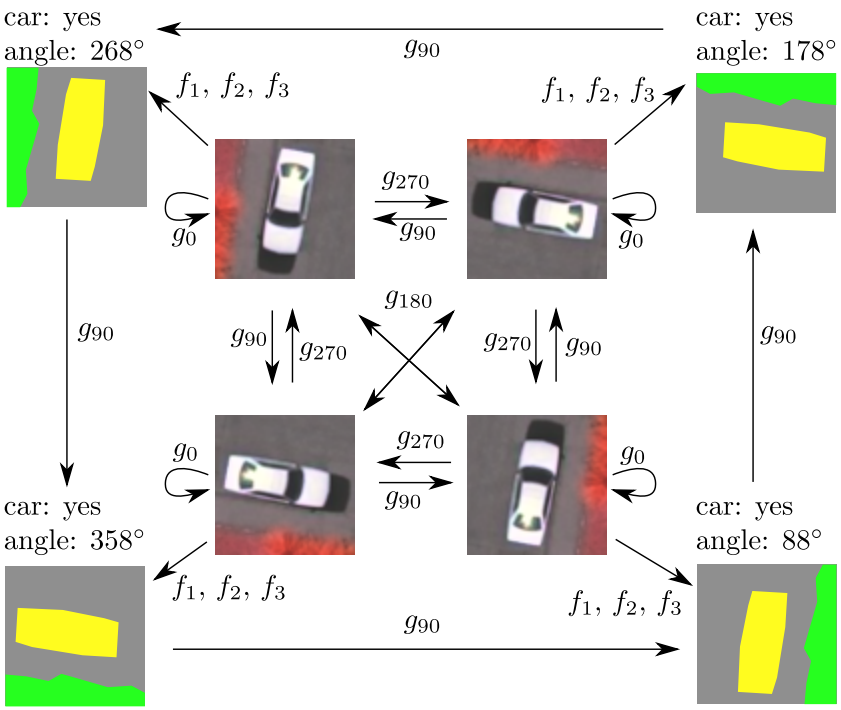


Figure 2.6: All four possible transformed images under the group of 90°-step rotations $G_{90} = \{g_0, g_{90}, g_{180}, g_{270}\}$. The rotation *invariant* function f_1 is a classifier that outputs 1 in the presence of a car in the image and 0 otherwise. The rotation *covariant* function f_2 estimates the absolute orientation of the car in the image. The semantic segmentation encoded by f_3 is *equivariant* with respect to rotations of the image.

Translation equivariance in Convolutional Neural Networks

Within DNNs, there is a family of models where hand crafted constraints play a specially important role: CNNs. These models are adapted to work on images by applying the same set of locals weights at each location of the image by means of a convolution operation, what can be thought of as looking for the same local pattern at every location. This allows to greatly reduce the model size and improve generalization (LeCun et al., 1989) by encoding a very strong prior in computer vision: *translation equivariance*. The advantages of translation equivariance provided by the use of convolutions have played a big role in the recent dominance of CNN based methods in computer vision.

Equivariance to other tranformations in CNNs

The use of the convolution operator naturally encodes translation equivariance in CNNs by forcing each convolutional layer to apply the same local feature extraction across all the translation space. Note that this is made simple by the fact that image data is already arranged in a way that corresponds to translation space. This means that applying a local operation at multiple locations of the translation space is done naturally and the output

can be arranged in the same way as the input, since every location (*i.e.* pixel) in the input image corresponds to a single location in translation space and is thus assigned a single output value. On the other hand, if we want to apply an operation at multiple "locations" of the rotation space (*e.g.* apply rotated versions of the same filter) we obtain multiple output scalars per pixel, corresponding to a new "rotation dimension". This results in a 3D tensorial output in roto-translation space, which requires special treatment when in turn used as the input of the following layer of a CNN. In the last few years there have been multiple approaches in the literature encoding equivariance or covariance to different rotation orbits, such as 90° stop rotations (Cohen and Welling, 2016a), arbitrary discreet rotations (Zhou et al., 2017) or continuous rotations (Worrall et al., 2016). A new approach for doing this efficiently is presented in Chapers 4 and 5.

Chapter 3

Geospatial Correspondences for Multimodal Registration

This chapter is based on:

Marcos, D., Hamid, R., and Tuia, D. (2016a). Geospatial correspondences for multimodal registration. In *Proceedings of the CVF/IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Abstract

The growing availability of very high resolution (<1 m/pixel) satellite and aerial images has opened up unprecedented opportunities to monitor and analyze the evolution of land-cover and land-use across the world. To do so, images of the same geographical areas acquired at different times and, potentially, with different sensors must be efficiently parsed to update maps and detect land-cover changes. However, a naïve transfer of ground truth labels from one location in the source image to the corresponding location in the target image is generally not feasible, as these images are often only loosely registered (with up to $\pm$ 50m of non-uniform errors). Furthermore, land-cover changes in an area over time must be taken into account for an accurate ground truth transfer. To tackle these challenges, we propose a mid-level sensor-invariant representation that encodes image regions in terms of the spatial distribution of their spectral neighbors. We incorporate this representation in a MRF to simultaneously account for nonlinear mis-registrations and enforce locality priors to find matches between multi-sensor images. We show how our approach can be used to assist in several multimodal land-cover update and change detection problems.

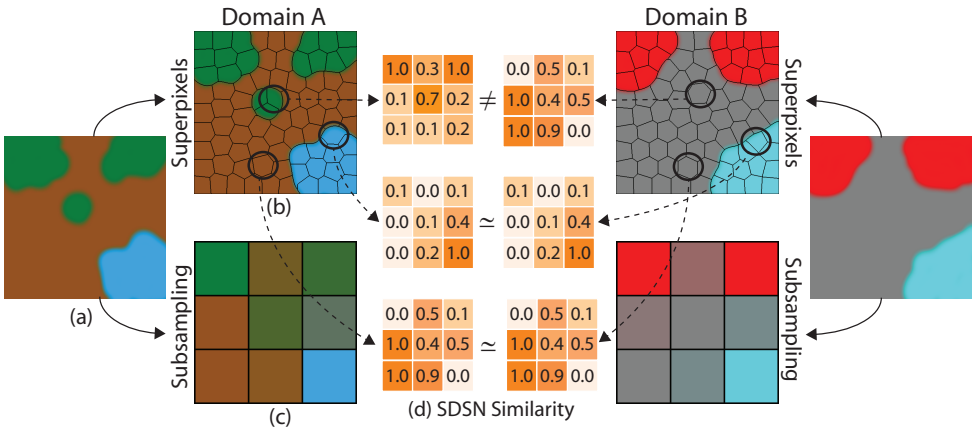


Figure 3.1: Illustration of Spatial Distribution of Spectral Neighbors (SDSN). (a): Remotely sensed image $\mathrm{I}_{\mathcal{A}}$ in domain A. (b): Superpixels computed for $\mathrm{I}_{\mathcal{A}}$. (c): Downsampled version of $\mathrm{I}_{\mathcal{A}}$. (d): The SDSN features are computed as similarities of a superpixel in (b) with every value in (c). The same procedure is applied to the image in domain $\mathcal{B}$. Superpixels belonging to the same land-cover class tend have similar SDSN values across domains.

3.1 Introduction

In recent years, there has been a tremendous increase in the amount and resolution of commercially available satellite imagery (Doe, 2014). This growth has opened numerous avenues to monitor and analyze the land-cover and land-use around the world, resulting in many novel applications including precision agriculture (Yang et al., 2013), population density estimation (Harvey, 2002), and location based services (Schiller and Voisard, 2004).

A key challenge common to these applications is the efficient generation of land-cover maps, *i.e.* segmenting remotely sensed images into semantic classes such as forests, roads, buildings *etc.* This problem is exacerbated by the need to frequently update these maps by accounting for the constant natural and man-made changes on the Earth's surface. The growing number of available air and space-borne sensors, together with their short revisit time makes the automatic updating of such maps with remote sensing data an important and challenging research direction (Lu et al., 2014).

Traditional mapping approaches cannot be directly applied to solve this problem, as the appearance-consistency assumption they make does not generally hold true for multi-sensor multi-temporal (heterogeneous) images. This is due to the large variation in acquisition conditions *e.g.* frequency bands, resolution, acquisition times and geometry. It is therefore common to view multi-sensor land-cover update from the perspective of domain adaptation (Ben-David et al., 2010) where correspondences between the *source* and *target* domains are defined using a *shared* feature-space.

In this work, we propose a novel mid-level representation that assists in performing domain

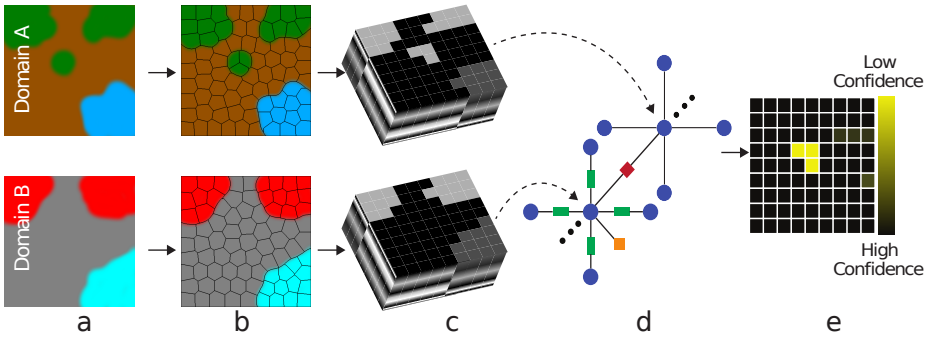


Figure 3.2: (a): Remotely sensed images. (b): Superpixel segmentation of the images. (c): Stack of domain invariant features computed for each superpixel maintaining the neighborhoods. (d): Graphical model used to match superpixels from both domains. (e): Contribution of each match to MRF cost function is used as confidence map of the matching, enabling the detection of areas with higher probability of having undergone a land cover change.

adaptation by extracting a domain-invariant feature for every image region (a super-pixel, Figure 3.1b) in each image in terms of the *spatial distribution* of its *spectral neighbors* (SDSN). Our representation is particularly geared towards image series acquired over the same geographical area at different times and using different sensors. In order to obtain domain invariance we exploit the fact that satellite images are loosely geo-registered, usually up to a non-uniform registration error of ± 50 m. We can therefore divide the images into a relatively coarse set of patches (Figure 3.1c) in order to obtain an approximately registered shared coordinate system. By encoding image regions in terms of their *spectral distances* from the mean value of each patch in their respective domains (Figure 3.1d), SDSN is able to provide a simple yet effective way to map information across different satellite sensors. This sensor invariance allows to use these features for multimodal image registration. We incorporate our SDSN representation in a MRF (Figure 3.2) where intra-domain edges are used to encourage smoothness and favor matches over short distances, while inter-domain edges encourage matching superpixels with similar domain-invariant features (Figure 3.2d). We show how this can be used for domain adaptation using two different strategies: direct transfer of land-cover ground truth (GT) (Section 3.4.1) and finding super-pixel pairs for unsupervised manifold alignment (Section 3.4.2). We also show how this approach can be used effectively for detecting land-cover changes in an unsupervised manner (Section 3.4.3).

3.2 Related Work

The problem of land-cover segmentation in multi-sensor and multi-temporal scenarios can be formulated as an instance of the more general problem of *domain adaptation*, which addresses the transfer of available domain-specific knowledge to a different but related

domain. Both semi-supervised (Daume III and Marcu, 2006; Wang and Mahadevan, 2011) and unsupervised (Gong et al., 2012; Gopalan et al., 2011) approaches have been proposed to perform domain adaptation.

For semi-supervised approaches, techniques involving co-training (Tur, 2009), label propagation (Xing et al., 2007), variants of expectation maximization (Dai et al., 2007) and SVM (Duan et al., 2009) have been successfully proposed. More recently, approaches involving co-regularization (Kumar et al., 2010) and data rotation (Pan et al., 2011) have also been put forth. These approaches still require some labeled examples in the target domain, which prevents their application to problems where such labels are not available.

Unsupervised domain adaptation is generally considered a harder problem since we do not have any labeled correspondence between the domains. In this regard, approaches relying on source-target partial distribution similarity (Gong et al., 2013), clustering (Shimodaira, 2000; Japkowicz and Stephen, 2002; Bruzzone and Marconcini, 2010), structural correspondence learning (Blitzer et al., 2007), domain divergence minimization (Blitzer et al., 2008), manifold alignment (Wang and Mahadevan, 2009) and deep learning (Ganin and Lempitsky, 2014) have been proposed. However, these methods require a certain level of correlation between the distributions of both domains. In this work, we put forth a strategy that is able to deal with distributions from different domains without requiring them to be correlated by exploiting the fact that for our setting the images are loosely spatially geo-registered.

A third approach, not always explicitly referred to as domain adaptation, consists of using engineered domain invariant features. It has been successfully applied in both remotely sensed (Vakalopoulou et al., 2015; Gueguen and Hamid, 2015) and natural images (Liu et al., 2011). Some of these features (*e.g.* SIFT (Liu et al., 2011) or shape descriptors (Gueguen and Hamid, 2015)) achieve domain-invariance at the cost of discarding relevant information, *e.g.* color in RGB imagery, while focusing only on geometrical information. In contrast, we propose to take into account spectral information while also offering the possibility to include task-specific appearance descriptors in the process. SDSN is related to the local self-similarity (LSS) (Shechtman and Irani, 2007) and global self-similarity (GSS) (Deselaers and Ferrari, 2010) features. However, unlike these approaches, SDSN uses approximate geographical correspondences to allow for cross-domain comparisons, hence offering both the expressiveness of GSS and the simplicity of LSS.

In remote sensing, domain adaptation has been traditionally used for land-cover map update tasks (Bruzzone and Serpico, 1997; Benedek and Szirányi, 2009). Most of these pipelines assume a perfect pixel-to-pixel registration between multi-temporal images, which is a serious limiting factor for high resolution and multi-sensor data. An object-based variant resides in the semantic tie points strategy proposed in (Montoya-Zegarra et al., 2013) and used in (Marcos-Gonzalez et al., 2015) for remote sensing domain adapta-

tion. An MRF-based approach (Liu et al., 2011) significantly relaxing the co-registration constraint is presented in (Vakalopoulou et al., 2015), where registration and change detection are simultaneously performed. They use several correlation similarity measures that imply using the same number of spectral bands in both domains. Our approach extends this work to the multi-sensor setting where feature spaces are usually composed by different types and number of spectral channels, making it more general.

3.3 Proposed Method

We use super-pixels as our basic computational unit since they reduce the size of the problem while offering a meaningful spatial support. In this work we use the Simple Linear Iterative Clustering (SLIC) segmentation method presented in (Achanta et al., 2012). Given an image $I_{\mathcal{D}}$ of size $m \times n$ in domain $\mathcal{D}$, and a SLIC segment size parameter s, we build a super-pixel image $H_{\mathcal{D}}$ of size roughly $(m/s \times n/s)$.

We formulate our problem as an object matching problem by using a MRF, similar to (Vakalopoulou et al., 2015; Liu et al., 2011). An important feature of this model is that the contribution to the MRF energy associated to each matched pair can be used as an estimate of matching confidence. Every super-pixel $H^j_{\mathcal{B}} \in H_{\mathcal{B}}$ in domain $\mathcal{B}$ (target) is matched to super-pixel $H^i_{\mathcal{A}} \in H_{\mathcal{A}}$ in the domain $\mathcal{A}$ (source) with a certain confidence relative to rest of the matches.

3.3.1 Spatial Distribution of Spectral Neighbors

Our main hypothesis is that objects that are spectrally similar in one domain tend to be spectrally similar in other domains, except when they have undergone a land cover change. For instance, a patch of vegetation in an RGB image is likely to have a similar color to other areas of vegetation in that image. At the same time, a patch of vegetation in a near infrared (NIR) image is likely to look very similar to other vegetated areas in the same NIR image. This within-image similarity is independent to how similar or dissimilar a particular patch of vegetation might look across the two images. We use this observation to encode each super-pixel $H^i_{\mathcal{D}}$ in domain $\mathcal{D}$ in terms of its similarity to other regions of the image (see Figure 3.1).

To do so, we start by computing a downscaled version $J_{\mathcal{D}}$ of the original image $I_{\mathcal{D}}$ as the average spectral signature of every non-overlaping $d \times d$ patch in $I_{\mathcal{D}}$. Here $J_{\mathcal{D}}$ is of size $(m/d \times n/d)$ and contains $Q = (mn)/d^2$ elements. We then compute the SDSN feature $\mathbf{z}^i_{\text{SDSN}}$ for $H^i_{\mathcal{D}}$ as:

$$\mathbf{z}^i_{\text{SDSN}} = [z^{i1}_{\text{SDSN}} \cdots z^{iq}_{\text{SDSN}} \cdots z^{iQ}_{\text{SDSN}}], \tag{3.1}$$

where

$$z^{iq}_{\text{SDSN}} = e^{-\sigma \|S(J^q_{\mathcal{D}}) - S(H^i_{\mathcal{D}})\|^2}. \tag{3.2}$$

Here, $S(J^q_{\mathcal{D}})$ and $S(H^i_{\mathcal{D}})$ are the spectrum (*e.g.* RGB color) associated to $J^q_{\mathcal{D}}$ and the mean spectrum of $H^i_{\mathcal{D}}$ respectively. Note that $\mathbf{z}^i_{\text{SDSN}}$ has Q dimensions. Each element z^{iq}_{SDSN} of $\mathbf{z}^i_{\text{SDSN}}$ encodes the similarity of the spectrum of $H^i_{\mathcal{D}}$ to the average spectrum of a particular patch of the image, $J^q_{\mathcal{D}}$. Downscaling allows for robustness against registration noise between the images in the different domains. For example, a 15 pixel shift in the original image becomes a sub-pixel shift of 0.15 pixels using downscaling factor of 100.

3.3.2 Matching Formulation

We build on previous matching approaches relying on MRF such as (Shekhovtsov et al., 2008; Liu et al., 2011; Dragomir Anguelov et al., 2005; Vakalopoulou et al., 2015). We define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where every edge $\epsilon_{ij} \in \mathcal{E}$ connects two nodes $i, j \in \mathcal{V}$ with a weight $c(\epsilon_{ij})$. Every node i corresponds to a super-pixel $H^i_{\mathcal{B}}$ in the target domain. Since we expect the misregistration shifts to be consistent within a region of the image, possibly with shift larger than the superpixel size, we consider mid-range connections for node i, $\mathcal{N}_i$, beyond first order neighborhoods. In our experiments (Section 3.4) we use a 25×25 super-pixel neighborhood. We make use of the SLIC grid initialization to define the neighborhood systems efficiently. We set weights $c(\epsilon_{ij})$ inversely proportional to the geographical distance between node i and its neighbors and we normalize them such that $\sum_{\epsilon_{ij} \in \mathcal{N}_i} c(\epsilon_{ij}) = 1$. Each node is defined by its geographical coordinates $\mathbf{p}_i = (p_i, q_i)$ and is assigned to a matching vector $\mathbf{u}_i = (u_i, v_i)$ towards a super-pixel $M_i = H^k_{\mathcal{A}}$ in the source domain, defined by its coordinates $\mathbf{p}_k = \mathbf{p}_i + \mathbf{u}_i$. M is a look-up table storing the currently selected matches. The matching process is formulated as an energy minimization over the graph $\mathcal{G}$ as:

$$E(M) = \sum_i \Theta_{\text{data}}(H^i_{\mathcal{B}}, H^k_{\mathcal{A}}) + \lambda_{\text{small}} \sum_i \Phi_{\text{small}}(\mathbf{u}_i) + \lambda_{\text{smooth}} \sum_i \Phi_{\text{smooth}}(\mathcal{N}_i). \tag{3.3}$$

The data term Θ_{data} measures the dissimilarity between $H^i_{\mathcal{B}}$ and its match $H^k_{\mathcal{A}}$, defined by:

$$\Theta_{\text{data}} = \sum_{\mathbf{z} \in Z} \alpha_{\mathbf{z}} \Theta_{\mathbf{z}} \tag{3.4}$$

where $\alpha_{\mathbf{z}}$ is the weight given to each dissimilarity measure $\Theta_{\mathbf{z}}$, computed using the feature $\mathbf{z}$, *e.g.* SDSN, SIFT, color, *etc.* Here Z defines the set of all features considered.

The dissimilarity between a pair of superpixels H^k_A and H^j_B in feature $\mathbf{z} \in Z$ is computed as:

$$\Theta_{\mathbf{z}}(H^k_{\mathcal{A}}, H^i_{\mathcal{B}}) = -log\big(\mathbf{z}(H^k_{\mathcal{A}})^\top \cdot \mathbf{z}(H^i_{\mathcal{B}})\big) \tag{3.5}$$

We normalize each feature to have unit ℓ_2-norm. To further spread the samples over the unit ball, we center every vector to zero mean. Note that the matrix version of this formulation can use optimized BLAS Level-3 (Golub and Van Loan, 2012) and therefore can be computed efficiently by optimally using all the resources of modern computing architecture.

In Equation (3.3), the term Φ_{small} penalizes big matching displacements and depends only on the matching vector:

$$\Phi_{\text{small}}(i) = \|\mathbf{u}_i\|_2 \tag{3.6}$$

Similarly, Φ_{smooth} penalizes matching vectors deviating too much from the average matching vector in a neighborhood:

$$\Phi_{\text{smooth}}(\mathcal{N}_i) = \|\mathbf{u}_i - \sum_{j \in \mathcal{N}_i} c(\epsilon_{ij})\mathbf{u}_j\|_2 \tag{3.7}$$

where all $j \in \mathcal{N}_i$ are the neighbors of i and each ϵ_{ij} the corresponding edge, with $c(\epsilon_{ij})$ being the edge weight.

The confidence of the match of node i in the target domain $\mathcal{B}$ is then defined as $-E(M_i)$.

3.3.3 Optimization

Since satellite images are loosely pre-aligned, the optimal solution does not have large $\mathbf{u}$. Therefore, we limit the search for a match for $i \in \mathcal{B}$ to a window of size $w \times w$ around the initial match. In practice, we initialize the system on the geographically nearest super-pixel in $\mathcal{A}$. Note that we can see the matching problem as a classification problem with w^2 classes, corresponding to every possible match for each super-pixel in $\mathcal{B}$(Dragomir Anguelov et al., 2005). To find a set of matches M that minimize Equation (3.3), we employ the ICM algorithm (Besag, 1986). Thanks to the grid structure of the graph we can use Fast Fourier Transform (FFT) to compute the energy in the form of a convolution, which significantly improves the efficiency of the algorithm. The fact that the initialization is never very far from the solution (Szeliski et al., 2006), the use of super-pixels and the FFT means that, for image pairs used in this work, the presented method typically converges in less than 10 seconds using ICM on a standard personal computer.

3.4 Experiments and Results

We apply our proposed representation to three different problems within the context of multimodal registration. In all the experiments the SDSN feature is compared to a

multi-scale SIFT feature over the average color channel with patch sizes of 9, 17 and 33 pixels (Vedaldi and Fulkerson, 2008) and a feature consisting of the common spectral bands, thereafter referred to as "color". In all the experiments using a set of two features, the values of $\alpha_{\mathbf{z}}$ have been set to 0.5.

3.4.1 Ground Truth Transfer

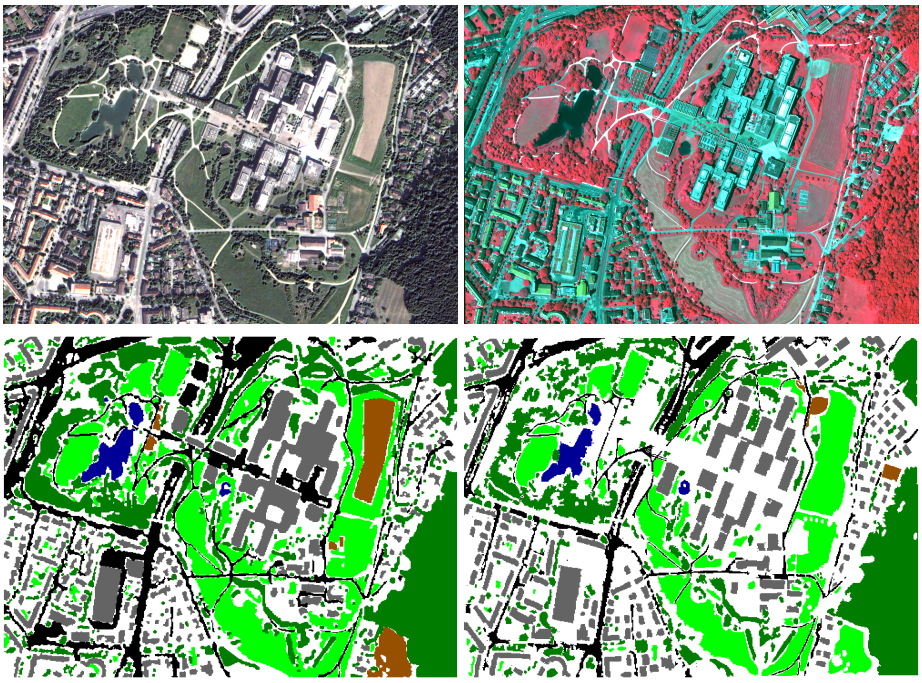
We aim to transfer the available ground truth (GT) from the source image to the target image, while simultaneously avoiding the regions that have likely undergone some land cover change. This transferred GT is then used to train a K-NN classifier in the target domain to generate an updated land cover map. The choice of K-NN classifier is due to its simplicity and distribution independence. We use a hand labeled GT of the target domain to validate the map obtained.

Dataset and Setup

The source domain consists of five QuickBird (DigitalGlobe, 2000a) satellite images of Zurich Switzerland taken in August 2002. They have four channels: near infrared, red, green and blue (NIR-R-G-B), and a resolution of about 0.62 cm/pixel. These image are a subset of the Zurich Summer dataset presented in (Volpi and Ferrari, 2015a). The target domain is a corresponding set of five NIR-R-G aerial images of the same area, with nearly the same footprint, captured during the campaign of summer 2013 and provided by the Swiss Federal Office of Topography (Swisstopo, 2018). We refer to this dataset as NIR-R-G Orthophoto data. The resolution of the target images is 25 cm/pixel. To test our approach in the case where source and target only share the R and G bands, we discard the NIR band of the QuickBird images and use exclusively the R-G-B bands throughout the experiments. Figure 3.3 shows an example image pair and the corresponding GT maps. In this dataset, the geo-registration error of the image pairs ranges from 5 m to 15 m. Each image in the source domain has a quite dense GT land-cover map consisting of between four and six classes among Roads, Buildings, Trees, Grass, Bare Soil, Water, Railways and Swimming Pools.

We treat each of the five image pairs as an independent GT transfer problem. For each image pair, we first re-scale them to the size of the smaller image in the pair. Note that this step is not required but it results in obtaining a similar number of superpixels, which helps getting a good matching. The images are then segmented with SLIC (Achanta et al., 2012), with the superpixel size of 10 pixels and the regularization parameter values set to 10. The SDSN features are computed using a downsampling factor d of 20 and a σ of 0.5. For the MRF matching we used $\lambda_{\text{smooth}} = 0.05$ and $\lambda_{\text{small}} = 0.05$.

After matching, 90% most confident matches are used for transferring the GT to the target



(a) (b)

Figure 3.3: One of the 5 image pairs from the Zurich dataset. (a): Quickbird (DigitalGlobe, 2000a) image, the source domain, and the corresponding GT. (b): False color Representation of the 3 band NIR-R-G Orthophoto data (Swisstopo, 2018) (2013) and its GT: Roads, buildings, trees, grass, water and bare soil. Best viewed in color.

image. We then used all the transferred GT to train a pixel-wise K-NN classifier with $k = 5$. The classified land cover map is compared to the hand labeled target GT for validation. We report results from QuickBird (DigitalGlobe, 2000a) to Orthophoto (Swisstopo, 2018) and vice versa.

Results

The classification results are shown in in Table 3.1. Using SIFT or color features alone produces results that are significantly worse than the results obtained using the proposed SDSN features. Using SDSN in conjunction with SIFT produces the best results on average. This is because SDSN and SIFT encode very different properties of the super-pixels, *i.e.* the former encodes spectral information in terms of global interactions across the whole image, while the latter encodes the local geometry. These two forms of information complement each other in describing land-cover changes, resulting in better GT transfer. We also compare our results with those obtained using a multi-modal mutual information-based registration method (Kroon and Slump, 2009) (Table 3.2).

Table 3.1: Average classification accuracy for ground truth transfer. The first column correspond to image pairs. $\mathcal{A}$ and $\mathcal{B}$ correspond to QuickBird (DigitalGlobe, 2000a) Orthophoto (Swisstopo, 2018) domains.

#	Transfer Dir.	Color	SIFT	Color +SIFT	SDSN	SDSN +SIFT	SDSN +Color
1	$\mathcal{A} \to \mathcal{B}$	0.652	0.551	0.579	0.694	**0.716**	0.678
	$\mathcal{B} \to \mathcal{A}$	0.281	0.438	0.371	0.548	**0.572**	0.414
2	$\mathcal{A} \to \mathcal{B}$	0.665	0.629	0.678	0.684	**0.690**	0.681
	$\mathcal{B} \to \mathcal{A}$	0.396	0.510	0.475	0.536	**0.541**	0.466
3	$\mathcal{A} \to \mathcal{B}$	0.711	0.714	0.705	0.731	**0.749**	0.733
	$\mathcal{B} \to \mathcal{A}$	0.538	0.582	0.561	0.561	**0.584**	0.563
4	$\mathcal{A} \to \mathcal{B}$	0.585	0.644	0.551	**0.730**	0.694	0.597
	$\mathcal{B} \to \mathcal{A}$	0.453	0.548	0.484	0.498	**0.557**	0.480
5	$\mathcal{A} \to \mathcal{B}$	0.559	0.766	0.756	0.782	**0.790**	0.786
	$\mathcal{B} \to \mathcal{A}$	0.599	0.690	0.705	**0.723**	0.711	0.720
Mean		0.544	0.607	0.586	0.649	**0.660**	0.612

Table 3.2: Numerical comparison with (Kroon and Slump, 2009), Average Accuracy.

Method	SDSN+SIFT	Affine (Kroon and Slump, 2009)	Non-rigid (Kroon and Slump, 2009)
AA	66.0%	63.7%	64.0%

Parameter Sensitivity and Circular Validation

We now focus on the sensitivity of our method to the choice of parameter values used when transferring the GT from source to target $\mathbf{A} \to \mathbf{B}$. In the left column of Figure 3.4 we see the result of applying this concept to the values of λ_{small} and λ_{smooth}, the SDSN's σ, and the down-scaling factor d. It can be observed that most image-pairs are not very sensitive to variations in the tested parameters showing the robustness of our framework to its various parameters.

We also study how a circular validation strategy (Bruzzone and Marconcini, 2010) can help with estimating a good set of parameters for our framework. In the case of GT transfer, this can be done by transferring the GT from source to target $\mathbf{A} \to \mathbf{B}$, and then from target back to source $\mathbf{A} \to \mathbf{B} \to \mathbf{A}$, where it is compared to the original GT for evaluation. This setting corresponds to the right column of Figure 3.4. It can be observed that the optimal values obtained during validation $\mathbf{A} \to \mathbf{B} \to \mathbf{A}$ are similar to the optimal values required for our original problem $\mathbf{A} \to \mathbf{B}$. This result shows that we can employ this circular validation strategy (Bruzzone and Marconcini, 2010) in practice to select the

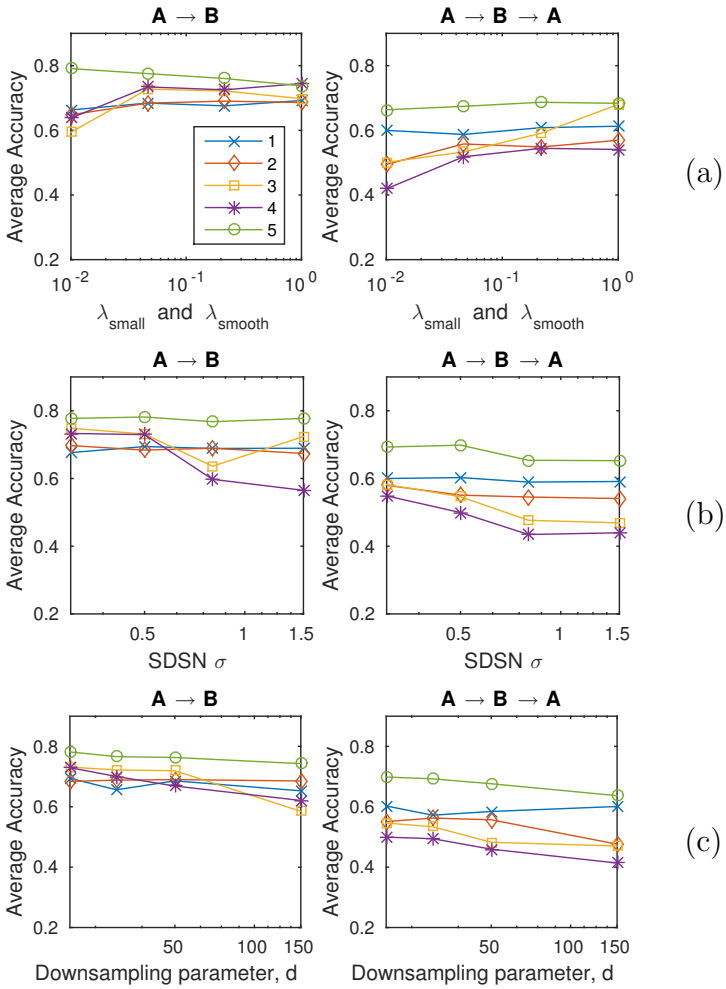


Figure 3.4: Classification accuracy in the five Zurich image pairs for different values of (a): λ_{small} and λ_{smooth} (both take the same value in this experiment), (b): σ value for the SDSN feature, (c): the downscaling parameter d for the SDSN feature. On the left, we transfer the GT from source to target. On the right is from source to target and back to source.

optimal parameter values required by our framework. Note that the downsampling factor d determines the number of operations required to compute the SDSN feature and affects the computation time in a quadratic manner (see Table 3.3). However, the results in Figure 3.4 suggest that this time can be greatly reduced at a small cost in accuracy.

Sensitivity to Perturbations in the Input

We use the image shown in Figure 3.3a in order to explore how sensitive our method is to three types of perturbations: 1) amount of change, 2) displacement and 3) rotation between the images. To do so, we match the image in Figure 3.3a with a perturbed version of itself. The land-cover changes were added by substituting vegetation superpixels

Table 3.3: Time (images 700×1000 pixels on a single CPU) to compute SDSN features wrt. downsampling, d.

d	2	5	10	20	50	100
time (s)	4.71	0.75	0.255	0.073	0.04	0.04

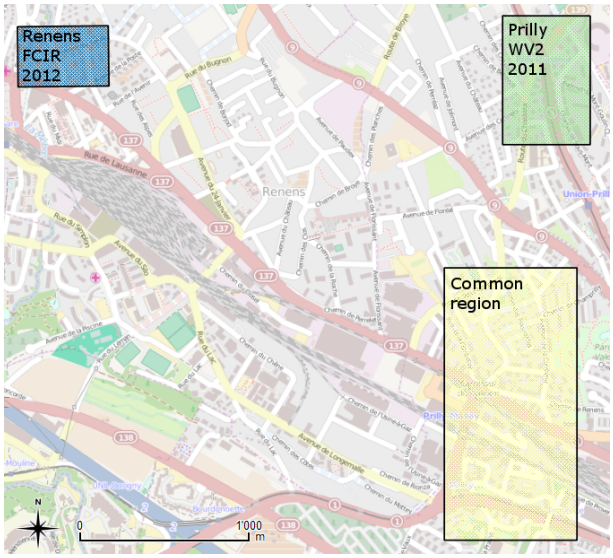


Figure 3.5: Layout of the used images in the area of Lausanne, Switzerland. Figure best viewed in color.

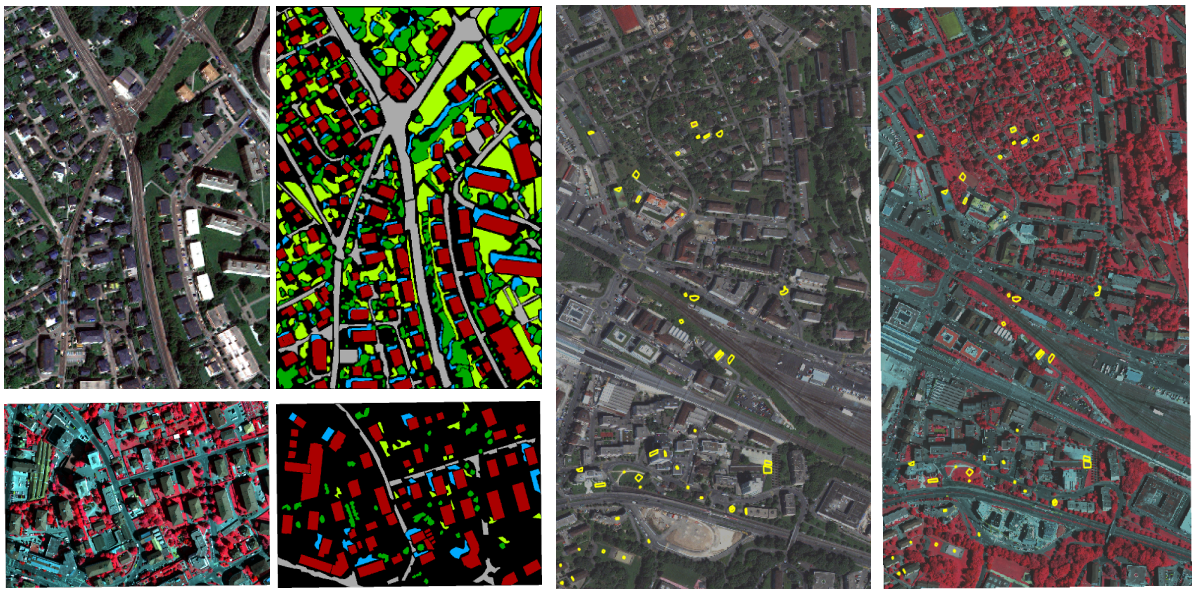
with bare soil ones. Performance was measured as the percentage of changed superpixels recalled, assuming that the amount of change had been correctly estimated. When considering displacement and rotation, the amount of change is fixed at 20%. Results in Table 3.4 show high robustness even for the most extreme amounts of change and displacement, which are the main sources of perturbation to be expected in remotely sensed images.

Table 3.4: Sensitivity to changes, displacements and rotations in the target ($\mathcal{B}$). Domains $\mathcal{A}$ and $\mathcal{B}$ consist of the RBG and NIR-R-G bands of Figure 3a respectively.

Change amount (%)	6	12	30	42
Recall (%)	96	97	96	84
Displacement (m)	10	20	30	50
Recall (%)	92	86	85	82
Angle (°)	1	3	6	10
Recall (%)	89	78	62	57

3.4.2 Unsupervised Manifold Alignment

We now explore the problem where the source and target images have a partial overlap, but there is no GT available in the overlap area. To perform domain adaptation, we



(a) (b)

Figure 3.6: (a): Representation of the 2 images used (left) and the respective GTs (right). From top to bottom: Prilly 2011 (WorldView 2 (DigitalGlobe, 2000b)) and Renens 2012 (NIR-R-G Orthophoto). Color legend: buildings, roads, grass, trees and shadows. (b2): Hand labeled super-pixel pairs (in yellow). Figure best viewed in color.

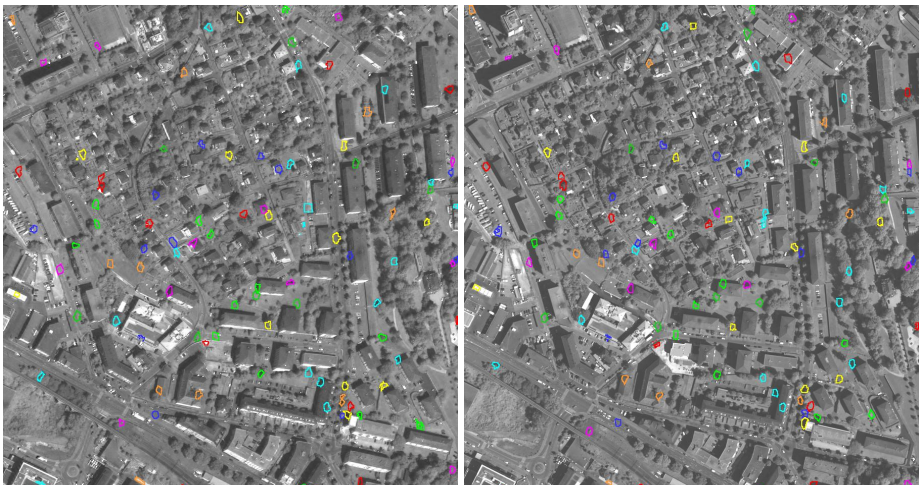
cannot simply register the labeled super-pixels directly. However, we can project both domains in a common latent space where both domains are similarly distributed. We use the same settings presented in (Marcos-Gonzalez et al., 2015), where a hand-labeled set of super-pixel pairs in the overlapping area are used to perform manifold alignment between the two domains. Instead of manual selection, we use the proposed method to automatically find a set of these super-pixel pairs.

We use the manifold alignment algorithm in (Wang and Mahadevan, 2011), where the local geometrical structure of the domain is preserved while enforcing weak class consistency using the matched super-pixel pairs. Once the domains are aligned, one can then directly train and test in the aligned domain.

To find a set of confident super-pixel matches that are also representative of the different land covers in the image, we partition the super-pixel spectra in the target domain using k-means clustering and select the most confident matches in each cluster.

Dataset and Setup

For these experiments, we use the dataset presented in (Marcos-Gonzalez et al., 2015). The images cover an area of Lausanne, Switzerland. The source domain is a World-View 2 (DigitalGlobe, 2000b) image, with 8 spectral bands in the visible and infrared



(a) (b)

Figure 3.7: Automatically generated super-pixel pair map using the proposed SDSN features. (a): Grayscale version of the World-View 2 (DigitalGlobe, 2000b) image used as source. (b): Grayscale version of the NIR-R-G Orthophoto (Swisstopo, 2018) image used as target. Matched segment pairs are within the neighborhood of each other and are marked with same color in order to imply correspondence. Best viewed in color.

region, taken in 2011 and the target domain is a 25 cm/pixel resolution NIR-R-G Orthophoto (Swisstopo, 2018). Figure 3.6a shows the areas of interest, with their corresponding GT, and Figure 3.6b the area common to both domains. The land-cover classification includes 5 classes, as shown in Figure 3.6a.

The parameters for SLIC were set to segment size of 20 pixels and regularization parameter of 10. The σ value for the SDSN was set to 0.5 and the downsampling factor for the low resolution image was set to 100. For the MRF matching, we used $\lambda_{\text{smooth}} = 10^{-2}$ and $\lambda_{\text{small}} = 10^{-2}$. We then partitioned the superpixels in the target domain into 26 clusters and randomly took 10 confident matches from each of the 20 most populated clusters.

For manifold alignment, we used the same settings as in (Marcos-Gonzalez et al., 2015). After projecting the data onto the latent feature space, we tested the performance of the alignment by classifying in the test image using only the labeled pixels from the source image. We report results using K-NN with $k = 5$ training with 400 labeled pixels per class. We also tried other classifiers, such as Random Forest and SVM (not shown in this paper), obtaining comparable trends. For each type of feature (color, SIFT, SDSN and SDSN+SIFT) we generated 5 instances of the super-pixel pair set. For each instance, as well as for the hand labeled super-pixel pairs, we computed 10 realizations of the manifold alignment and classification training set.

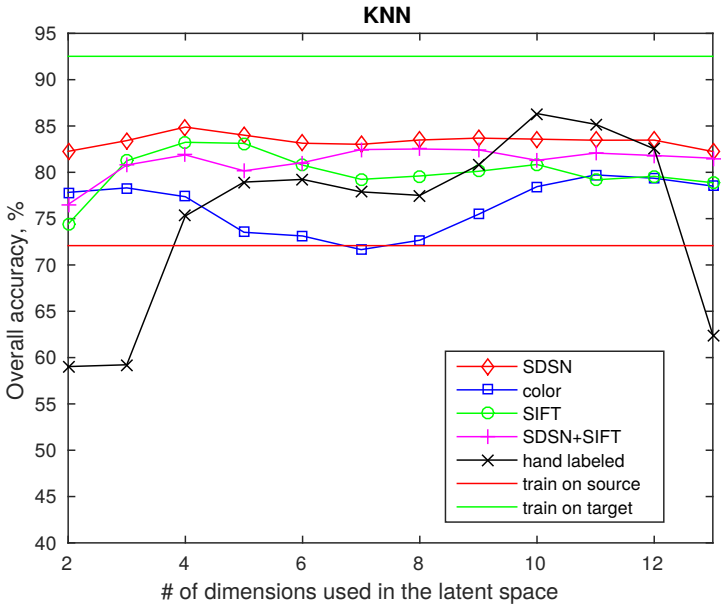


Figure 3.8: Classification results (as overall accuracy) on the manifold alignment experiment using K-NN.

Results

An example of an automatically generated super-pixel pair set with SDNS feature matching is shown in Figure 3.7. Notice how the high-confidence super-pixels pairs are distributed across the whole image evenly.

In Figure 3.8 we see the average classification results. We see how the automatically generated super-pixel pair sets using SDSN, SIFT or SDSN+SIFT perform substantially better than the hand labeled map in the vast majority of cases. Moreover, in this experiment, using SDSN+SIFT does not seem to have an advantage over SDSN alone. This is possibly due to the presence of higher rise buildings, compared to the Zurich dataset, and the acquisition angle difference between domains. This has a big impact on the local geometry, to which SIFT is highly sensitive.

3.4.3 Change Detection

The main assumption underlying the SDSN feature is that spectral neighborhood relations are domain invariant and can be used to match with high confidence the areas that remain unchanged between domains. Considering the other end of the confidence spectrum, we can build a map of low confidence areas that can be used for change detection.

Datasets and Setup

To test our framework on change detection, we use an image-pair from the dataset in Section 3.4.1, shown in Figure 3.9a and 3.9b, with change GT in green. We also apply our

method to an RGB image-pair from Google Earth of an agricultural area near Melbourne, Australia taken on 10/17/2014 and 01/03/2015, Figure 3.11a-b. We chose this location as it represents a case where the three similarity measures, using SDSN, SIFT and color, provide different notions of change. Our pre-processing for this experiment is the same as in Section 3.4.1. The MRF parameters λ_{smooth} and λ_{small} are set to 0.2. We set them higher in this case compared to the GT transfer setting because here we want to penalize non-smooth high-confidence matches more. The matching is done in both directions, *i.e.*, $\mathbf{A} \rightarrow \mathbf{B}$ and $\mathbf{B} \rightarrow \mathbf{A}$. The two resulting confidence maps are then combined by taking, for each location, the minimum value among the two maps.

Results

Figures 3.9c-f show as heat-maps the low confidence matches, in yellow. We calculated the confidence using 4 different invariant features within our MRF framework: the common bands (color), SIFT, SDSN and SDSN+SIFT. In the case of SIFT, Figure 3.9e, the algorithm detects the illumination changes in the forest (lower right corner of the image) as the dominant changes. Using the common bands R and B, Figure 3.9c, highlights mostly illumination changes on the roads and rooftops, with only changes #3, 6, 12 and 14 being clearly detected. SDSN, Figure 3.9d, shows a better correlation with the labeled changes, while detecting also changes in building's shadows. This undesired effect is reduced by using SDSN+SIFT. SDSN clearly detects changes #2, 3, 4, 5, 6, 8, 9, 10, 11, 12 and 14 while maintaining a false positive ratio comparable to or even lower than the color feature. We present the ROC curves for change detection in Figure 3.10. It can be observed that results incorporating the SDSN feature are significantly better than those obtained using either color or SIFT features. Once more, using SDSN+SIFT results in a slightly better performance than SDSN alone.

Figure 3.11c-e show the non confident areas on the Google Earth image pair using the common bands, SIFT and SDSN features respectively. We can clearly see how each map corresponds to a different notion of change. Using the common bands, in this case all R, G and B, highlights the areas in which the color change is stronger, such as the dark green vegetation turning into bright bare soil. SIFT is correlated with geometrical changes in the image, such as the cloud shadow in the center top of Figure 3.11a. On the other hand, SDSN highlights the the areas that undergo an uncommon transformation compared to the predominant set of transformations. Given that most areas have either changed from vegetation to bare-soil or *vice versa*, these uncommon transformations include vegetation that continues to be vegetation (*e.g.* the field near top left corner) and bare soil that stays bare soil (*e.g.* roads). While the former transformation is an anomaly we would like to detect, the latter is an artifact due to the spectral similarity between roads and bare-soil classes.

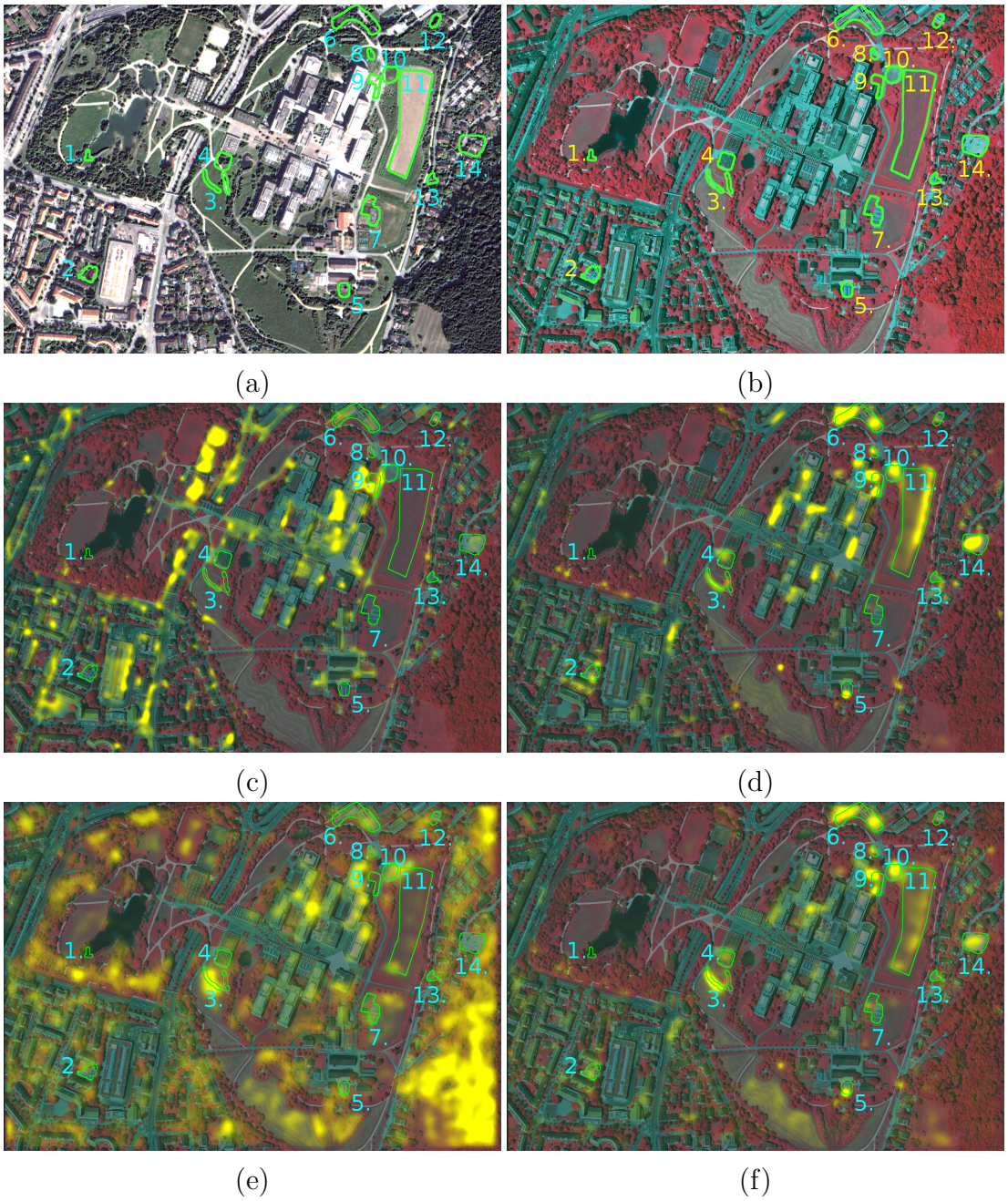


Figure 3.9: (a) Image in domain $\mathcal{A}$ and (b) $\mathcal{B}$. Change GT marked in green. (c-f) The low confidence areas, those contributing the most to the MRF energy, are shown in yellow. The matching has been performed with the same parameters using the following features: (c) common spectral bands, (e) SIFT , (d) SDSN and (f) SDSN+SIFT. Best viwed in color.

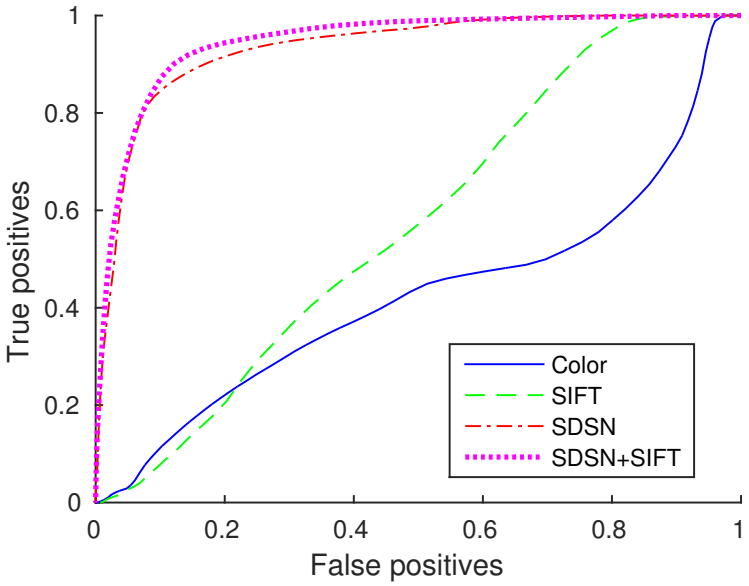


Figure 3.10: Correctly detected change pixels versus false positives for all different values of the detection threshold.

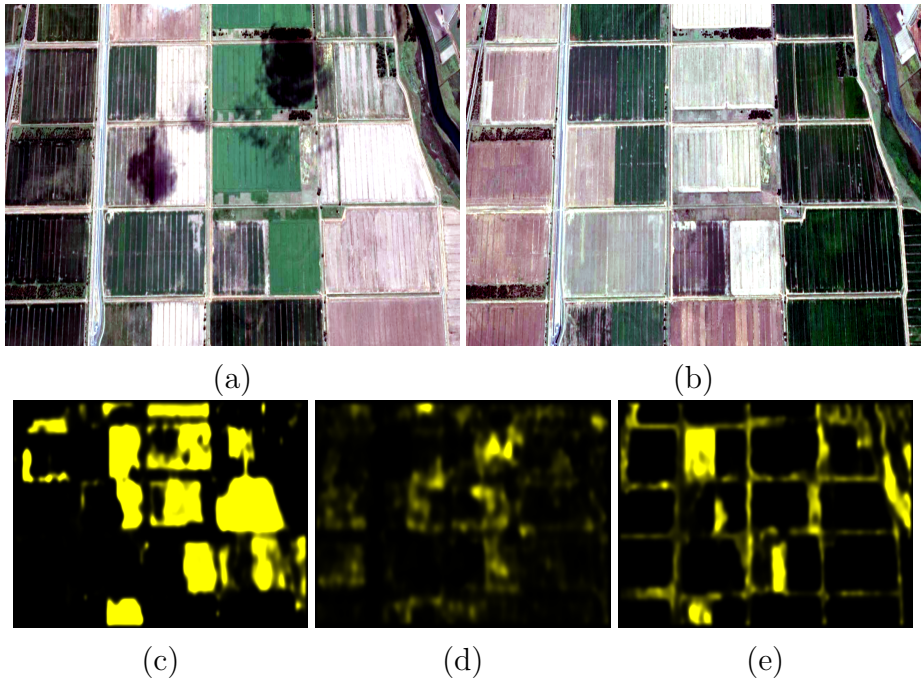


Figure 3.11: (a, b): Google Earth images taken in 10/2014 and 01/2015 respectively. (c-e): Low matching confidence areas are shown in yellow, (c) using color, (d) SIFT, (e) SDSN.

3.5 Conclusion

We proposed the spatial distributions of spectral neighbors (SDSN) as a cross-domain feature for multi-sensor, multi-temporal image pairs of sub-meter resolution. We showed that SDSN can help to match the super-pixels between two images from overlapping areas while distinguishing between the spectral changes that are the artifacts of different acquisition conditions from those due to real land-cover changes. We showed the usefulness of SDSN for land-cover map update and for change detection. We incorporate the SDSN representation into a Markov Random Field to account for nonlinear misregistrations and to enforce a locality prior in order to find matches between multi-sensor, multi-temporal images. Furthermore, we compare SDSN with other features commonly used in remote sensing image registration. Our results demonstrate that SDSN performs significantly better than the alternatives considered, maximizing domain invariance and resulting in better classification and change detection.

Chapter 4

Rotation equivariant vector field networks

This chapter is based on:

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. (2017). Rotation equivariant vector field networks. In *Proceedings of the CVF/IEEE International Conference on Computer Vision (ICCV)*

Abstract

In many computer vision tasks, we expect a particular behavior of the output with respect to rotations of the input image. If this relationship is explicitly encoded, instead of treated as any other variation, the complexity of the problem is decreased, leading to a reduction in the size of the required model.

In this paper, we propose Rotation Equivariant Vector Field Networks (RotEqNet), a CNN architecture encoding rotation equivariance, invariance and covariance. Each convolutional filter is applied at multiple orientations and returns a vector field representing magnitude and angle of the highest scoring orientation at every spatial location. We develop a modified convolution operator relying on this representation to obtain deep architectures. We test RotEqNet on several problems requiring different responses with respect to the inputs' rotation: image classification, biomedical image segmentation, orientation estimation and patch matching. In all cases, we show that RotEqNet offers extremely compact models in terms of number of parameters and provides results in line to those of networks orders of magnitude larger.

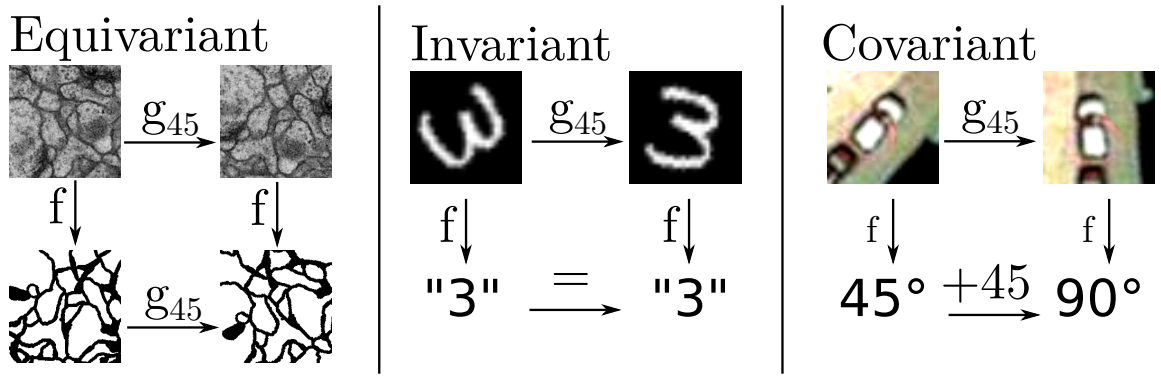


Figure 4.1: Desirable behaviors with respect to rotation of the inputs: (left) equivariance in segmentation; (center) invariance in classification; (right) covariance in absolute orientation estimation. g_{45} is an operator that rotates the input image by 45°.

4.1 Introduction

In many real life problems, such as overhead (aerial or satellite) or biomedical image analysis, there are no dominant up-down or left-right relationships. For example, when detecting cars in aerial images, the object's absolute orientation is not a discriminant feature. If the absolute orientation of the image is changed, e.g. by following a different flightpath, we would expect the car detector to score the exact same values over the same cars, just in their new position on the rotated image, independently from their new orientation along the image axes. In this case, we say that the problem is rotation *equivariant*: rotating the input is expected to result in the same rotation in the output. On the other hand, if we were confronted with a classification setting in which we are only interested in the presence or absence of cars in the whole scene, the classification score should remain the same, no matter the absolute orientation of the input scene. In this case the problem is rotation *invariant*. The more general case would be rotation *covariance*, in which the output changes as a function of the rotation of the input, with some predefined behavior. Taking again the cars example, a rotation covariant problem would be to retrieve the absolute orientation of cars with respect to longitude and latitude: in this case, a rotation of the image should produce a change of the predicted angle.

Throughout this article we will make use of the terms equivariance, invariance and covariance of a function $f(\cdot)$ with respect to a transformation $g(\cdot)$ in the following sense:

- equivariance: $f(g(\cdot)) = g(f(\cdot))$,
- invariance: $f(g(\cdot)) = f(\cdot)$,
- covariance: $f(g(\cdot)) = g'(f(\cdot))$,

where $g'(\cdot)$ is a second transformation, which is itself a function of $g(\cdot)$. With the above definitions, equivariance and invariance are special cases of covariance. We illustrate these properties in Figure 4.1. Refer to Section 2.4.4 for more details.

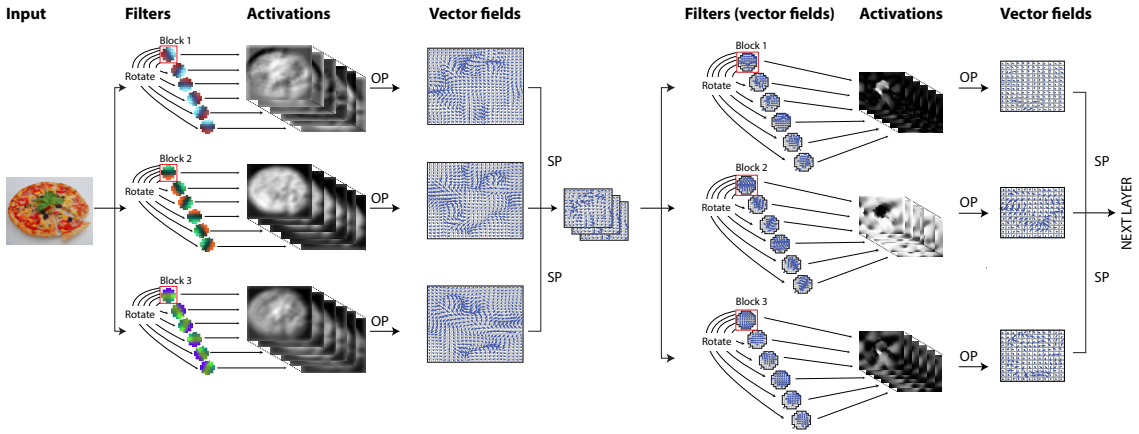


Figure 4.2: Example of the first two layers of RotEqNet. Each layer learns only three canonical filters (red squares) and replicates them across six orientations. The output of the first block are three vector field maps, which are further convolved by vector field filters in the second block (OP: orientation pooling; SP: spatial pooling).

In this paper, we propose a CNN architecture that naturally encodes these three properties: RotEqNet. In the following, we will recall how CNNs achieve translation invariance, before discussing our own proposition.

4.1.1 Dealing with translations in CNNs

The success of CNNs is partly due to the translation equivariant nature of the convolution operation. The convolution of an image $\mathbf{x} \in \mathbb{R}^{M \times N \times d}$ with a filter $\mathbf{w} \in \mathbb{R}^{m \times n \times d}$, written $\mathbf{y} = \mathbf{w} * \mathbf{x}$, is obtained by applying the same scalar product operation over all overlapping $m \times n$ windows (unit stride) on $\mathbf{x}$. If $\mathbf{x}$ undergoes an integer translation in the horizontal and vertical directions by (p, q) pixels, the same pixel neighborhoods in $\mathbf{x}$ will exist in the translated $\mathbf{x}$, but again translated by (p, q) pixels. Therefore, any operation involving fixed neighborhoods such as the convolution is translation equivariant.

A crucial consequence of learning convolution weights is a drastic reduction in the number of parameters. Without the translation equivariance assumption, each local window would have a different set of weights. Forcing weights to be shared across locations, known as *weight tying*, reduces the number of learnable parameters proportionally to the number of pixels in the image and hardcodes translation equivariance within the model. This fact is vital for the applicability of deep neural networks to images (Le Cun et al., 1990).

4.1.2 Incorporating rotation equivariance in CNNs

RotEqNet shows similar advantageous characteristics when dealing with rotations: by encoding equivariance, we are able to strongly reduce the number of parameters while

keeping similar or better accuracy across different tasks.

However, applying the exact same reasoning of weight tying for rotations is not straightforward. To follow the same logic, one should apply R rotated versions of each convolutional filter, resulting in R feature maps per filter. The dimensionality of subsequent filters would therefore increase with R, strongly increasing model size and requirements for runtime memory usage.

One way of reducing the size of the model while keeping rotation equivariance would be to propagate only the maximum value occurring across R feature maps. However, deeper layers would have no information about the orientation of features at previous layers.

We propose a trade-off between these two approaches by keeping the maximum value across the R feature maps, but in the form of a 2D vector field that captures its *magnitude* and *orientation* and propagates it through all the layers of the network.

4.2 Related work

Two families of approaches explicitly account for rotation invariance or equivariance: 1) those that transform the representation (image or feature maps) and 2) those that rotate the filters. RotEqNet belongs to the latter.

1) Rotating the inputs: Jaderberg et al. (2015) propose the Spatial Transformer layer, which learns how to crop and transform a region of the image (or a feature map) before passing it to the next layer. This transforms relevant regions into a canonical form, improving the learning process by reducing geometrical appearance variations in subsequent layers. TI-pooling (Laptev et al., 2016) inputs several rotated versions of a same image to the same CNN and then performs pooling across the different feature vectors at the first fully connected layer. Such scheme allows another subsequent fully connected layer to choose among rotated inputs to perform classification. Cheng et al. (2016b) employ in every minibatch several rotated versions of the input images. Their representations after the first fully connected layer are then encouraged to be similar, forcing the CNN to learn rotation invariance. Henriques and Vedaldi (2016) warp the images such that the translation equivariance inherent to convolutions is transformed into rotation and scale equivariance.

On the one hand, these methods have the advantage of exploiting conventional CNN implementations, since they only act on data representations. On the other hand, they can only consider global transformations of the input images. While this is well suited for tasks such as image classification, it limits their applicability to other problems (e.g. semantic segmentation), where the local relative orientation of certain objects with respect

to surroundings is what matters. Instead, RotEqNet is based on specific CNN building blocks designed to deal with local orientation information. Therefore, RotEqNet can approach diverse tasks such as classification, fully convolutional semantic segmentation, detection and regression.

It is worth mentioning that standard data augmentation strategies belong to this first family. They rely on random rotations and flips of the training samples (Simard et al., 2003): given abundant training samples and enough model capacity, a CNN might learn that different orientations should score the same by learning equivalent filters at different orientations (Lenc and Vedaldi, 2015). Unlike this, RotEqNet is well suited for problems with limited training samples that can profit from reduced model sizes, since the behavior with respect to rotations is hardcoded and it does not need to be learned.

2) Rotating the filters: Gens and Domingos (2014) tackle the problem of the exploding dimensionality (discussed in Section 4.1.2) by applying learnable pooling operations and sampling the symmetry space at each layer. This way, they avoid applying the filters exhaustively across the (high dimensional) feature maps by selectively sampling few rotations. By doing so, only the least important information is lost from layer to layer. Cohen and Welling (2016a,b) use a smaller symmetry group, composed of a flipping and four 90° rotations and perform pooling within the group. They apply it only in deeper layers, since they found that pooling in the early layers discards important information and harms the performance. Instead of explicitly defining a symmetry group, Ngiam et al. (2010) pool across several untied filters, thus letting the network learn the type of invariance. Sifre and Mallat (2013) use hand crafted wavelets that are separable in the roto-translational space, allowing for more efficient computations. Another approach to avoid the dimensionality explosion is to limit the depth of the network: Sohn and Lee (2012) and Kivinen and Williams (2011) propose such a scheme with Restricted Boltzmann Machines (RBM), while Marcos et al. (2016c) consider supervised CNNs consisting of a single convolutional layer.

These works find a compromise between the computational resources required and the amount of orientation information kept throughout the layers, by either keeping the model shallow or accounting for a limited amount of orientations. With RotEqNet, we avoid such compromise by pooling multiple orientations and passing forward both the maximum magnitude *and* the orientation at which it occurred. This modification allows to build deep rotation equivariant architectures, in which deeper layers are aware of the dominant orientations. At the same time, the dimensionality of feature maps and filters is kept low by discarding information about non-maximum orientations, thus reducing memory requirements.

The most similar approaches to RotEqNet are the recently proposed Harmonic Networks (H-Nets) (Worrall et al., 2016) and Oriented Response Networks (ORN) (Zhou et al., 2017), both of which use an enriched feature map explicitly capturing the underlying

orientations. They do so by using either complex circular harmonics (H-Nets) or the full vector of oriented responses (ORN). H-Nets offer a very compact feature map, but are limited to learning filters that are a combination of circular harmonic wavelets. On the other hand, ORN allows to learn arbitrary filters, but relies on a much less compact representation of the feature maps, leading to heavier models both in terms of size and memory requirements. RotEqNet provides the best of both worlds: the compactness of the former with the flexibility of the latter. These properties make it particularly suitable to address problems characterized by limited training samples, as we will see in the experiments.

4.3 Rotation equivariant vector field networks

We focus on achieving rotation equivariance by performing convolutions with several rotated instances of the same *canonical* filter (see Figure 4.2). The canonical filter $\mathbf{w}$ is rotated at R different evenly spaced orientations.In the experiments (Section 4.4) we deal with problems requiring either full invariance, equivariance or covariance, so we use the interval $\boldsymbol{\alpha} = [0°, 360°]$. However, this interval can be adapted to a known range of tilts. The output of the filter $\mathbf{w}$ at a specific location consists of the magnitude of the maximal activation across the orientations and the corresponding angle. If we convert this polar representation into Cartesian coordinates, each filter $\mathbf{w}$ produces a vector field feature map $\mathbf{z} \in \mathbb{R}^{M \times N \times 2}$, where the output of each location consists of two values $[u, v] \in \mathbb{R}^2$ implicitly encoding the maximal activation in both magnitude and direction. Since the feature maps have become vector fields, from this moment on the filters must also be vector fields, as seen in the right part of Figure 4.2.

The advantage of representing $\mathbf{z}$ in Cartesian coordinates is that the horizontal and vertical components $[u, v]$ are orthogonal, and thus a convolution of the two vector fields can be computed on each component independently using standard convolutions (see Eq. (4.5)).

4.3.1 RotEqNet building blocks

RotEqNet requires specific building blocks to handle vectors fields as inputs and/or outputs (Figure 4.2). In the following, we present our reformulation of traditional CNN blocks to account for both vector field activations and filters. The implementation[1] is based on the MatConvNet (Vedaldi and Lenc, 2015) toolbox[2].

[1] Will be made available at http://github.com/di-marcos/RotEqNet
[2] http://www.vlfeat.org/matconvnet

Rotating convolution (RotConv)

Given an input image with $m/2$ zero-padding $\mathbf{x} \in \mathbb{R}^{M+m/2\times N+m/2\times d}$, we apply the filter $\mathbf{w} \in \mathbb{R}^{m\times m\times d}$ at R orientations, corresponding to the angles:

$$\alpha_r = \frac{360}{R} r \quad \forall r = 1, 2 \dots R. \tag{4.1}$$

Each one of these rotated versions of the canonical filters (highlighted by red squares in Figure 4.2) is computed by resampling $\mathbf{w}$ with bilinear interpolation after rotation of α_r degrees around the filter's center.

$$\mathbf{w}^r = T(g_{\alpha_r})(\mathbf{w}), \tag{4.2}$$

where g_α is the α degrees rotation operator. Interpolation is always required unless only rotations of multiples of 90° are considered. In practice, this means that the rotation equivariance will only be approximate.

Since the rotation can force weights near the corners of the filter to be relocated outside of its spatial support, only the weights within a circle of diameter m pixels are used to compute the convolutions. The output tensor $\mathbf{y} \in \mathbb{R}^{M\times N\times R}$ consists of R feature maps computed as:

$$\mathbf{y}^{(r)} = (\mathbf{x} * \mathbf{w}^r) \quad \forall r = 1, 2 \dots R, \tag{4.3}$$

where $(*)$ is the convolution operator. The tensor $\mathbf{y}$ encodes the roto-translation output space such that rotation in the input corresponds to a translation across the feature maps. Note that only the canonical filter $\mathbf{w}$ is actually stored in the model. During backpropagation, gradients corresponding to each rotated filter $\nabla\mathbf{w}^r$ are aligned back to the canonical form and added:

$$\nabla\mathbf{w} = \sum_r T(g_{-\alpha_r})(\nabla\mathbf{w}^r). \tag{4.4}$$

This block can be applied on conventional CNN feature maps (left side of Figure 4.2) or on vector field feature maps (right side of Figure 4.2). In the second case it is computed on each component *independently* and the resulting 3D tensors added:

$$(\mathbf{z} * \mathbf{w}) = (\mathbf{z}_u * \mathbf{w}_u) + (\mathbf{z}_v * \mathbf{w}_v), \tag{4.5}$$

where subscripts u and v denote the horizontal and vertical components.

It is important to note that the image rotation operator $T(g_\alpha)$ requires an additional step when $\mathbf{w} \in \mathbb{R}^{m\times m\times 2}$ is a 2D vector field. The components of $\mathbf{w}^r = T(g_{\alpha_r})(\mathbf{w})$ have to be computed as:

$$\mathbf{w}_u^r = cos(\alpha_r) g_{\alpha_r}(\mathbf{w}_u) - sin(\alpha_r) g_{\alpha_r}(\mathbf{w}_v) \tag{4.6}$$

$$\mathbf{w}_v^r = cos(\alpha_r) g_{\alpha_r}(\mathbf{w}_v) + sin(\alpha_r) g_{\alpha_r}(\mathbf{w}_u) \tag{4.7}$$

Orientation pooling (OP):

Given the output 3D tensor $\mathbf{y}$, the role of the orientation pooling is to convert it to a 2D vector field $\mathbf{z} \in \mathbb{R}^{M \times N \times 2}$. This avoids the exploding dimensionality problem by only keeping information about the maximally activating orientation of $\mathbf{w}$. First, we extract a 2D map of the largest activation magnitudes, $\boldsymbol{\rho} \in \mathbb{R}^{M \times N}$, and their corresponding orientations, $\boldsymbol{\theta} \in \mathbb{R}^{M \times N}$. Specifically, for activations located at $[i, j]$:

$$\boldsymbol{\rho}[i, j] = \max_r \mathbf{y}[i, j, r], \tag{4.8}$$

$$\boldsymbol{\theta}[i, j] = \frac{360}{R} \arg\max_r \mathbf{y}[i, j, r]. \tag{4.9}$$

This can be treated as a polar representation of a 2D vector field as long as $\boldsymbol{\rho}[i, j] \geq 0 \quad \forall i, j$, a condition that is met when using any function on $\mathbf{y}$ that returns non-negative values prior to the OP. We employ the common ReLU operation, defined as $\texttt{ReLu}(x) = \max(x, 0)$, to $\boldsymbol{\rho}$, as it provides non-saturating, sparse nonlinear activations offering stable training. Then, this representation can be transformed into Cartesian coordinates as:

$$\mathbf{u} = \texttt{ReLu}(\boldsymbol{\rho}) \cos(\boldsymbol{\theta}) \tag{4.10}$$

$$\mathbf{v} = \texttt{ReLu}(\boldsymbol{\rho}) \sin(\boldsymbol{\theta}) \tag{4.11}$$

with $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{M \times N}$. The 2D vector field $\mathbf{z}$ is then built as:

$$\mathbf{z} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{v} \tag{4.12}$$

Spatial pooling (SP) for vector fields

Max-pooling is commonly used in CNNs to obtain some invariance to small deformations and reducing the size of the feature maps. This is done by downsampling the input feature map $\mathbf{x} \in \mathbb{R}^{M \times N \times d}$ to $\mathbf{x}_p \in \mathbb{R}^{\frac{M}{p} \times \frac{N}{p} \times d}$. This operation is performed by taking the maximum value contained in each one of the C non-overlapping $p \times p$ regions of $\mathbf{x}$, indexed by c. It is computed as $\mathbf{x}_p[c] = \max_{i \in c} \mathbf{x}[i]$, which can be expressed as:

$$\mathbf{y}_p[c] = \mathbf{y}[j], \text{ where } j = \arg\max_{i \in c} \mathbf{y}[i]. \tag{4.13}$$

This allows us to define a max-pooling for vector fields as:

$$\mathbf{z}_p[c] = \mathbf{z}[j], \text{ where } j = \arg\max_{i \in c} \boldsymbol{\rho}[i], \tag{4.14}$$

where $\boldsymbol{\rho}$ is a standard scalar map containing the magnitudes of the vectors in $\mathbf{z}$.

Batch normalization (BN) for vector fields

BN (Ioffe and Szegedy, 2015) normalizes every feature map in a mini-batch to zero mean and unit standard deviation. It improves convergence by training with stochastic gradient descent.

In our case, since working with vector fields of magnitude and orientation of activations, BN should only normalize magnitudes of the vectors to unit standard deviation. It would not make sense to normalize the angles, since their values are already bounded and changing their distribution would alter important information about relative and global orientations. Given a vector field feature map $\mathbf{z}$ and its map of magnitudes $\boldsymbol{\rho}$, we compute batch normalization as:

$$\hat{\mathbf{z}} = \frac{\mathbf{z}}{\sqrt{\text{var}(\boldsymbol{\rho})}}. \tag{4.15}$$

4.3.2 Computational considerations

Although RotEqNet allows for smaller models, they might require a higher count of convolutions than a comparable standard CNN. For instance, with the architecture used for MNIST-rot in Section 4.4, a standard CNN requires $4\times$ more filters per layer to saturate performance, compared to RotEqNet. At the same time, RotEqNet requires $R/4 = 4.25\times$ (for $R = 17$) more convolutions. This results in RotEqNet saving $10\times$ in model memory, $2\times$ in data memory at a price of requiring just $1.5\times$ more computing time. This is because, although the convolution count is higher, the number of feature maps per convolution is smaller. Less feature maps mean smaller convolution filters and the possibility to use larger mini batches, both factors contributing to a faster training.

4.4 Experiments

We explore the performance of RotEqNet on datasets where the orientation of the patterns of interest is arbitrary. This is very often the case in biomedical and overhead imaging, since the orientation of the camera is usually not correlated with the patterns of interest. We apply RotEqNet to problems from these two fields, as well to MNIST-rot, a randomly rotated handwritten digit recognition benchmark. We also perform a study on the trade-off between invariance and accuracy in a synthetic patch matching problem. These case studies allow us to analyze the performance of RotEqNet in problems requiring equivariance, covariance and invariance to rotations and to analyze the effectiveness of RotEqNet to perform accurately with very small model architectures and limited training samples.

Table 4.1: Network architecture used on the MNIST-rot dataset. Layer parameters are in white and variables are shaded in gray.

Type	Size
Input	28×28
RotConv, 2×2 SP	9×9, 6 filt. $14 \times 14 \times 6$
RotConv, 2×2 SP	$9 \times 9 \times 6$, 16 filt. $7 \times 7 \times 16$
RotConv 2×2 SP	$9 \times 9 \times 16$, 32 filt. $1 \times 1 \times 32$
Fully connected	$1 \times 1 \times 32$, 128 filt. $1 \times 1 \times 128$
FC, Softmax	$1 \times 1 \times 128$, 10 filt.
Output	$1 \times 1 \times 10$

4.4.1 Invariance: MNIST-rot

MNIST-rot (Larochelle et al., 2007) is a variant of the original MNIST digit recognition dataset, where a random rotation between 0° and 360° is applied to each 28×28 digit image. The training set is also considerably smaller than the standard MNIST, with 12k samples, from which 10k are used for training and 2k for validation. The test set consists of 50k samples. Since we aim at predicting the correct label independently from the rotation, this problem requires rotation invariance.

Model: We test four CNN models with the same architecture, but different number of filters per layer. The largest model we used is shown in Table 4.1 and involves 100k parameters. The models are trained for 90 epochs, starting with a learning rate of 0.1 and reducing it gradually to 0.001. The weight decay is kept constant at 0.01. We use a dropout rate of 0.7 in the fully connected layer and batch normalization before every convolutional layer. The number of orientations is set to $R = 17$.

Table 4.2: Error rate on the MNIST-rot dataset trained on the train-val subset.

Method	Error rate (in %)
SVM (Larochelle et al., 2007)	10.38±0.27
TIRBM (Sohn and Lee, 2012)	4.2
H-Net (Worrall et al., 2016)	1.69
ORN (Zhou et al., 2017)	1.54
TI-pooling (Laptev et al., 2016)	1.2
RotEqNet (Ours)	**1.09**
RotEqNet, only scalar field	2.01
RotEqNet, test-time augmentation	**1.01**

Test time data augmentation: We observe an important contribution of data augmentation at test time, a technique often used with approximately invariant or equivariant CNNs (Fakhry et al., 2016; Hu et al., 2015). In particular, we input to the network several rotated versions of the same image using fixed angles between 0° and 90°. Rotation-based data augmentation at test time might seem counter-intuitive in a rotation invariant model, but the different rotations coupled to resampling of images and filters (cf. Section 4.3.1) will produce slightly different activations. The final prediction is given by the average of such scores. We report results obtained with and without this type of augmentation.

Comparison to data augmented training: In order to disentangle the contributions of data augmentation and RotEqNet, we trained the RotEqNet model and a standard CNN with the same architecture and 10× more parameters. In Tab. 4.3, we show the results for these models trained on both MNIST-rot and 10k digits from the original MNIST, with and without data augmentation. We observe how both methods complement each other.

Table 4.3: Results on MNIST and MNIST-rot using a standard CNN or RotEqNet, with and without data augmentation.

	Train on MNIST		Train on MNIST-rot	
	No augm.	Augm.	No augm.	Augm.
CNN	57%	2.3%	4.9%	2.2%
RotEqNet	20%	1.1%	1.4%	1.1%

Results: We first studied the behavior of RotEqNet with respect to the total number of parameters and compared it to the state-of-the-art TI-pooling (Laptev et al., 2016). Figure 4.3 shows the results for both methods trained on the training set with different model sizes. The latter was achieved by varying the number of filters per layer, keeping the same architecture. RotEqNet requires approximately two orders of magnitude less parameters to obtain the same accuracy as TI-Pooling.

We report the test error in Table 4.2. RotEqNet obtains an error of 1.09%, a small improvement with respect to the state-of-the-art TI-pooling (Laptev et al., 2016), but with almost 100× less parameters. Test-time data augmentation further reduces the error to 1.01%, thus improving significantly over TI-Pooling and over the more recent H-Net (Worrall et al., 2016) and ORN (Zhou et al., 2017).

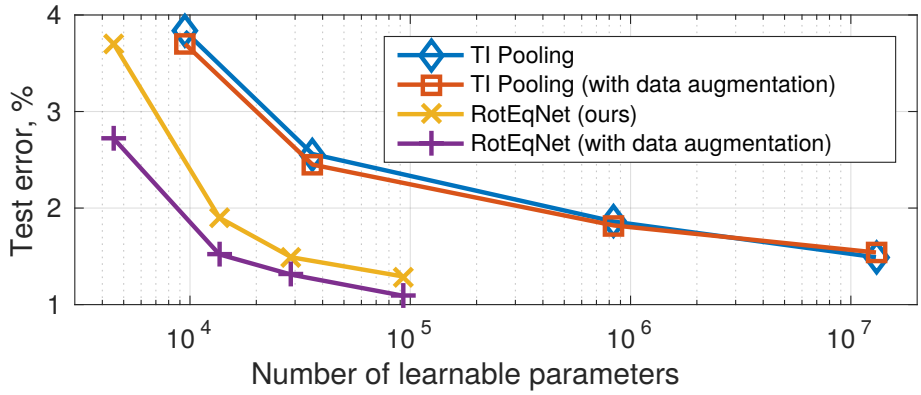
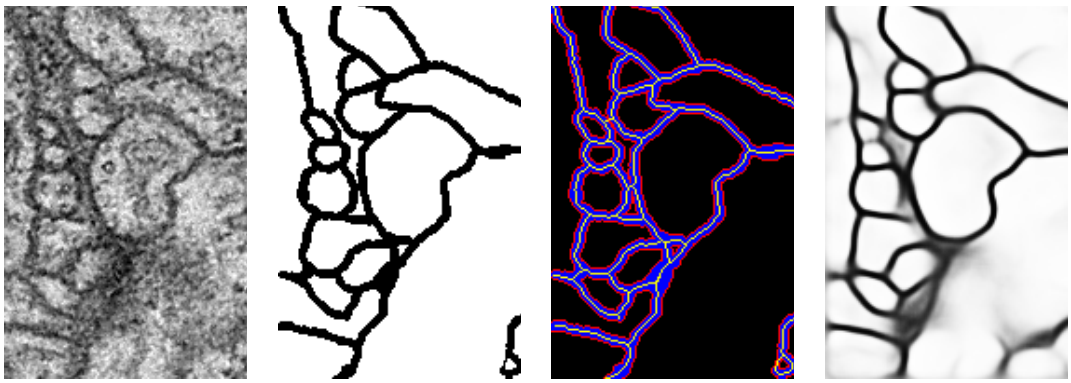


Figure 4.3: Performance of RotEqNet and TI-Pooling on MNIST-rot with respect to the number of parameters.

4.4.2 Equivariance: ISBI 2012 Challenge

This benchmark (Arganda-Carreras et al., 2015) involves segmentation of neuronal structures in electron microscope (EM) stacks (Cardona et al., 2010). In this problem we need to precisely locate the neuron membrane boundaries. Therefore, a rotation of the inputs should lead to the same rotation in the output, making the ISBI 2012 problem a good candidate to study rotation equivariance.

The data consist of two EM stacks of *drosophila* neurons, each composed of 30 images of size 512×512 pixels (Figure 4.4a). One stack is used for training and the other for testing. The ground truth for the training stack consists of densely annotated binary images (Figure 4.4b). The ground truth for the test stack is private and the results are to be submitted to an evaluation server [3].



(a) (b) (c) (d)

Figure 4.4: Example validation image (#30) of the ISBI 2012 challenge. (a) Image (190×130 pixels). (b) Membrane ground truth. (c) The pre-processed 3-class ground truth: black is non-membrane, yellow is membrane center, red is membrane border and blue is non-class. (d) Probability map produced by RotEqNet.

[3] http://brainiac2.mit.edu/isbi_challenge/

Table 4.4: Network architecture used with ISBI 2012 challenge data. Layer parameters are in white and variables are shaded in gray.

Type	**Size**
Input	512×512
RotConv,	9×9, N filt.
OP, 2×2 SP	$256 \times 256 \times N \times 2$
RotConv,	9×9, $2N$ filt.
OP, 2×2 SP	$128 \times 128 \times 2N \times 2$
RotConv	$9 \times 9 \times 2N \times 2$, $3N$ filt.
OP, 2×2 SP	$64 \times 64 \times 3N \times 2$
RotConv, OP	$9 \times 9 \times 3N \times 2$, $4N$ filt.
Upsample and stack	$512 \times 512 \times 10N$
RotConv fully	$1 \times 1 \times 10N \times 2$, $5N$ filt.
connected	$512 \times 512 \times 5N$
RotConv	$9 \times 9 \times 5N \times 2$, $4N$ filt.
OP	$512 \times 512 \times 4N$
Fully	$1 \times 1 \times 4N \times 2$, $8N$ filt.
connected	$512 \times 512 \times 8N$
FC, Normalize	$1 \times 1 \times 8N$, 3 filt.
Output	$512 \times 512 \times 3$

Model: We transform the original binary problem into a three class segmentation problem: 1) non-membrane, 2) central membrane pixels and 3) external membrane pixels. Pixels in the membrane but not belonging to either 2) or 3) are considered to be unlabeled (Figure 4.4c). This way, we can assign a higher penalization to the non-membrane pixels next to the membrane and a lower one to those in the middle of the cells. The central membrane scores are used as the final binary prediction (Figure 4.4d).

Since we are dealing with a dense prediction problem with spatial autocorrelation at different resolution levels, we apply three RotConv blocks with spatial pooling. We then upsample the output of each block to the size of the original image, before concatenating them and applying two more RotConv blocks. Table 4.4 shows the architecture. The parameter N is used to change the size of the model. We evaluated the results with $N = 2$ and with an ensemble of three models, with $N = [1, 2, 3]$.

Comparison to data augmented training: we evaluated the RotEqNet model ($N = 2$) and an equivalent standard CNN with $10\times$ more parameters on 5 held out validation images. RotEqNet seems not to profit as much from data augmentation as its standard CNN counterpart, but improves the CNN solution in all the cases considered, as illustrated in Table 4.5.

Results: A detailed explanation on the evaluation metrics used in the challenge can be found on the ISBI 2012 challenge website[3], as well as in Arganda-Carreras et al. (2015).

Table 4.5: ISBI results on the validations set using a standard CNN or RotEqNet, with and without data augmentation.

	No augm.	Augm.
CNN	0.9232	0.9572
RotEqNet	0.9726	0.9790

Table 4.6: Scores on the held out test set of the ISBI 2012 Challenge.

Method	**Rand. Thin**	**Inf. Thin**	# params.
CUMedVision (Chen et al., 2016a)	0.9768	0.9886	-
IAL MC/LMC (Beier et al., 2016)	**0.9826**	**0.9894**	-
DIVE (Fakhry et al., 2016)	0.9685	0.9858	5.7M
PolyMtl (Drozdzal et al., 2016)	0.9689	0.9861	11M
U-Net (Ronneberger et al., 2015)	**0.9728**	**0.9866**	33M
RotEqNet ($N = 2$)	0.9599	0.9806	30k
RotEqNet, 3 models	0.9712	0.9865	100k

The winners of the challenge were Chen et al. (2016a), although Beier et al. (2016) have the highest scores at the time of writing. These two works rely on complex post-processing pipeline. Our rotation equivariant prediction provides results comparable to other state-of-the-art methods only relying on the raw CNN softmax output (Drozdzal et al., 2016; Fakhry et al., 2016; Ronneberger et al., 2015) (see Table 4.6).

4.4.3 Covariance: car orientation estimation

Estimating car orientations from above-head imagery requires rotation covariant models. We use the dataset provided by the authors of Henriques and Vedaldi (2016), which is based on Google Map images. It is composed by 15 tiles, where cars' bounding boxes and corresponding orientations come from manual annotation. We implement our approach in similarly to Henriques and Vedaldi (2016). We crop a 48×48 square patch around every car, based on the bounding box center point. We then use these crops for both training and testing of the model. As in Henriques and Vedaldi (2016), we use the cars in the first 10 images (409 cars) for training and those in the last 5 images (209 cars) for testing. We did not use the cars whose center was nearer than 38 pixels from the image border, in order to avoid artifacts.

Model: We want to learn a covariant function with respect to rotations, since a rotation by $\Delta\alpha^\circ$ in the input image results in a change by $\Delta\alpha^\circ$ in the predicted angle. In particular, we train on sine and cosine of α°, since they are continuous with respect to $\Delta\alpha^\circ$. The network's architecture is illustrated in Table 4.7. For the output we use a tanh non-linearity, followed by a normalization of the output vector to unit-norm. The first fully connected layer (FC1) is a RotConv block with a single filter ($R = 21$) *not* followed by

Table 4.7: Architecture of the car orientation estimation network.Parameters are in white and variables are shaded in gray.

Type	**Size**
Input	48×48
RotConv	11×11, 3 filt.
OP	$38 \times 38 \times 4 \times 2$
RotConv	$11 \times 11 \times 3 \times 2$, 6 filt.
OP	$28 \times 28 \times 6 \times 2$
RotConv	$11 \times 11 \times 6 \times 2$, 3 filt.
OP, 2×2 SP	$9 \times 9 \times 3 \times 2$
RotConv fully	$9 \times 9 \times 3 \times 2$, 1 filt.
connected (FC1)	$1 \times 1 \times 21$
FC2, Hardcoded	$1 \times 1 \times 21$, 2 filt.
Output	$1 \times 1 \times 2$

by an Orientation Pooling, meaning that the subsequent feature vector has 21 dimensions instead of just one. We can expect this vector to undergo a circular shift when the input image is subject to a rotation. We hardcode the two mappings of the following layer (FC2) to $[\sin(360/R), \sin(2 \cdot 360/R), \ldots \sin(R \cdot 360/R)]$ and $[\cos(360/R), \cos(2 \cdot 360/R), \ldots \cos(R \cdot 360/R)]$. This ensures that there will be no preferred orientations inherited from a biased training set. The weight decay and learning rate are 10^{-2} and $5 \cdot 10^{-3}$ respectively, for the 80 epochs. All the filters were initialized from a normal distribution with zero mean and $\sigma = 10^{-3}$. The final models correspond to the average of the weights of the last 30 epochs.

Results: Table 4.8 reports the average test error. The use of RotEqNet substantially improves the results, outperforming by more than 20% the previous state-of-the-art method (Henriques and Vedaldi, 2016). In Figure 4.5, we show the error distribution in the test set for the hybrid model. Note how most samples, 82.7%, are predicted with less than 15° of orientation error, while most of the contribution to the total error comes from the 6.7% of samples with errors larger than 150°, in which the front of the car has been mistaken with the rear.

Table 4.8: Mean error in the prediction of car orientations.

Method	**Avg. error (°)**	# params
CNN Henriques and Vedaldi (2016)	28.87	27k
Warped-CNN Henriques and Vedaldi (2016)	26.44	27k
RotEqNet (Ours)	24.07	5k
RotEqNet (Ours)	**20.46**	9k

Sensitivity to R: In order to study the sensitivity of RotEqNet to the number of angles R, we trained the model using $R = 21$ and tested it for different values (see Figure 4.6).

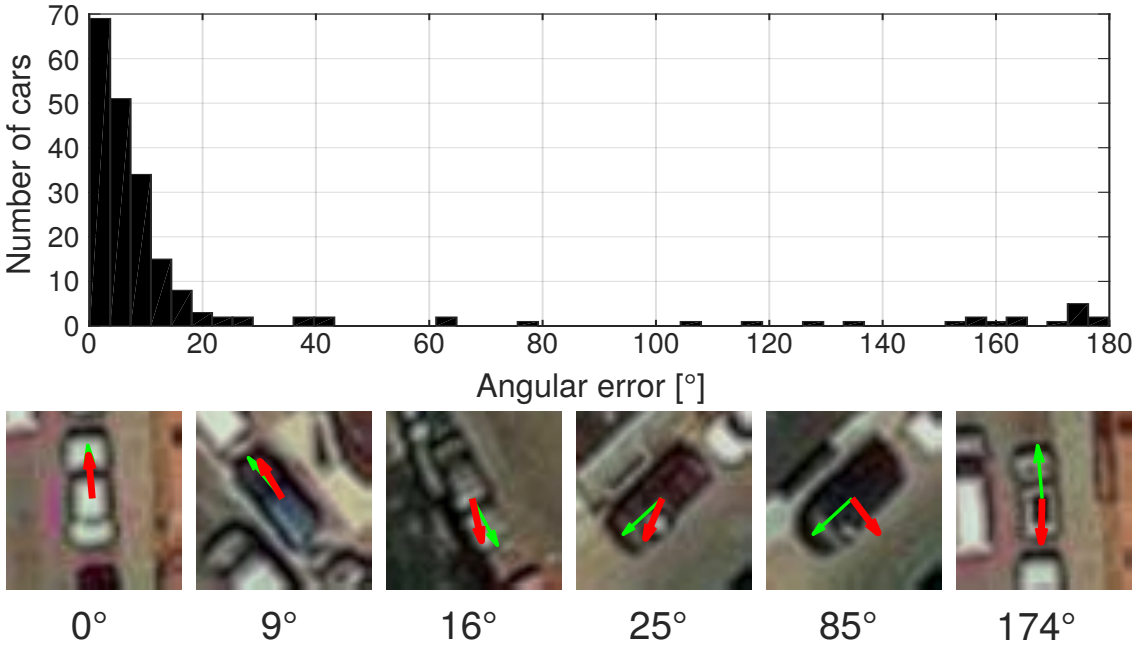


Figure 4.5: Distribution of the errors in the test set (top). Examples (bottom) of correctly and incorrectly identified orientations. Ground truth arrows in green (thin) and predictions in red (thick).

We observed relatively small changes in the test error for $R > 17$.

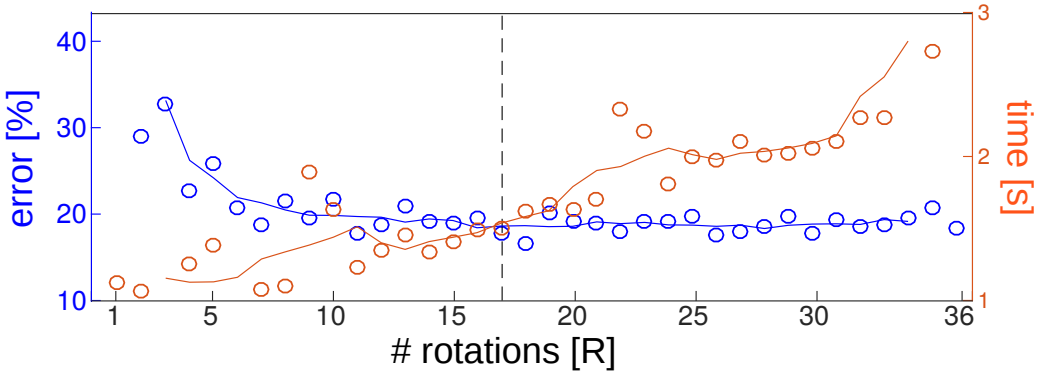


Figure 4.6: Error (left y-axis, blue) vs computational time (right y-axis, red) for the number of filters considered. The vertical dashed line denotes $R = 17$.

4.4.4 Invariance 2: robustness in patch matching

Patch matching is widely used in many image processing and computer vision problems, such as registration, 3D reconstruction and inpainting. The aim is to find matching pairs of patches (*e.g.* the same features in the two different images of the same object). In this setting, the differences in orientation are often considered to be a nuisance. Although handcrafted features such as SIFT are still widely used as baselines to measure similarity,

recent works have shown that learning ad-hoc features with siamese CNNs (Simo-Serra et al., 2015; Zagoruyko and Komodakis, 2015) can perform substantially better.In the following, we apply RotEqNet to analyze how this problem can benefit from a tunable amount of rotation invariance.

Depending on the problem at hand, one might have a prior on how much rotation invariance is required. Although CNN-based descriptors are more robust to relative rotations between matching pairs than SIFT, they still tend to perform poorly for large angular differences (Simo-Serra et al., 2015). To showcase how RotEqNet allows to tune the amount of rotation invariance, we trained a siamese network with three RotConv blocks, with 3, 6 and 32 filters of size 9×9 respectively, totaling 40k parameters. The last fully connected block provides 32 scalar features. We trained it on 20k samples from the Notredame dataset (Winder and Brown, 2007) with a distance-based objective function (Simo-Serra et al., 2015; Zagoruyko and Komodakis, 2015).

After training, the number of bins in the last Orientation Pooling layer can be modified, thus yielding multiple descriptors per sample. For instance, if the number of bins is set to 4, one 32-dimensional descriptor will be produced for each quadrant, thus resulting in a 128-dimensional descriptor for the patch. We analyze robustness in patch matching by increasing the rotation of the patches and the number of bins, and compare our results to those obtained by SIFT and the features from a pre-trained VGG network (Simo-Serra et al., 2015). We use patches extracted from an urban photograph that are then paired to a shifted (by one pixel) and rotated version of itself. Results in Fig.4.7 show that RotEqNet with a single bin is much more robust to rotations than VGG and SIFT descriptors, even when the main orientation assignment is used. As a trade-off, it performs slightly worse for small rotations. However, by increasing the number of bins we can invert this tendency and improve the matching accuracy for small angles (and trade off accuracy on large rotations): using two bins (i.e. a 64-dimensional descriptor), we clearly outperform the baselines on small angles and still have 60% of correct matches for rotations around 45° (compared to less than 10% for SIFT and VGG).

4.5 Limitations and future work

Forcing the Orientation Pooling block to choose the most activating orientation could result in exacerbating noise when there is no main orientation on either the input or the filter. This is because the arbitrarily chosen orientation can have a big impact on the output, and how it will interact with filters in the following layer, but no meaning. This problem is amplified by the use of scalar products between the vector elements of the filter and its input, which assumes that the orientation of these vectors is relevant. This issue could be improved by using a custom similarity metric between vector elements such that symmetries in the filters or the input are taken into account.

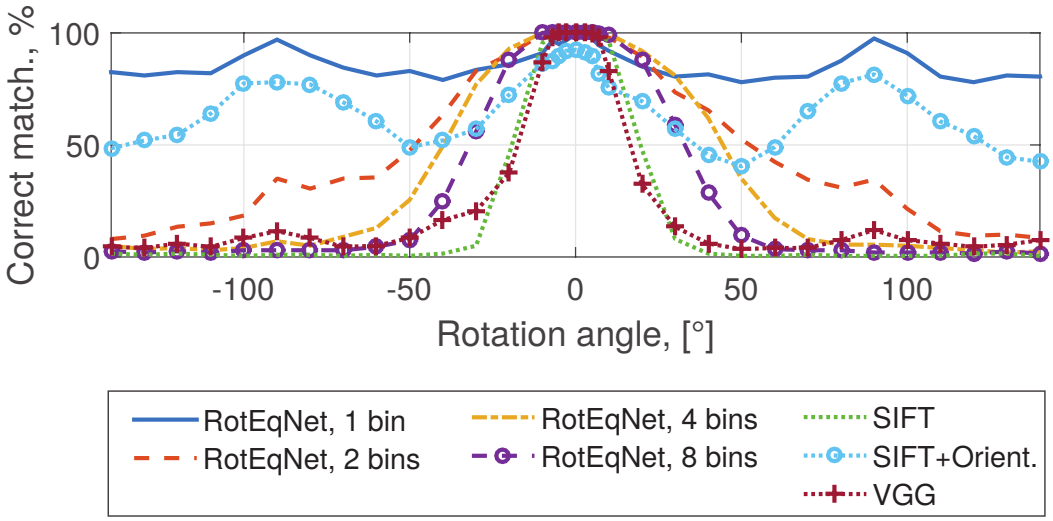


Figure 4.7: Matching accuracy vs. rotation applied to one of the elements in each matching pair in a synthetic dataset. RotEqNet allows to trade-off some accuracy at small rotations for more robustness by changing the number of bins in the last Orientation Pooling layer.

4.6 Conclusion

We have presented a new way of hard-coding into CNNs predefined behaviors with respect to rotations. This is achieved by applying each filter at different orientations and extracting a vector field feature map, encoding the maximum activation in terms of magnitude and angle.

Experiments on classification, segmentation, orientation estimation and matching show the suitability of this approach for solving a wide variety of problems that are inherently rotation equivariant, invariant or covariant. These results suggest that taking into account only the dominant orientations is sufficient to tackle successfully a range of problems.

Chapter 5

Land cover mapping at very high resolution with rotation equivariant CNNs: Towards small yet accurate models

This chapter is based on:

Marcos, D., Volpi, M., Kellenberger, B., and Tuia, D. (2018c). Land cover mapping at very high resolution with rotation equivariant cnns: Towards small yet accurate models. *ISPRS Journal of Photogrammetry and Remote Sensing*

Abstract

In remote sensing images, the absolute orientation of objects is arbitrary. Depending on an object's orientation and on a sensor's flight path, objects of the same semantic class can be observed in different orientations in the same image. Equivariance to rotation, in this context understood as responding with a rotated semantic label map when subject to a rotation of the input image, is therefore a very desirable feature, in particular for high capacity models, such as CNNs. If rotation equivariance is encoded in the network, the model is confronted with a simpler task and does not need to learn specific (and redundant) weights to address rotated versions of the same object class. In this work we propose a CNN architecture called Rotation Equivariant Vector Field Network (RotEqNet) to encode rotation equivariance in the network itself. By using rotating convolutions as building blocks and passing only the the values corresponding to the maximally activating orientation throughout the network in the form of orientation encoding vector fields, RotEqNet treats rotated versions of the same object with the same filter bank and therefore achieves state-of-the-art performances even when using very small architectures trained from scratch. We test RotEqNet in two challenging sub-decimeter resolution semantic labeling problems, and show that we can perform better than a standard CNN while requiring one order of magnitude less parameters.

5.1 Introduction

In this paper we consider the task of *semantic labeling*, which corresponds to the automatic assignment of each pixel to a set of predefined land-cover or land-use classes. The classes are selected specifically for the task to be solved and define the learning problem for the model.

When using low- to mid-resolution multispectral imagery (e.g. Landsat), it is customary to assume that the spectral information carried by a pixel is sufficient to classify it into one of the semantic classes, thus reducing the need for modeling spatial dependencies. However, when dealing with VHR imagery, i.e. imagery in the meter to sub-decimeter resolution range, the sensor trades off spectral resolution to gain spatial details. Such data is commonly composed of RGB color channels, occasionally with an extra NIR band. Due to this trade-off, single pixels tend not to contain sufficient information to be assigned with high confidence to the correct semantic class, when relying on spectral characteristics only. Moreover, depending on the task, some classes can be semantically ambiguous: a typical example is land use mapping, where objects belonging to different classes can be composed of the same material (e.g. road and parking lots), thus making analysis based on spectra of single pixels not suitable. To resolve both problems, spatial context needs to be taken into account, for example via the extraction and use of textural (Regniers et al., 2016), morphological (Dalla Mura et al., 2010; Tuia et al., 2015), tree-based (Gueguen and Hamid, 2015) or other types (Malek et al., 2014) of spatial features. These features consider the neighborhood around a pixel as part of its own characteristics, and allow to place spectral signatures in context and solve ambiguities at the pixel level (Fauvel et al., 2013). The diverse and extensive pool of possible features led to a surge in works focusing on the automatic generation and selection of discriminant features (Harvey et al., 2002; Glocer et al., 2005; Tuia et al., 2015), aimed at preventing to compute and store features that are redundant or not suited for a particular task.

Another common approach to reduce the computational burden while enforcing spatial reasoning is to extract local features from a support defined by unsupervised segmentation. Also, spatial rules can be encoded by Markov random fields, where spatial consistency is usually enforced by minimizing a neighborhood-aware energy function (Moser et al., 2013) or specific spatial relationships between the classes (Volpi and Ferrari, 2015b).

In the situations described above, a successful solution comes at the cost of having to manually engineer a high-dimensional set of features potentially covering all the local variations of the data in order to encode robust and discriminative information. In this setting, there is no guarantee that the features employed are optimal for a given semantic labeling problem. These problems raised the interest of the community in solutions avoiding to manually engineer the feature space, solutions that are extensively studied under the *deep learning* paradigm. The aim of deep learning is to train a parametric

system learning feature extraction *jointly* with a classifier (Goodfellow et al., 2016), in an end-to-end manner. When focusing on image data, Convolutional Neural Networks (CNNs, (LeCun et al., 1998)) are state-of-the-art. Their recognized success follows from new ground-breaking results in many computer vision problems. CNNs stand out thanks to their ability to learn complex problem-specific features, while jointly optimizing a loss (e.g. a classifier, a regressor, etc.). Thanks to recent hardware advances accelerating CNN training consistently, as well as the existence of pre-trained models to get started, CNNs have become one of the most studied models in recent remote sensing research dealing with VHR imagery, as we briefly review below.

The first models proposed studied the effectiveness of translating computer vision architectures directly to aerial data for tile classification. In that sense, a single label was retrieved per image tile, thus tackling what in computer vision is called the *image classification problem*[1]: authors in Castelluccio et al. (2015) and Penatti et al. (2015) studied the effect of fine-tuning models trained on natural image classification problems, in order to adapt them quickly to above-head image classification. Their results suggested that such a strategy is relevant for image classification and can be used to reuse models trained on a different modality. Transposing these model in the semantic labeling problem is also possible, typically applying the models using a sliding window centered at each location of the image, as tested in Campos-Taberner et al. (2016). However, the authors also came to three important conclusions: *i)* models trained from scratch (in opposition to fine-tuned models from vision) tend to provide better results on specific labeling tasks; *ii)* by predicting a single label per patch, the one corresponding to the pixel on which the patch is centered, these models are not able to encode explicit label dependencies in the output space and *iii)* the computational overhead of the sliding window approach is extremely large. Such conclusions support the use of network architectures that have been developed specifically for semantic labeling problems: recent efforts tend to consider *fully convolutional* approaches (Long et al., 2015), where the CNN does not only predict a single label per patch, but actually provides directly the label map for all the pixels that compose the input tile. The approaches proposed vary from spatial interpolation (Maggiori et al., 2017), fully convolutional models (Audebert et al., 2016), deconvolutions (Volpi and Tuia, 2017), stacking activations (Maggiori et al., 2016) to hybridization with other classifiers (Liu et al., 2017), but they all are consistent in one observation: fully convolutional architectures drastically reduce the inference time and naturally encode some aspect of output dependencies, in particular learning dependent filters at different scales, thus reducing the need of cumbersome postprocessing of the prediction map.

While these works open endless opportunities for remote sensing image processing with CNNs, they also showed one of the biggest downsides of these models: CNNs tend to need large amounts of ground truth to be trained, and setting up the architecture, as well as

[1]This is not to be confused with the *semantic labeling* problem we address in this paper, which is the task of attributing a label to every pixel in the tile.

selecting hyperparameters, can be troublesome, since cross-validation is often prohibitive in terms of processing time. Note that it is often that case when the number of parameters is larger than the number of training samples, which makes regularization techniques and data augmentation a must-do, at the cost of significantly slowing model training. Our contribution aims at addressing this drawback of CNNs, *i.e.* the large model sizes and need for labels when there is a limited availability of ground truth. In this paper, we propose to tackle the problem by exploiting a property of objects and features in remote sensing images: *their orientation is arbitrary.*

Overhead imagery differs from natural images in that the *absolute* orientation of objects and features within the images tends to be irrelevant for most tasks, including semantic labeling. This is because the orientation of the camera in nadir-looking imagery is most often arbitrary. As a consequence, the label assigned to an element in the image should not change if the image is taken with a different camera orientation. We call this property *equivariance*, and it is a property that recently attracted a lot of interest in image analysis (Lei et al., 2012; Cheng et al., 2016a).

Given a rotation operator, $g_\alpha(\cdot)$, we say that a function $f(\cdot)$ is equivariant to rotations if $f(g_\alpha(\cdot)) = g_\alpha(f(\cdot))$, invariant to rotations if $f(g_\alpha(\cdot)) = f(\cdot)$ and, more generally, covariant to rotations if $f(g_\alpha(\cdot)) = h(f(\cdot))$, with $h(\cdot)$ being some function other than $g_\alpha(\cdot)$. Note that, in the case of semantic labeling, the property we are interested in is equivariance, although it becomes invariance if we consider a single pixel at a time. We will therefore use the terms equivariance and invariance interchangeably in this paper.

With CNNs, equivariance to the rotation of inputs can be approximated by randomly rotating the input images during training, a technique known as *data augmentation* or *jittering* (Leen, 1995). If the CNN has enough capacity and has seen the training samples in sufficient number of orientations, it will learn to be invariant to rotations (Lenc and Vedaldi, 2015). While this kind of data augmentation greatly increases the generalization accuracy, it does not offer any advantage in terms of model compactness, since similar filters, but with different orientations, need to be learned independently. A different approach, hard coding such invariances within the model, has the two main beneficial effects: first, the model becomes robust to variations which are not discriminative, as a standard CNN with enough filters would learn; and second, model-based invariance can be interpreted as some form of regularization (Leen, 1995). This added robustness ultimately lead to models which have high capacity (as high as a standard CNN) but with lower sample complexity.

There has been a recent surge in works that explore ways of encoding model-based rotation invariance in CNNs. (Laptev et al., 2016) perform a rotation of the input image in order to reduce the sample complexity of the problem and (Jaderberg et al., 2015) extend this to affine transformations. These approaches provide invariance to a global rotation of the input image and not to local relative rotations, and are therefore not very well suited

for segmentation tasks. (Cohen and Welling, 2016a) encode equivariance to shifts and to rotations by multiples of 90^o by tying filter weights, while (Zhou et al., 2017) use linearly interpolated filters. These two methods are in principle suited for segmentation tasks. The former is limited to invariance to 90^o rotations and the latter, although offering more flexibility, has the drawback of requiring a trade-off between the number of rotations and the memory requirements, bringing the authors to use 8 orientations, at multiples of 45^o. (Worrall et al., 2016) reduce the space of possible filters to combinations of complex harmonic wavelets, thusachieving perfect equivariance to rotations. By doing so, they obtain a more compact internal representation by encoding oriented activations as complex valued feature maps, but at the cost of reducing the expressiveness of each filter.

In this paper, we consider a solution that combines the advantages of these two last methods. Our model applies an arbitrary number of rotated instances of each filter at every location, in a way that each filter activation is composed by *a vector* of activations (as opposed to a scalar in standard CNN), thus representing the activation of each rotated filter. We then propose to max-pool these activations, compressing the information in a simple 2D vector that represents the magnitude and orientation of the maximally activating filter. This allows us to encode fine-grained rotation invariance (*i.e.* very small angles) and, at the same time, to avoid constraining the filters to any particular class, thus enabling more expressive filters. The proposed Rotation Equivariant Vector Field Network (RotEqNet, 4) achieves model-based invariance while reducing the number of required parameters by around one order of magnitude. This is done by sharing the same convolutional filter wights across all angles, thus providing regularization to irrelevant modes of variations (Leen, 1995).

In addition, the decrease in sample complexity allows models to be trained more efficiently. In this paper, we also analyze the effect that the amount of available ground truth has on the performance of CNNs learning for semantic labeling of overhead imagery based on two public datasets at submetric resolution: the Vaihingen and the Zeebruges benchmarks (see Section 5.3).

In Section 5.2 we briefly present the intuition behind RotEqNet, as well as its main components. In Section 5.3 we present the data and the setup of the experiments presented and discussed in Section 5.4.

5.2 Rotation Equivariant Vector Field Networks (RotEqNet)

In this paper, we propose to make a CNN equivariant to rotations by rotating filters and considering only the maximal activation across rotations. This section first recalls basics

about CNNs and then presents the RotEqNet modules as a way of extending any CNN architecture into a rotation equivariant one. For more details about RotEqNet, the reader can refer to chapter 4.

5.2.1 Convolutional neural networks for semantic labeling

In this section we briefly present the building blocks of CNNs, as well as an example of a fully convolutional architecture to perform semantic labeling.

CNN building blocks

CNNs consist of a cascade of operations applied to an input image $\mathbf{x} \in \mathbb{R}^{M \times N \times d}$ such that it can be nonlinearly transformed in the desired output. The number of such operations defines the *depth* of the network. CNNs are often organized in convolutional blocks as the one depicted in Figure 5.1.

The convolution operator

$$\mathbf{y} = \mathbf{x} * \mathbf{w} + b, \tag{5.1}$$

between $\mathbf{x}$ and a filter $\mathbf{w} \in \mathbb{R}^{m \times m \times d}$, where $b \in \mathbb{R}$ is the bias, produces a feature map $\mathbf{y} \in \mathbb{R}^{M-m+1 \times N-m+1}$ by applying locally a scalar product operation between $\mathbf{w}$ and every patch in $\mathbf{x}$ of size $m \times m$ in a sliding window manner. A convolution block in a CNN corresponds to the convolution of the image with a series of filters, which are represented in different colors in Figure 5.1.

The dimensionality of the activations equals the number of filters in the layer. To control the spatial extent of the activations after convolutions, it is common to apply zero-padding, which does not influence the value of the activations, but does compensate for the amount of pixels lost at the borders of the image. In order to obtain an advantage from the depth of the model, in terms of expressive power, it is necessary to apply some non-linear operation to the output of each convolution. The most common is the rectified linear unit (ReLU), which clips the negative values to zero, as $y = \max(0, x)$.

Once the activations are obtained, they are often pooled in the spatial domain, for example by taking the maximum value occurring in a very small (usually $2\times$ 2) local window. This operation, called *max-pooling* is represented in Figure 5.1 by the red squares and, besides the obvious effect of reducing the amount of data, also allows the filters in the next layer to 'see' more context of the original image: looking again at the schematic in Figure 5.1, if the first filters see a 3×3 region, those of the second layer (orange cube on the right) will see a 3×3 region in the reduced activations map, which coresponds to a 7×7 region in the original image. By cascading several convolutional and max-pooling blocks, the network actually becomes aware of a wide context around the pixel being considered,

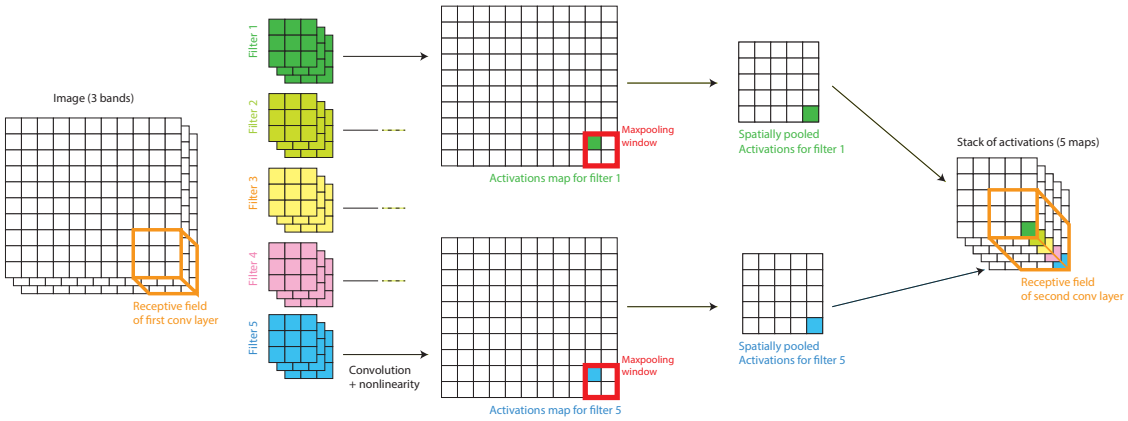


Figure 5.1: Schematic of the first convolutional layer of a CNN. This layer learns Nf = 5 filters of size $3 \times 3 \times 3$ and applies a spatial pooling halving the size of the activation map (only two out of the five activation maps are shown for clarity). In the activation maps, the colored pixels (in green or blue) correspond to those receiving information from the receptive field marked in orange in the input image (left).

while reducing the number of required learnable parameters, and provides invariance to local (at each layer) and global (at the network level) translations. The latter is evident in image classification problems: an image contains a cat independently of where it is located. For semantic labeling tasks, max-poolings have the effect of learning locally consistent and multi-scale filters.

Two other operators are often used to improve the learning process: batch normalization and dropout. The former normalizes each feature map within a batch to have zero mean and unit standard deviation. The latter sets a certain proportion of randomly selected feature maps to zero during training, thus preventing the filters from depending too much on one another.

From patch classification to (dense) semantic labeling

Early CNN models in vision were designed for tile (or image) classification, i.e. to provide a single label for an image. In semantic labeling, we are interested in obtaining a dense prediction, i.e. a map where each pixel is assigned to the most likely label. As stated in the introduction, this can be achieved in a number of ways, including fully convolutional models (Long et al., 2015). A very simple way to perform dense predictions is to use the activation maps themselves as features to train a classifier predicting the label of every pixel. If max-pooling operations have been performed, spatial upsampling, e.g. by interpolation, is required to bring all activations to the same spatial resolution. One of these approaches, known as "hypercolumns" (Hariharan et al., 2015), using fixed upsamplings, is represented in Figure 5.2. It follows the intuition that the different activation maps con-

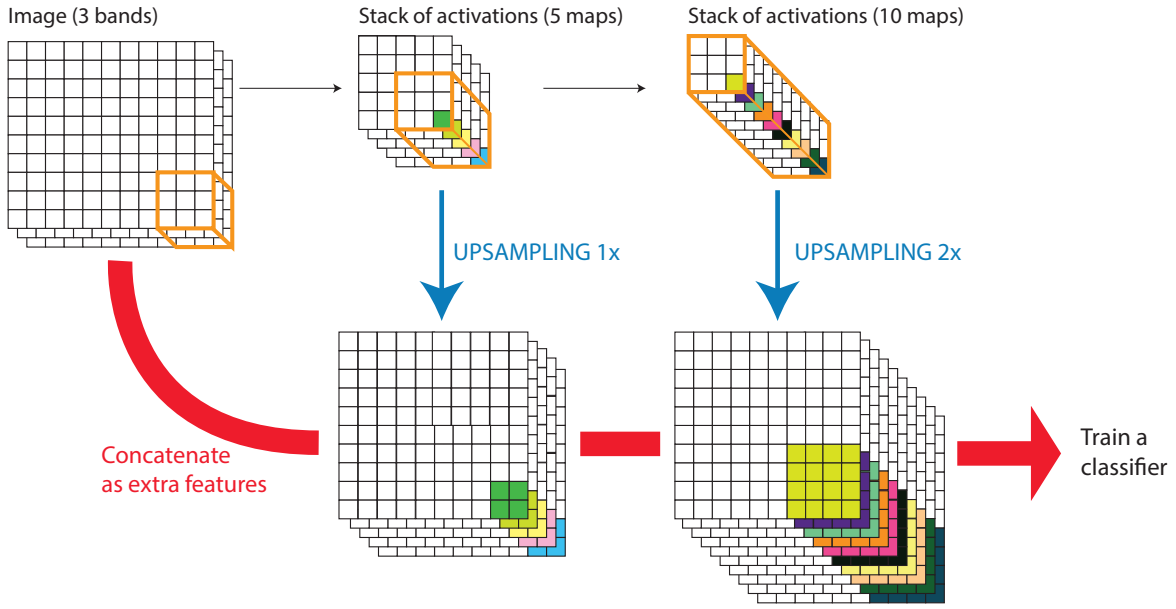


Figure 5.2: Schematic of the model considered for dense semantic labeling. Each activation map in the CNN (top part) is upsampled at the original image resolution (blue arrows), concatenated to the original image (red arrow) and fed to a local fully connected layer (1×1 convolutions), in this example using 18 features.

tain information about specific features extracted at different scales, from low-level ones (first layers react to corners, gradients, etc.) to more semantic and contextual ones (last layers activate to class-specific features). Therefore, a stack of such features can be used to learn an effective classifier for dense semantic labeling tasks, where spatial information is crucial. In remote sensing, the idea of hypercolumns was used in the architecture proposed by (Maggiori et al., 2016). In the experiments, we will use this architecture for dense semantic labeling, using two fully connected layers as classifier (see Section 5.3.2 for details), and train the model end-to-end.

5.2.2 From translation to rotation invariance

As mentioned above, CNNs are translation equivariant by design (see Section 2.4.4). To understand why it is more complex to achieve natural equivariance to rotations by means of convolutions, we will briefly summarize how translation equivariance is obtained by standard CNNs before moving to rotation equivariance and our proposed solution, RotEqNet.

Translation equivariance in CNNs

The convolution of an image $\mathbf{x}$ with a filter $\mathbf{w}$ (Eq. (5.1)) is computed by applying the same dot product operation over all overlapping $m \times m$ windows on $\mathbf{x}$. If $\mathbf{x}$ is shifted by an integer translation in the horizontal and vertical directions, given a reference location, the same neighborhoods in $\mathbf{x}$ will exist in the translated $\mathbf{x}$. The corresponding convolution output, except for some possible border effects, are exactly the same up to some global translation constant. For this reason, neighborhood-based operations are translation equivariant when applied to images. The fact that the operation is local and produces a single scalar per neighborhood has another advantageous effect: the output can be effortlessly re-arranged in useful ways. Typically, the spatial structure of the activations is set to match the one of the input (as in CNNs, see Figure 5.1).

Rotation equivariance in RotEqNet

If we want the operator to be equivariant to rotation, the structure of the layer activations becomes more complex. One possibility would be to return a series of values corresponding to the convolution of $\mathbf{x}$ with rotated versions of the canonical filter $\mathbf{w}$. In this case, the activations $\mathbf{y}$ would be a 3D tensor where a translation in the 3^{rd} dimension corresponds to a rotation of $\mathbf{w}$. The covariance achieved in this way could easily be transformed into equivariance by means of pooling across orientations, since the value returned at each image location will remain constant when the image is rotated and thus a rotation of the input image will result in the same rotation of the output feature map.

In particular, we propose to perform a single-binned max-pooling operation across the newly added orientation dimension. At each location, it fires on the largest activation across orientations, returning its value (magnitude) *and* the angle at which it occurred. This way, we are able to keep the 2D arrangement of the image and activations throughout the CNN layers, while achieving rotation equivariance as provided by this pooling strategy. Furthermore, this strategy allows the network to make use of the information about the orientation of feature activations observed in previous layers. Similar to spatial max-pooling, this *orientation pooling* propagates only information about the maximal activation, discarding all information about non-maximal activations. This has the drawback of potentially loosing useful information (e.g. when two orientations are equally discriminant), but offers the advantage of reducing the memory requirements of both the model and the feature maps along the network, making them independent of the number of rotations used. Since the result of such pooling is no longer a scalar as in conventional CNNs, but a 2D vector (magnitude and angle), each activation map can now be treated as a vector field. Figure 5.3 schematizes this intuition and shows a RotEqNet convolutional block in comparison to the standard CNN convolutional block of Figure 5.1.

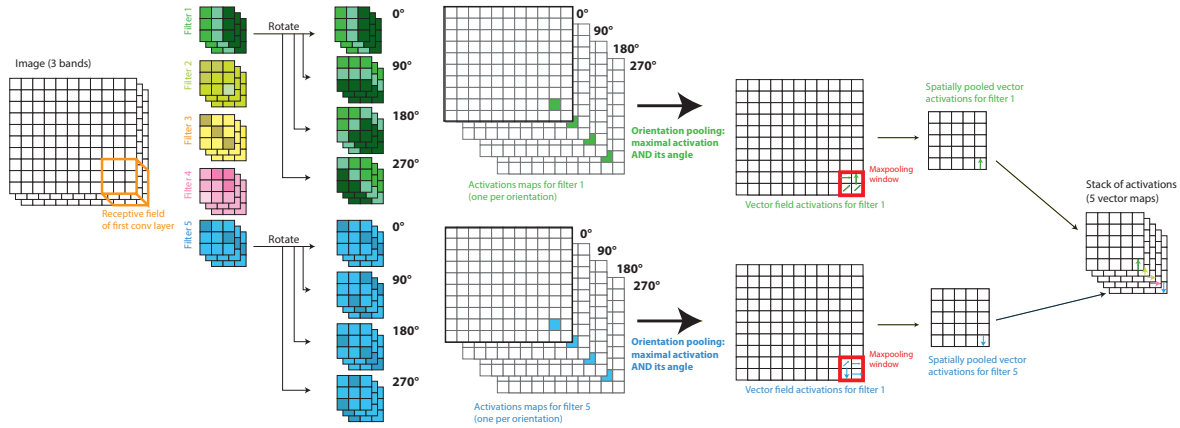


Figure 5.3: Schematic of the first convolutional layer of RotEqNet. This layer learns Nf = 5 filters of size $3 \times 3 \times 3$. Each filter is rotated to a pre-defined range of angles and the activation at each orientation is computed. Then an orientation pooling retains only the maximal activation and the angle that generated it, thus providing a vector activation per pixel (represented by colored arrows). This vector map is pooled spatially as for conventional CNNs. The output is a stack of vector activations. In the activation maps, the colored pixels (in green or blue) correspond to those receiving information form the receptive field marked in orange in the input image (left). For clarity, only two out of the five activation maps are shown.

5.2.3 RotEqNet modules

RotEqNet essentially involves rotating CNN filters and pooling across the orientation space to retrieve the maximal activations and their angle observed at each location and per filter. To achieve such behavior, several building blocks of CNNs must be re-designed in order to accommodate vector field inputs / outputs. In this section, we briefly summarize how the convolution and pooling operators have been modified. Modifications of spatial pooling and batch normalization are straightforward and we invite the reader to consult Chapter 4 for more details.

Rotating convolution

As introduced above, rotation equivariance can be achieved by computing the activations on a series of rotated versions of the filters being learned. This boils down to calculate rotated versions of each main (or *canonical*) filter at R orientations $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_R]$. In case of remote sensing images, for which the orientation might be completely arbitrary, $\boldsymbol{\alpha}$ can span the entire 360° rotation space, while in other applications with a clear top-down relations, one could limit the angles to a smaller range (e.g. it is unlikely that a tree depicted in a ground level image is oriented in the left-right direction, but some tilts due to camera shake could be present).

The rotation of the filter is obtained by resampling $\mathbf{w}$ with bilinear interpolation, after rotation of α_r degrees around the filter center. The position $[i', j']$ after rotation of a specific filter weight, originally located at $[i, j]$ in the canonical form, is

$$[i', j'] = [i, j] \begin{bmatrix} \cos(\alpha_r) & \sin(\alpha_r) \\ -\sin(\alpha_r) & \cos(\alpha_r) \end{bmatrix}. \tag{5.2}$$

Coordinates are relative to the center of the filter. Since the rotation can force weights near the corners of the filter to be relocated outside of its spatial support, only the weights within a circle of diameter m pixels, the filter size, are used to compute the convolutions. The output tensor for filter $\mathbf{w}$, of size $\mathbf{y} \in \mathbb{R}^{M \times N \times R}$, consists of R feature maps (see the center part of Figure 5.3), each one computed as

$$\mathbf{y}^{(r)} = \mathbf{x} * \mathbf{w}_r + b \quad \forall r = 1, 2 \dots R, \tag{5.3}$$

where $(*)$ is a standard convolution operator, and b is a shared bias across all rotations. The tensor $\mathbf{y}$ encodes the roto-translation output space such that rotation of the input corresponds to a translation across the feature maps. Only the canonical, non rotated, version of $\mathbf{w}$ is actually stored in the model. During backpropagation, gradients flow through the filter with maximal activation, very similarly to the max-pooling case. Consequently, the gradients have to be aligned to the rotation of the canonical filter, which is recovered thanks to the angle as given by the orientation pooling. Thus, filters are updated as:

$$\nabla \mathbf{w} = \sum_r \texttt{rotate}(\nabla \mathbf{w}_r, -\alpha_r), \tag{5.4}$$

Orientation pooling

The rotating convolution above outputs a set of R activations per canonical filter, each one corresponding to one rotated version of $\mathbf{w}$. To avoid the explosion of dimensionality related to the propagation of all these activations to the next layer, we perform a pooling across the space of orientation aiming as pushing forward only the information relative to the direction of maximal activation. In order to preserve as much information as possible, we keep two kinds of information: the *magnitude of activation* and the *orientation that generated it*.

To do so, we extract a 2D map of the largest activations $\boldsymbol{\rho} \in \mathbb{R}^{M \times N}$ and their corresponding orientations $\boldsymbol{\theta} \in \mathbb{R}^{M \times N}$. Specifically, for activations located at $[i, j]$ we get:

$$\rho[i, j] = \max_r y[i, j, r], \tag{5.5}$$

$$\theta[i, j] = \frac{360}{R} \arg\max_r y[i, j, r]. \tag{5.6}$$

This can be treated as a polar representation of a 2D vector field as long as $\rho[i,j] \geq 0 \quad \forall i,j$. This condition is met when using a function on $\mathbf{y}$ that returns non-negative values: we therefore employ the ReLU operation, defined as $\texttt{ReLU}(\rho) = \max(\rho, 0)$. In the backward pass the magnitude of the incoming 2D vector gradient is passed to the corresponding maximally activated position, $y[i,j,r_{max}]$, as is done with standard max-pooling.

Dealing with vector inputs

Note that the orientation pooling block outputs vector fields, where each location in the activation carries both the maximal magnitude and its orientation observed in polar representation (see the rightmost matrix in Figure 5.3). This means that the output of such pooling is vectorial and cannot be used anymore in a traditional convolutional layer. However, if we convert this polar representation into Cartesian coordinates, each filter $\mathbf{w}$ produces a vector field feature map $\mathbf{z} \in \mathbb{R}^{M \times N \times 2}$, where the output of each location consists of two values $[u,v] \in \mathbb{R}^2$ encoding the same information.

$$\mathbf{u} = \texttt{ReLU}(\boldsymbol{\rho}) \cos(\boldsymbol{\theta}) \tag{5.7}$$

$$\mathbf{v} = \texttt{ReLU}(\boldsymbol{\rho}) \sin(\boldsymbol{\theta}) \tag{5.8}$$

Since the horizontal and vertical components $[u,v]$ are orthogonal, the convolution of two vector fields can be computed summing standard convolutions calculated separately in each component:

$$(\mathbf{z} * \mathbf{w}) = (\mathbf{z}_u * \mathbf{w}_u) + (\mathbf{z}_v * \mathbf{w}_v), \tag{5.9}$$

By using this trick, we can now calculate convolutions between vector fields and design deep architectures which are rotation equivariant.

5.3 Data and setup

5.3.1 Datasets

We test the proposed system on two recent benchmarks that raised significant interest thanks to the dense ground truth provided over a set of sub-decimeter resolution image tiles: the Vaihingen and Zeebruges data, which are briefly described below. Both datasets consist of three optical bands and a Digital Surface Model (Digital Surface Model (DSM)).

Since since using the DSM has been shown to improve the segmentation results ((Audebert et al., 2017; Marmanis et al., 2016; Volpi and Tuia, 2017)) we use it in all of our experiments. We do this by stacking the DSM with the optical data and treating it as an additional band, as in (Volpi and Tuia, 2017), since this has almost no impact in the total number of model parameters.

Vaihingen benchmark

The Vaihingen benchmark dataset has been provided to the community as a challenge organized by the International Society for Photogrammetry and Remote Sensing (ISPRS) Commission III.4, the "2D semantic labeling contest"[2]. The dataset is composed of 33 orthorectified image tiles acquired over the town of Vaihingen (Germany), with an average size of 2494×2064 and a spatial resolution of 9 cm. Among the 33 frames, 16 are fully annotated and distributed to participants, while the remaining ones compose the test set and their ground truth is not distributed. Images are composed by 3 channels: near infrared (NIR), red (R) and green (G). The challenge also provides a DSM coregistered to the image tiles. We use a normalized version of the DSM (Normalized Digital Surface Model (nDSM)), where the heights are relative to the nearest ground pixel, redistributed by (Gerke, 2015). One of the training tiles is illustrated in Figure 5.4.

The task involves 6 land-cover / land-use classification classes: "impervious surfaces" (roads, concrete flat surfaces), "buildings", "low vegetation", "trees", "cars" and a class of "clutter" to group uncategorized surfaces and noisy structures. Classes are highly imbalanced, with the classes "buildings" and "impervious surfaces" accounting for roughly 50% of the data, while classes such as "car" and "clutter" account only for 2% of the total labels.

In our setup, 11 of the 16 fully annotated image tiles are used for training, and the remaining ones (tile ID 11, 15, 28, 30, 34) for testing, as in (Sherrah, 2016; Volpi and Tuia, 2017; Maggiori et al., 2017).

Zeebruges benchmark

This benchmark has been acquired in 2011 over the city of Zeebruges (Belgium) and it is has been provided as part of the IEEE GRSS Data Fusion Contest in 2015 Campos-Taberner et al. (2016)[3]. It is composed by seven tiles of 10000×10000 pixels. The tiles have a spatial-resolution of 5 cm and represent RGB channels only. Five of the seven images are released with labels Lagrange et al. (2015) and used for training, while the

[2] `http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html`

[3] `http://www.grss-ieee.org/community/technical-committees/data-fusion/2015-ieee-grss-data-fusion-contest/`

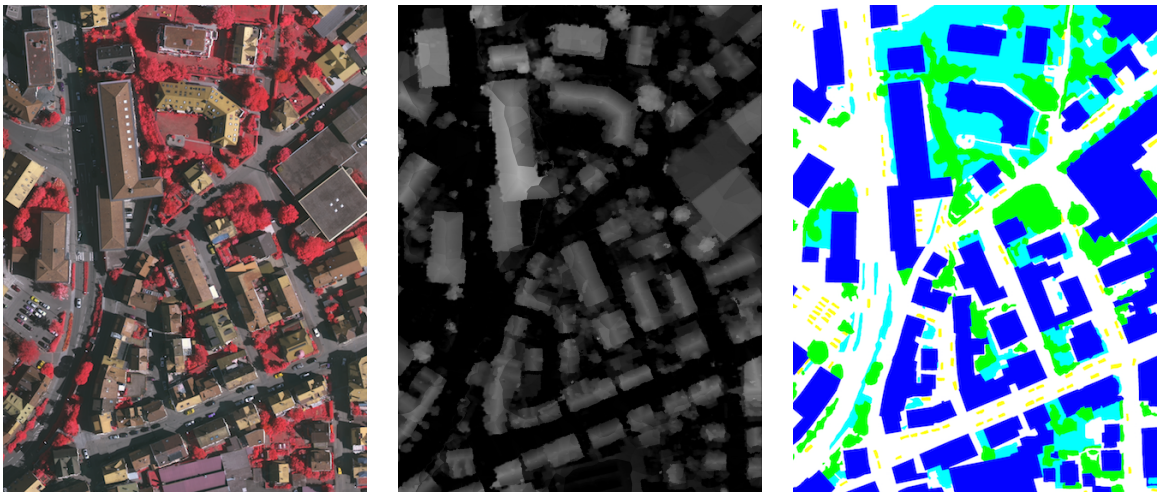


Image tile #1 nDSM ground truth

Figure 5.4: one of the training tiles of the Vaihingen dataset: left: image; center: nDSM; right: ground truth.

remaining two are kept for testing the generalization accuracy, accordingly to the challenge guidelines. This dataset also comes with a Lidar point cloud, that we processed into a DSM by averaging point clouds locally and interpolating where necessary. One of the training tiles is illustrated in Figure 5.5.

The semantic labeling problem involves 8 classes, as proposed in Lagrange et al. (2015): the same six as in the Vaihingen benchmark, plus a "water" and "boats" class. It is worth mentioning that, since a large portion of the are is covered by a harbour, most of the structures and cargo containers are labeled as "clutter". Another major difference to the Vaihingen dataset is that the "Water" class is predominant, as it represents 30% of the training data, while cars and boats together count just a mere 1%. Also, 1% of the data belongs to an "uncategorized" class, which is not accounted for in the labeling problem. The lack of a NIR channel and a higher sample diversity make this benchmark more challenging than the previous one, as we will see in Section 5.4.

5.3.2 Experimental setup

CNN architecture

We use a RotEqNet architecture based on hypercolumns (Hariharan et al., 2015), in which every convolutional layer before the concatenation is a rotating convolution. After the concatenation, only standard fully connected layers are used, since 1×1 convolutions are inherently rotation equivariant. See Figure 5.6 for a schematic of the full architecture used. In both experiments we build the baseline CNN with the exact same architecture

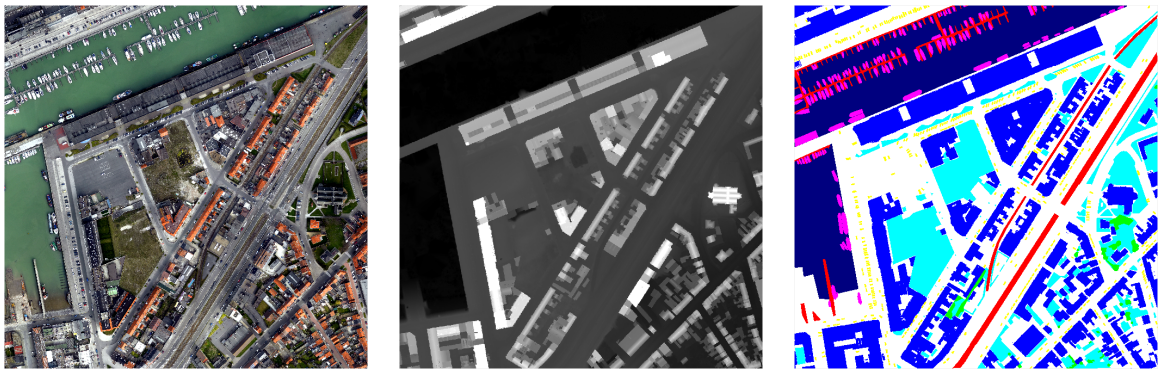


Image tile RGB nDSM ground truth

Figure 5.5: one of the training tiles of the Zeebruges dataset: left: image; center: nDSM; right: ground truth.

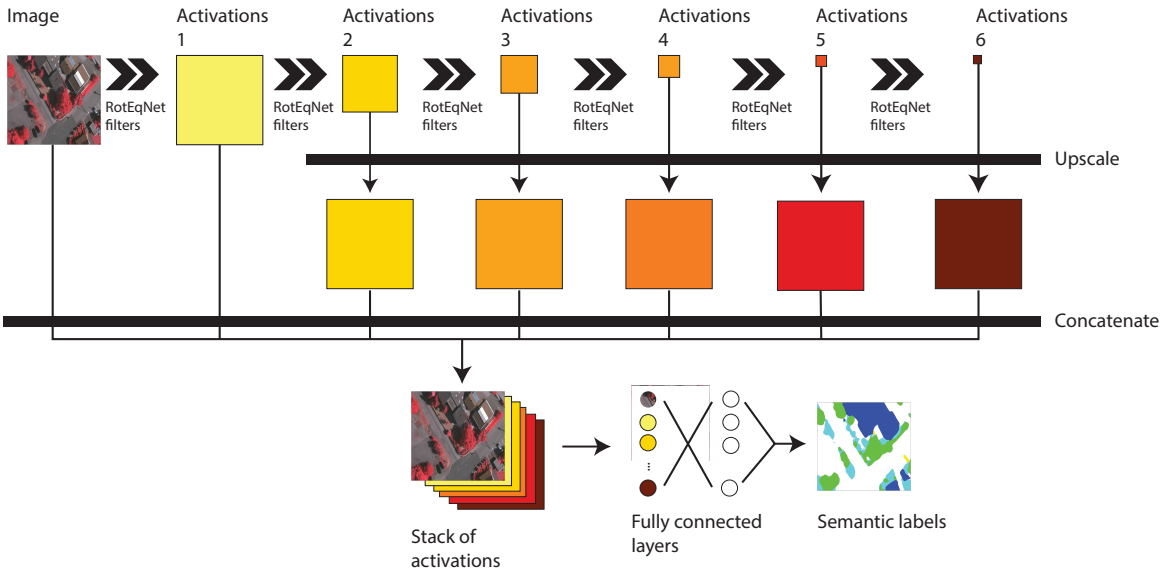


Figure 5.6: Hypercolumn based architecture used in all our experiments. Note that all the layers are rotation equivariant, since all the convolutional layers are either RotEqNet convolutions or fully connected (1×1) standard convolutions, which are also rotation equivariant by construction.

as its RotEqNet counterpart, but with four fold more filters in each layer, resulting in approximately 10 fold more parameters. At this model size the performance started to saturate.

We use an architecture with six convolutional layers, each with downsampling by a factor of 2 using max-pooling. The number of filters per layer in each of the convolutional layers is set as $[2, 2, 3, 4, 4, 4] \cdot \mathrm{Nf}$, where the i^{th} element of the vector represents the number of filters in the i^{th} layer, such that Nf is the only parameter used to change the size of the models. All the convolutional layers use 7×7 filters. This size allows to capture oriented

patterns in the corresponding image or feature map, as seen in Figure 5.7. After applying ReLU and batch normalization, each activation map is then upsampled to the size of the original image using bilinear interpolation and concatenated, together with the raw image (see bottom part of Figure 5.2), and processed with three layers of 1×1 convolutions with $[50 \cdot \mathrm{Nf}, 50 \cdot \mathrm{Nf}, C]$ filters, where C is the number of classes. This is followed by a softmax normalization. The 1×1 convolutions implement a local fully connected layer, or, in other terms, performs local classification by a Multi-Layer Perceptron (MLP). These 1×1 convolutions are inherently rotation equivariant, so we use standard convolutions as in Long et al. (2015). The whole pipeline is learned jointly end-to-end.

Our model performs arbitrarily dense prediction, i.e. given an arbitrarily sized input patch the output will always be a prediction map with the same size. As a result, the CNN architecture is fixed and the dataset reshaped to a series of fixed-sized patches. For the Vaihingen dataset, the spatial extent of the inputs is 512×512, while for Zeebruges is 500×500. Note that the input size does not influence the results.

All models are trained with stochastic gradient descent with momentum (fixed to 0.9), while other hyperparameters are tuned by minimizing validation errors on 30 samples randomly selected from the training set. The batch size is 4 in RotEqNet and 2 in the standard CNN, because of the latter's higher memory requirements. In both benchmarks, we perform data augmentation consisting of random rotations, uniformly sampled between 0^o and 360^o, and randomly flipping the tiles in the vertical or horizontal dimension. Note that performing full 360^o rotations for data augmentation is not strictly necessary when using RotEqNet, but it has been shown to be additionally improve the performance (see results in) and doing so makes a comparison with standard CNNs easier, since they are trained under more similar conditions. As will be discussed below, data augmentation is required by standard CNNs in order to be able learn rotation invariance from examples, given that enough filters can be learned. Regarding RotEqNet, rotating inputs does not have a direct effect on learning diverse filters, but rather on data interpolation making a same input tile looking different numerically (an effect also improving training for standard CNNs).

All models are trained from scratch and filters are initialized using the improved Xavier method.

The hardware used in all experiments consists of a single desktop with 32 GB of RAM and a 12 GB Nvidia Titan X GPU.

5.3.3 Experimental Setup

5.3.4 Vaihingen

In the case of Vaihingen, we report a comparison between RotEqNet and standard CNNs trained without rotating convolutions. In order to test the sensitivity to the amount of ground truth, we train three models per architecture, using respectively 4%, 12% and 100% of the available training set. We compare architectures with the same structure and number of layers, only varying the number of filters. We compare a small RotEqNet model with a CNN of larger capacity (but no built-in rotation equivariance). The size of both models was chosen to be the smallest that would obtain over 87% overall accuracy on the validation set, which is in line with the results published in (Volpi and Tuia, 2017). The final number of filters defined in this way was found to be Nf= 3 for RotEqNet ($\approx 10^5$ parameters) and Nf= 12 for the standard CNN ($\approx 10^6$ parameters). The models using the full dataset were trained for 22 epochs. In the RotEqNet models the learning rate was $2 \cdot 10^{-2}$ in the first 11 epochs, followed by six epochs at $4 \cdot 10^{-3}$ and five at $8 \cdot 10^{-4}$, while the weight decay was $4 \cdot 10^{-2}$, $4 \cdot 10^{-3}$ and $8 \cdot 10^{-4}$ respectively. In the standard models those values were halved. This difference is due to a larger number of gradient update iterations in the standard CNN caused by the need to use a smaller mini-batch due to the larger memory requirements. For the experiments with a reduced training set the number of epochs was increased such that all the models would see the same number of iterations (*i.e.* mini-batches).

5.3.5 Zeebruges

In the case of Zeebruges, we compare RotEqNet with results from the literature Campos-Taberner et al. (2016), as they report results obtained with much larger architectures, both pre-trained or learned from scratch. Since this dataset is more complex than the previous one, we increased the model size and trained three RotEqNet models with Nf= $[4, 5, 7]$. The training schedule consisted of 34 epochs, with the first 12 epochs using a learning rate of $1 \cdot 10^{-2}$, 12 more with $2 \cdot 10^{-3}$ and 10 at $4 \cdot 10^{-4}$. The weight decay for the same segments was set to $6 \cdot 10^{-2}$, $1.2 \cdot 10^{-2}$ and $2.4 \cdot 10^{-3}$ respectively.

5.4 Results and discussion

5.4.1 Vaihingen

Table 5.1 shows the results in terms of the per class F1 scores, the overall accuracy (OA) and average accuracy (AA) for the experiments on the Vaihingen dataset. We

observe that both models reach over 87% OA when using the whole dataset, in line with recent publications and with the accuracy obtained by RotEqNet in the withhold test set, evaluated as 87.6% by the benchmark server. This only drops to around 84.7% when just 4% of the training set is used, suggesting that this dataset is highly redundant.

Table 5.1: Results on the Vaihingen validation set. F1 scores per class and global average (AA) and overall accuracies (OA). Best result per row is in dark gray, second in light gray.

Model	RotEqNet			CNN			CNN-FPL*	ORN
# params.	10^5			10^6			10^7	10^5
% train set	4%	12%	100%	4%	12%	100%	100%	100%
Impervious	88.0	88.7	89.5	86.9	88.8	**89.8**	-	88.1
Buildings	94.1	94.6	94.8	92.5	92.9	94.6	-	**95.4**
Low veg.	71.6	75.6	**77.5**	74.5	74.5	76.8	-	70.9
Trees	82.3	85.6	86.5	83.3	84.4	86.0	-	**92.1**
Cars	62.7	62.5	**72.6**	52.7	54.4	54.5	-	59.7
OA	84.6	86.6	87.5	84.8	85.5	87.4	**87.8**	87.0
AA	78.4	80.5	**83.9**	76.1	77.1	78.2	81.4	81.2

* = from (Volpi and Tuia, 2017)

The advantage of using RotEqNet becomes more apparent when measuring the AA, mostly because of an improved accuracy detecting the car class. We hypothesize that RotEqNet might be better suited to detect objects with clear and consistent boundaries, such as cars, because it is being forced to learn oriented filters, better adapted to detect edge-like features. Surprisingly, RotEqNet improves its performance gap with respect to the standard CNN when the amount of available ground truth increases. This suggests that encoding rotation invariance allows the model to concentrate more on solving the semantic labeling task, rather than having to learn to be invariant to rotations.

As a comparison, we show the results recently published by (Volpi and Tuia, 2017) using a much larger model and those obtained by applying the method by (Zhou et al., 2017) (ORN) with a model of the same architecture and size as ours.

In order to glimpse at what is being learned by both models, in Figure 5.7 we show all the 7×7 filters learned in the first convolutional layer in each model. Note that the values near the corner of the filters in the RotEqNet model are zero because the support of the filter is a disk circumscribed in the 7×7 grid. Out of the six filters learned by RotEqNet in the first layer, three seem to have specialized in learning corner features involving vegetation (the reddish tones means high response to the near infrared channel), one on low lying impervious surfaces and two in high impervious surfaces, which could be interpreted as rooftops. On the other hand, a majority of the standard filters seem to be relatively less structured and respond to some combination of color and height. We can also see a few instances of edge detectors that have been learned in different orientations. Note that the

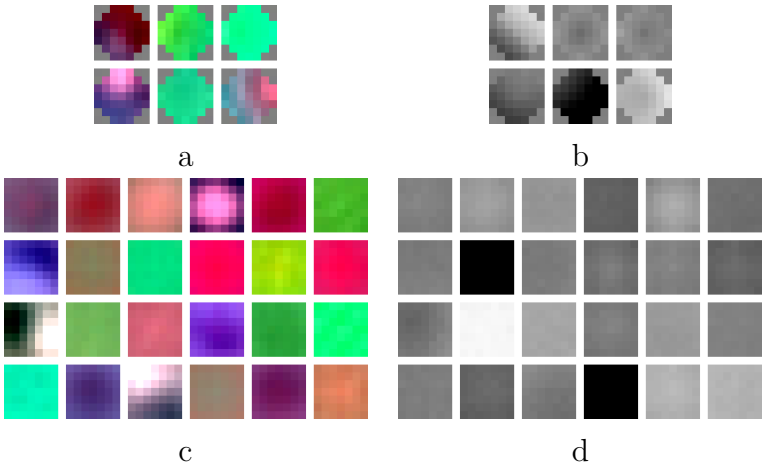


Figure 5.7: Visualization of all the filters learned on the first layer of the RotEqNet model on Vaihingen, a) on the optical channels, b) on the nDSM, and of the Standard CNN model, c) on the optical channels, d) on the nDSM. The filters are not normalized to appreciate the relative importance of each channel.

particular orientation of the RotEqNet filters is arbitrary and any other rotated version could have been learned as the canonical filter.

RotEqNet does not need to learn filters that are rotated versions of each other because all of these versions are explored by applying each filter at many different orientations. This means that, while standard CNNs require data augmentation to perform well in a rotation equivariant setting, RotEqNet extracts features at different orientations and keeps the largest activations, effectively analyzing the input at different orientations without rotating it explicitly.

Fig. 5.8 shows a few examples of the obtained classification maps. We see how RotEqNet performs better on smaller objects, such as cars or the grass path in the second image, but generates less smooth edges. The latter is possibly due to different orientation for certain features being chosen in contiguous pixels.

5.4.2 Zeebruges

The results in Table 5.2 show the performance of the proposed method on the Zeebruges dataset compared to the last published results in (Campos-Taberner et al., 2016). Although the authors of (Campos-Taberner et al., 2016) were not aiming at obtaining lightweight models, the two best results they report are obtained by CNNs containing of the order of 10^7 parameters, while RotEqNet achieves comparable results with a mere 10^5 parameters, two orders of magnitude less. In particular, out the models used by (Campos-Taberner et al., 2016), the VGG/SVM model consists of a linear SVM classifier trained on features extracted by a VGG network with $2.5 \cdot 10^7$ parameters, while the AlexNet model

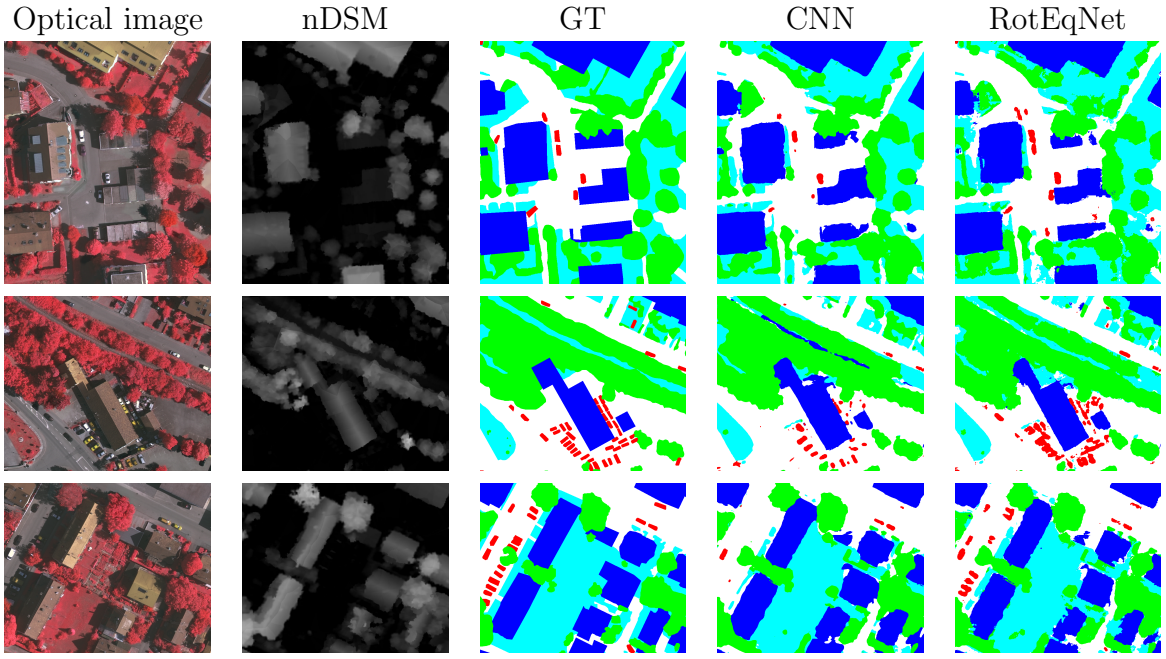


Figure 5.8: Examples of classification maps obtained in the Vaihingen validation images with the RotEqNet and the standard CNN models.

is a pretrained CNN that has been fine tuned end-to-end on the benchmark's training set. It has around $6 \cdot 10^7$ parameters. A RotEqNet network with $1.4 \cdot 10^5$ parameters, with Nf=4, was enough to obtain better results than the VGG/SVM model, and one with $4.3 \cdot 10^5$ parameters, Nf=7, was enough to close the accuracy gap with the fine tuned AlexNet. These results highlight the advantage in terms of model size reduction that can be obtained by sparing the model from having to learn to be equivariant to rotations. In this dataset we see again that RotEqNet performs particularly well on the car and the building classes, both associated with strong edge features, while it lags behind with respect to both competing models in the tree class, which contains rather anisotropic features.

5.4.3 Computational time

On the one hand, due to the additional burden of requiring to interpolate the rotated filters and the linear dependency between the number of orientations R and the number of convolutions to compute, RotEqNet can potentially increase the computational time required with respect to a standard CNN. On the other hand, the reduction in the number of feature maps, which is independent of R, can compensate for this if R is small enough. As we can see in Table 5.3, the RotEqNet model tested on the Vaihingen validation set and trained with $R = 16$ outperforms the standard CNN in terms of speed up to $R = 64$. Note that all tests are performed on a single CPU to make the results more comparable.

Table 5.2: Results on Zeebruges. F1 scores per class and global average (AA) and overall accuracies (OA) and Cohen's Kappa. Best result per row is in dark gray, second in light gray.

Model	RotEqNet			VGG/SVM*	AlexNet*
# parameters	$1.4 \cdot 10^5$	$2.2 \cdot 10^5$	$4.3 \cdot 10^5$	$2.5 \cdot 10^7$	$6 \cdot 10^7$
Impervious	74.75	74.98	77.41	67.66	**79.10**
Water	98.46	98.56	**98.69**	96.50	98.20
Clutter	31.19	36.27	48.99	45.60	**63.40**
Low Vegetation	76.58	77.89	**78.73**	68.38	78.00
Building	69.26	75.11	**79.07**	72.70	75.60
Tree	59.35	68.95	71.30	78.77	**79.50**
Boat	39.13	43.59	44.55	**56.10**	44.80
Car	56.26	**56.61**	52.54	33.90	50.80
OA	79.2	80.8	82.6	76.6	**83.32**
AA	69.2	73.2	75.3	-	-
Kappa	0.73	0.75	0.77	0.70	**0.78**

* = from (Campos-Taberner et al., 2016)

Table 5.3: Computational time of the forward pass in a single CPU and accuracy on the validation set of the Vaihingen dataset. The RotEqNet models are tested with different values of the number of orientations, R.

Model	RotEqNet					CNN
R	8	16	32	64	128	-
OA	86.90	87.89	87.81	87.71	87.65	87.47
AA	80.69	84.34	85.18	85.33	85.51	78.18
Kappa	0.82	0.84	0.84	0.83	0.83	0.83
Time per tile (s)	1.4	1.7	2.3	3.1	5.0	4.2

5.5 Conclusion

Deep learning models, and in particular convolutional neural networks, have shown their potential for remote sensing image analysis. By learning filters directly from data, they allow to learn and encode spatial information without engineering the feature space in a problem-dependent way. But if these models have potential, they still suffer from the need for an extensive (and comprehensive) training set, cumbersome hyperparameter tunning and considerable computing resources, both in terms of memory and operations. In this paper, we have explored the possibility of reducing such requirements by encoding one prior information about the images: the fact that their orientation, as well as that of objects it contains, is often arbitrary. Such prior can be exploited by making the CNN

model rotation equivariant, i.e. by forcing the network to react in the same way each time it encountered the same semantic class, independently from the spatial orientation of the features. We achieved this behavior by applying rotating convolutions, where a canonical filter is applied at many orientations and the maximal activation is propagated through the CNN. The proposed RotEqNet therefore has minimal memory and storage requirements, since it does not need to learn filters which respond to each specific orientation and thus generates less intermediate feature maps at runtime. Rotation equivariance is encoded within the model itself (similarly to how CNNs achieve translation equivariance) and propagating only maximal activations reduces the model size and runtime memory requirements while keeping most of the orientation information.

We applied the proposed framework to two subdecimeter land cover semantic labeling benchmarks. The results show two main tendencies: on one hand, that explicitly encoding rotation equivariance in deep learning dense semantic labeling models allows for much smaller models, between one and two orders of magnitude compared to traditional CNNs. On the other hand, they also show that a CNN encoding equivariance in its structure – rather than through data augmentation – also provides robustness against varying amounts of training data, allowing to train efficiently and perform well in modern remote sensing tasks. This last point is of particular importance when considering that the amount of available labels can vary enormously in remote sensing depending on the mode of acquisition and the problem at hand.

RotEqNet is not limited to semantic labeling tasks. Its logic can be applied to any deep model involving convolutions where a predefined behavior with respect to rotations is expected. As shown in chapter 4, it can be applied to various applications requiring rotation invariance, equivariance or even covariance, which opens doors for the application of RotEqNet to tackle problems of detection (cars, airplanes, trees) or regression (super-resolution, biophysical parameters) when only limited labeled instances are at hand.

Chapter 6

Learning deep structured active contours end-to-end

This chapter is based on:

Marcos, D., Tuia, D., Kellenberger, B., Zhang, L., Bai, M., Liao, R., and Urtasun, R. (2018b). Learning deep structured active contours end-to-end. In *Proceedings of the CVF/IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Abstract

The world is covered with millions of buildings, and precisely knowing each instance's position and extents is vital to a multitude of applications. Recently, automated building footprint segmentation models have shown superior detection accuracy thanks to the usage of Convolutional Neural Networks (CNN). However, even the latest evolutions struggle to precisely delineating borders, which often leads to geometric distortions and inadvertent fusion of adjacent building instances. We propose to overcome this issue by exploiting the distinct geometric properties of buildings. To this end, we present Deep Structured Active Contours (Deep Structured Active Contour (DSAC)), a novel framework that integrates priors and constraints into the segmentation process, such as continuous boundaries, smooth edges, and sharp corners. To do so, DSAC employs Active Contour Models (ACM), a family of constraint- and prior-based polygonal models. We learn ACM parameterizations per instance using a CNN, and show how to incorporate all components in a structured output model, making DSAC trainable end-to-end. We evaluate DSAC on three challenging building instance segmentation datasets, where it compares favorably against state-of-the-art.

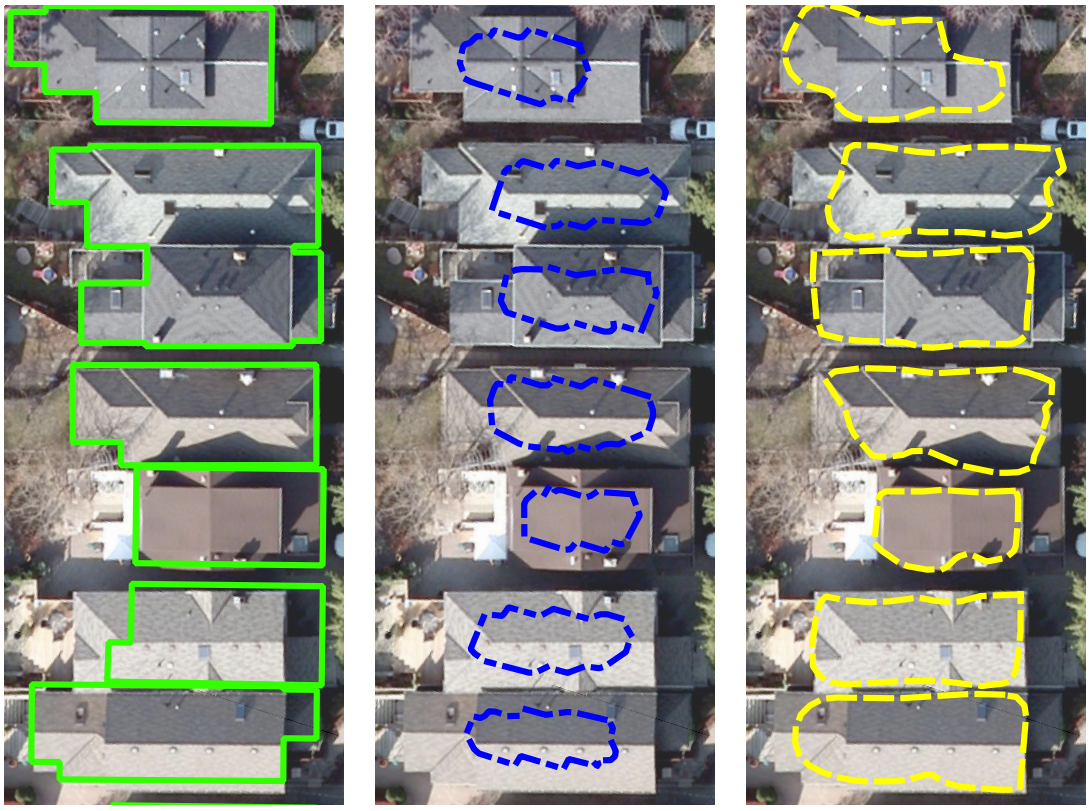


Figure 6.1: DSAC uses a CNN to predict the energy function used by an Active Contour Model (ACM) to modify an initial instance polygon using learned geometric priors. Left: image from the TorontoCity validation dataset with ground truth polygons, center: initial polygons provided by (Bai and Urtasun, 2017), right: results of DSAC.

6.1 Introduction

Accurate footprints of individual buildings are of paramount importance for a wide range of applications, such as census studies (Xie et al., 2015), disaster response after earthquakes (Sahar et al., 2010) and developmental assistances like malaria control (Franke et al., 2015). Automating large-scale building footprint segmentation has thus been an active research field, and the emergence of high-capacity models like fully convolutional networks (Fully Convolutional Network (FCN)s) (Gupta et al., 2014), together with vast training data (Wang et al., 2016), has led to promising improvements in this field.

Most studies address semantic segmentation of buildings, which consists of inferring a class label (e.g. "building") densely for each pixel over the overhead image of interest (Kaiser et al., 2017; Maggiori et al., 2017; Montoya-Zegarra et al., 2015; Volpi and Tuia, 2017). While this approach may provide global statistics such as building area coverage estimation, it comes short at yielding estimations at the instance level. In computer vision, this

problem is known as *instance segmentation*, where models provide a segmentation mask on a per-object instance basis. Solving this task is far more challenging than semantic segmentation, since the model has to understand whether any two building pixels belong to the same building or not. Precise delineation of object borders, with sharp corners and straight walls in the case of buildings, is a task that CNNs generally perform poorly at (Dai et al., 2016b): as a result, building segmentations from CNNs commonly have a high detection rate, but fail in terms of spatial coverage and geometric correctness.

Active Contour Models (ACM (Kass et al., 1988)), also called *snakes*, may be considered to address this issue. ACMs augment bottom-up boundary detectors with high-level geometric constraints and priors. They work by constraining the possible outputs to a family of curves (e.g. closed polygons with a fixed number of vertices), and optimizing them by means of energy minimization based on both the image features and a set of shape priors such as boundary continuity and smoothness. Additional terms have been proposed, among which the balloon term (Cohen, 1991) is of particular interest: it mimics the inflation of a balloon by continuously pushing the snakes' vertices outwards, thus preventing it to collapse to a single point. By expressing object detection as a polygon fitting problem with prior knowledge, ACMs have the potential of approaching object edges precisely and without the need for additional post-processing. However, the original formulation lacked flexibility, since it relied on low-level image features and a global parameterization of priors, when a more useful approach would be to penalize strongly the curvature in the regions of the boundary known to be straight or smooth and reduce the penalization in the regions that are more likely to form a corner. Moreover, the balloon term has so far only been included as a post-energy global minimization force and does not take part in the energy minimization defining the snake.

In this paper, we propose to combine the expressiveness of deep CNNs with the versatility of ACMs in a unified framework, which we term Deep Structured Active Contours (DSAC). In essence, we employ a CNN to learn the energy function that would allow an ACM to generate polygons close to a set of ground truth instances. To do so, DSAC leverages the original ACM formulation by learning high-level features and prior parameterizations, including the balloon term, in one model and *on a local basis, i.e.* penalizing each term differently at each image location. We cast the optimization of the ACM as a structured prediction problem and find suitable features and parameters using a Structured Support Vector Machine (Structured Support Vector Machine (SSVM) (Altun et al., 2007; Tsochantaridis et al., 2005)) loss. As a consequence, DSAC is trainable end-to-end and able to learn and adapt to a particular family of object instances. We test DSAC in three building instance segmentation datasets, where it outperforms state-of-the-art models.

Contributions This work's contributions are as follows:

- We formulate the learning of the energy function of an ACM as a structured prediction problem;
- We include the balloon term of the ACM into the energy formulation;
- We propose an end-to-end framework to learn the guiding features and local priors with a CNN.

6.2 Related work

Building footprint extraction Most current automated approaches make use of 3D information extracted from ground or aerial LIDAR (Wang et al., 2006), or employ humans in the loop (Brooks et al., 2015). The use of a polygonal shape prior has been shown to substantially improve the results (Sun et al., 2014) of systems based on color imagery and low level features. Recent efforts employ deep CNNs for semantic segmentation and allowed a great leap towards full automation of building segmentation (Kaiser et al., 2017). Works considering building instance segmentation are scarcer and the task has been recently defined as far-from-being solved (Wang et al., 2016), despite the interest shown by the participation to numerous contests aiming at automatic vectorization of building footprints from overhead imagery: SpaceNet[1], DSTL[2] or OpenAI Challenge[3]. Our proposed DSAC aims at making high-level geometric information available to CNN based methods as a step towards bridging this gap.

Instance segmentation in Computer Vision Since instance segmentation combines object detection and dense segmentation, many proposed pipelines attempt at fusing both tasks in either separate or end-to-end trainable models. For example, (Dai et al., 2016a) employ a multi-task CNN to detect candidate objects and infer segmentation masks and class labels per detection. (Fathi et al., 2017) train a CNN on pairs of locations and predicts the likelihood for the pair to belong to the same object. (Romera-Paredes and Torr, 2016) apply an attention-based RNN sequentially on deep image features to trace object instances in propagation order. (Bai and Urtasun, 2017) refine an existing semantic segmentation map by predicting a distance transform to the nearest boundary. High level relationships are accounted for in (Royer et al., 2016; Zhang et al., 2016b) by means of an instance MRF applied to the CNN's output.

All these methods employ pixel-wise CNNs and are thus not apt to integrating output shape priors directly, as polygonal output models would be. Only a few works deal with CNNs that explicitly produce a polygonal output. In (Castrejon et al., 2017), a

[1]https://wwwtc.wpengine.com/spacenet
[2]https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection
[3]https://werobotics.org/blog/2018/01/10/open-ai-challenge/

recursive neural network is used to generate a segmentation polygon node by node, while in (Rupprecht et al., 2016) a CNN predicts the direction of the nearest object boundary for each node in a polygon and uses it as a data term in an ACM. However, the first model is tailored towards a different problem (interactive segmentation and correction) and does not allow the inclusion of strong priors, and the second decouples the CNN training from ACM inference, thus lacking the end-to-end training capabilities of the proposed DSAC.

Active contours The first ACMs were introduced by Kass *et al.* in 1988 under the name of snakes (Kass et al., 1988). Variants of this original try to overcome some of its limitations, such as the need for precise initializations, or the dependence on user interaction. In (Gunn and Nixon, 1997) the authors propose to use two coupled snakes that better capture the information in the image. The above mentioned balloon force was introduced by (Cohen, 1991).

Although some modifications (Kichenassamy et al., 1995) have been proposed to improve the data term of the original paper, they rely on simple assumptions about the appearance of the objects and on global parameters for weighting the different terms in the energy function. The proposed DSAC leverages the original formulation by including local prior information, i.e. values weighting the snakes' energy function terms on a per-pixel basis, and learns them using a CNN. Although this work focuses on curvature priors useful for segmenting objects of polygonal shape, other priors can be enforced with ACMs, such as convexity for biomedical imaging (Royer et al., 2016).

Structured learning with CNNs Structured prediction (Taskar et al., 2005) allows to model dependencies between multiple output variables and hence offers an elegant way to incorporate prior rule sets on output configurations. End-to-end trainable structured models exceed traditional two-step solutions by enriching the learning signal with relations at the output level. Although these models have been applied to a variety of problems (Belanger and McCallum, 2016; Chen et al., 2015; Schwing and Urtasun, 2015), we are not aware of any work dealing with instance level segmentation.

We use a structured loss as a learning signal to a CNN such that it learns to coordinate the different ACM energy terms, which are heavily interdependent.

6.3 Method

We present the details of a modified ACM inference algorithm with image-dependent and local penalization terms as well as the structured loss that is used to train a CNN

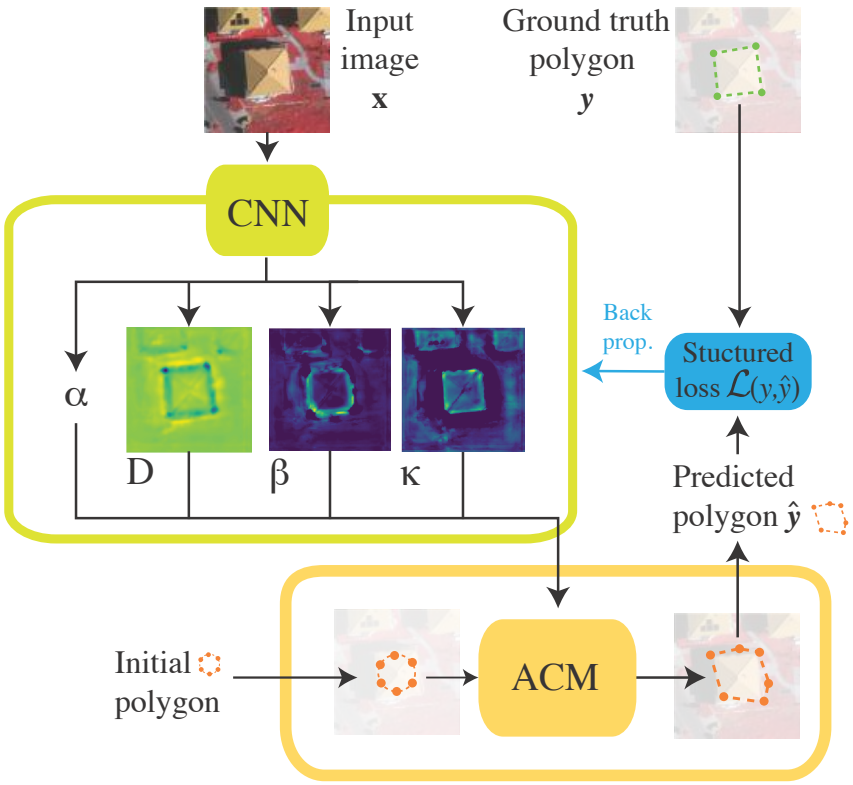


Figure 6.2: DSAC idea. The CNN predicts the values of the energy terms to be used by the active contour model (ACM): a global α for the length penalization and maps for local D, the data term, β, the curvature penalization and κ, the balloon term. After ACM inference, a structured loss is computed and given to the CNN, whose parameters can then be updated using backpropagation.

to generate these penalization maps. A diagram of the proposed method is shown in Figure 6.2. The proposed training algorithm proceeds as exposed in Algorithm 1.

Note that i) DSAC does not depend on any particular ACM inference algorithm, and ii) the chosen ACM algorithm does not need to be differentiable.

6.3.1 Locally penalized active contours

An active contour (Kass et al., 1988) can be represented as a polygon $\mathbf{y} = (\mathbf{u}, \mathbf{v})$ with L nodes $\mathbf{y}_s = (u_s, v_s) \in \mathbb{R}^2$, with $s \in \{1, \dots, L\}$, where each s represents one of the nodes of the discretized contour. The polygon $\mathbf{y}$ is then deformed such that the following energy function is minimized:

Data: $\mathcal{X}, \mathcal{Y}$: image/polygon pairs in the training set.
$\mathcal{Y}^0$: corresponding polygon initializations.
for $\mathbf{x}_i, \mathbf{y}_i \in \mathcal{X}, \mathcal{Y}$ **do**
  CNN inference: D, α, β, $\kappa \leftarrow CNN_\omega(\mathbf{x}_i)$
  ACM inference: $\hat{\mathbf{y}}_i \leftarrow ACM(D, \alpha, \beta, \kappa, \mathbf{y}_i^0)$
  $\frac{\partial \mathcal{L}}{\partial D}$, $\frac{\partial \mathcal{L}}{\partial \alpha}$, $\frac{\partial \mathcal{L}}{\partial \beta}$, $\frac{\partial \mathcal{L}}{\partial \kappa} \leftarrow \hat{\mathbf{y}}^i, \mathbf{y}_i$ and Eqs. 6.18-6.19
  Compute $\frac{\partial \mathcal{L}}{\partial \omega}$ using backpropagation
  Update CNN: $\omega \leftarrow \omega - \eta \frac{\partial \mathcal{L}}{\partial \omega}$
end

Algorithm 1: The DSAC training algorithm. At every iteration, the CNN forward pass is followed by ACM inference, which yields a contour that is used to compute the structured loss.

$$E(\mathbf{y}, \mathbf{x}) = \sum_{s=1}^{L} \left[D\big(\mathbf{x}, (\mathbf{y}_s)\big) + \alpha\big(\mathbf{x}, (\mathbf{y}_s)\big) \left| \frac{\partial \mathbf{y}}{\partial s} \right|^2 + \beta\big(\mathbf{x}, (\mathbf{y}_s)\big) \left| \frac{\partial^2 \mathbf{y}}{\partial s^2} \right|^2 \right] + \sum_{u,v \in \Omega(\mathbf{y})} \kappa(\mathbf{x}, (u, v)), \tag{6.1}$$

where $D\big(\mathbf{x}\big) \in \mathbb{R}^{M \times N}$ is the data term, depending on input image, of size $M \times N$, $\mathbf{x} \in \mathbb{R}^{M \times N \times d}$, $\alpha\big(\mathbf{x}\big), \beta\big(\mathbf{x}\big) \in \mathbb{R}^{M \times N}$ are the terms encouraging short and smooth polygons respectively, $\kappa(\mathbf{x})$ is the balloon term and $\Omega(\mathbf{y})$ is the region enclosed by $\mathbf{y}$. The notation $D\big(\mathbf{x}, (\mathbf{y}_s)\big)$ means the value in $D\big(\mathbf{x}\big)$ indexed by the position $\mathbf{y}_s = (u_s, v_s)$.

Due to their local nature, D,β and κ are $M \times N$ maps in our experiments while α is treated as a single scalar.

Data term

This term identifies areas of the image where the nodes of the polygon should lie. In the literature, $D\big(\mathbf{x}\big)$ is usually some predefined function on the image, typically related to the image gradients. $D(\mathbf{x})$ should learn to provide relatively low values along the boundary of the object of interest and high values elsewhere. During ACM inference, the direction of steepest descent $-\nabla D(\mathbf{x}) = -\big[\frac{\partial D(\mathbf{x})}{\partial u}, \frac{\partial D(\mathbf{x})}{\partial v}\big]$ is used as the data force term, moving the contour towards regions where D is low.

Internal terms

In the literature, the values of α and β are generally a single scalar, meaning that the penalization has the same strength in all parts of the object. This leads to a trade-off

between over-smoothing corner regions and under-smoothing others. We avoid this trade-off by assigning different β penalizations to each pixel, depending on which part of the object lies underneath.

The internal energy $E_{int} = \alpha\big(\mathbf{x}, (\mathbf{y}_s)\big)|\mathbf{y}'|^2 + \beta\big(\mathbf{x}, (\mathbf{y}_s)\big)|\mathbf{y}''|^2$ penalizes the length (membrane term) and curvature (thin plate term) of the polygon. In order to obtain the direction of steepest descent, we can express the internal energy as a function of finite differences:

$$E_{int} = \sum_{s=0}^{L} \alpha(\mathbf{y}_s)\Big|\frac{\mathbf{y}_{s+1} - \mathbf{y}_s}{\Delta s}\Big|^2 + \beta(\mathbf{y}_s)\Big|\frac{\mathbf{y}_{s+1} - 2\mathbf{y}_s + \mathbf{y}_{s-1}}{\Delta s^2}\Big|^2, \tag{6.2}$$

and compute the derivative of E_{int} w.r.t. the coordinates of node s, $\mathbf{y}_s$, expressed as a sum of scalar products:

$$\frac{\partial E_{int}}{\partial \mathbf{y}_s} = \frac{2}{\Delta s}[-\alpha_{s-1}, \alpha_{s-1} + \alpha_s, -\alpha_s] \cdot [\mathbf{y}_{s-1}, \mathbf{y}_s, \mathbf{y}_{s+1}]^\top + \frac{2}{\Delta s^2}[\beta_{s-1}, -2\beta_s - 2\beta_{s-1}, \beta_{s-1} + 4\beta_s + \beta_{s+1}, -2\beta_{s+1} - 2\beta_s, \beta_{s+1}] \cdot [\mathbf{y}_{s-2}, \mathbf{y}_{s-1}, \mathbf{y}_s, \mathbf{y}_{s+1}, \mathbf{y}_{s+2}]^\top. \tag{6.3}$$

The Jacobian matrix (in this case with two column vectors) can then be expressed as a matrix multiplication:

$$\frac{\partial E_{int}}{\partial \mathbf{y}} = (A + B)\mathbf{y} \tag{6.4}$$

where $A(\alpha)$ is a tri-diagonal matrix and $B(\beta)$ is a penta-diagonal matrix.

Balloon term

The original balloon term (Cohen, 1991) consists of adding an outwards force of constant magnitude in the normal direction of each node, thus inflating the contour. As with the β term, we propose to increase its flexibility by allowing it to take a different value at each image location.

In (Cohen, 1991), the balloon term is only considered as a force added after the direction of steepest descent for the other energy terms has been computed. In DSAC, the SSVM formulation requires to express it in the form an energy.

The normal direction to the contour at $\mathbf{y}_s$ follows the vector:

$$\mathbf{n}_s = \big[\mathbf{y}_{s+1} - \mathbf{y}_{s-1}\big]_{+90^o} = \big[v_{n+1} - v_{n-1}, u_{n-1} - u_{n+1}\big]. \tag{6.5}$$

This can be rewritten such that the whole set of L normal vectors is expressed as:

$$\mathbf{n} = \Big[C\mathbf{v}, \mathbf{u}^\top C\Big] \tag{6.6}$$

where C is a tri-diagonal matrix with 0 in the main diagonal, 1 in the upper diagonal and -1 in the lower diagonal.

Integrating this expression with respect to $\mathbf{u}$ and $\mathbf{v}$, we obtain the scalar E_b, corresponding to the polygon's area (by the shoelace formula to compute the area of a polygon):

$$E_b = \kappa[\mathbf{u}^\top C\mathbf{v}] = \sum_{u,v\in\Omega(\mathbf{y})} \kappa \tag{6.7}$$

Instead of maximizing the area of the polygon, which would be the result of pushing nodes in the normal direction, we propose to use a more flexible term that maximizes the integral of the values of a map $\kappa(\mathbf{x}) \in \mathbb{R}^{M\times N}$ over the area enclosed by the contour, $\Omega(\mathbf{y})$. If we discretize the integral to the pixel values that conform κ, we obtain:

$$E_k = \sum_{u,v\in\Omega(\mathbf{y})} \kappa(u,v) \tag{6.8}$$

After this modification we need to recompute the force form of this term by finding the $L \times 2$ Jacobian matrix $[\frac{\partial E_k}{\partial u_s}, \frac{\partial E_k}{\partial v_s}]$, $s \in [1, L]$.

This corresponds to how a perturbation in u_s and v_s would affect E_k. Since the perturbations are considered to be very small, we assume that the distribution of the $\kappa(u, v)$ values along the segments $[\mathbf{y}_s, \mathbf{y}_{s+1}]$ and $[\mathbf{y}_{s-1}, \mathbf{y}_s]$ will be identical to the one in $[\mathbf{y}_s + \Delta\mathbf{y}, \mathbf{y}_{s+1}]$ and $[\mathbf{y}_{s-1}, \mathbf{y}_s + \Delta\mathbf{y}]$, respectively. As shown in Figure 6.3, this boils down to summing a series of trapezoid areas, forming the two depicted triangles, each one weighted by its assigned κ value.

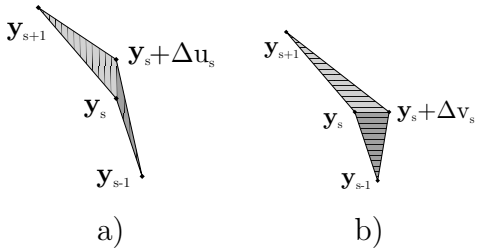


Figure 6.3: A perturbation of $\mathbf{y}_s$ in either the u or v direction would result in a change in area highlighted as two shaded triangles sharing the same base.

In Figure 6.3a, both triangles have bases of length Δu_s and heights $v_{s-1} - v_s$ and $v_{s+1} - v_s$, while in Figure 6.3b the bases are Δv_s and the heights $u_{s-1} - u_s$ and $u_{s+1} - u_s$.

To obtain the κ weighted areas in Figure 6.3a, we compute:

$$\Delta E_k = \frac{\Delta u_s}{v_{s-1} - v_s} \sum_{h=0}^{v_{s-1}-v_s} h\kappa(h)\Delta h + \frac{\Delta u_s}{v_{s+1} - v_s} \sum_{h=0}^{v_{s+1}-v_s} h\kappa(h)\Delta h, \tag{6.9}$$

and therefore the force term we need for inference is:

$$\frac{\partial E_k}{\partial u_s} = \frac{1}{v_{s-1} - v_s} \sum_{h=0}^{v_{s-1}-v_s} h\kappa(h)\Delta h + \frac{1}{v_{s+1} - v_s} \sum_{h=0}^{v_{s+1}-v_s} h\kappa(h)\Delta h \tag{6.10}$$

The same for Figure 6.3b can be obtained by swapping u and v.

These derivatives point in the normal direction when the values of κ are equal in all locations.

6.3.2 Active contour inference and implementation

When solving the ACM inference, Eq. (6.1), the four energy terms can be split into external terms E_{ext}: the data (D) and balloon energies (E_k); and internal terms E_{int}: the energies penalizing length (α) and curvature (β). Since E_{int} depends only on the contour $\mathbf{y}$, we can use the implicit Euler method, with $\mathbf{y}^{t+1}$ in both sides of the expression, to get an improved stability and better convergence:

$$\mathbf{y}^{t+1} = \mathbf{y}^t - \frac{dE_{ext}}{d\mathbf{y}^t} - (A + B)\mathbf{y}^{t+1}. \tag{6.11}$$

This comes at the price of having to compute a matrix inversion when solving for $\mathbf{y}^{t+1}$:

$$\mathbf{y}^{t+1} = (I + A + B)^{-1}\Big(\mathbf{y}^t - \frac{dE_{ext}}{d\mathbf{y}^t}\Big), \tag{6.12}$$

where I is the identity matrix. An efficient implementation of the ACM inference is critical for the usability of the method, since thousands of iterations are typically required by CNNs to be trained, and the ACM inference has to be performed at each iteration. We have implemented the described locally penalized ACM using a Tensorflow graph. The typical inference time is under 50 ms on a single CPU for the settings used in this paper.

6.3.3 Structured SVM loss

Since no ground truth is available for the penalization terms, we frame the problem as structured prediction, in which loss augmented inference is used to generate negative examples to complement the positive examples of the ground truth polygons. The weights of the energy terms can then be modified such that the energy corresponding to the ground truth is lowered, while the one of the loss augmented results, which are presumed to be wrong, is increased.

Given a collection of ground truth pairs $(\mathbf{y^i}, \mathbf{x^i}) \in \mathcal{Y} \times \mathcal{X}, i = 1 \dots N$, and a task loss function $\Delta(\mathbf{y}, \hat{\mathbf{y}})$, we would like to find the CNN parameters ω such that, by optimizing Eq. (6.1) and thus obtaining the inference result:

$$\hat{\mathbf{y}}^i = \arg\min_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}, \mathbf{x}^i, \omega) \tag{6.13}$$

one could expect a small $\Delta(\mathbf{y}^i, \hat{\mathbf{y}}^i)$. The problem becomes:

$$\hat{\omega} = \arg\min_{\omega} \sum_i \Delta(\mathbf{y}^i, \arg\min_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}, \mathbf{x}^i, \omega)) \tag{6.14}$$

Since $\Delta(\mathbf{y}^i, \hat{\mathbf{y}}^i)$ could be a discontinuous function, we can substitute it by a continuous and convex upper bound, such as the hinge loss. By adding an ℓ_2 regularization and summing for all training samples, this becomes the max-margin formulation:

$$\mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega) = \frac{1}{2}\|\omega\|^2 + \quad C \sum_i \left(\max_{\mathbf{y} \in \mathcal{Y}} \left[0, \Delta(\mathbf{y}, \mathbf{y}^i) - E(\mathbf{y}, \mathbf{x}^i; \omega) + E(\mathbf{y}^i, \mathbf{x}^i; \omega) \right] \right). \tag{6.15}$$

Since $\mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega)$ is convex but not differentiable, we compute the subgradient, which requires to find the most penalized constraint with the current ω:

$$\hat{\mathbf{y}}^i = \arg\max_{\mathbf{y} \in \mathcal{Y}} \left[\Delta(\mathbf{y}, \mathbf{y}^i) - E(\mathbf{y}, \mathbf{x}^i; \omega) \right] \tag{6.16}$$

This means to first run the ACM, by iterating over Eq. (6.12), using the current ω and an extra term corresponding to the task loss $\Delta(\mathbf{y}, \mathbf{y}^i)$. Once we obtain $\hat{\mathbf{y}}^i$, we can then compute the subgradient as:

$$\frac{\partial \mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega)}{\partial \omega} = \omega + C \sum_i \left(\frac{\partial E(\mathbf{y}^i, \mathbf{x}^i; \omega)}{\partial \omega} - \frac{\partial E(\hat{\mathbf{y}}^i, \mathbf{x}^i; \omega)}{\partial \omega} \right) \tag{6.17}$$

We compute the subgradients of the loss with respect to each of the four outputs as

$$\frac{\partial \mathcal{L}(\mathbf{y}^i, \mathbf{x}^i, \omega)}{\partial D_\omega(\mathbf{x}^i)} = [(u, v) \in \mathbf{y}^i] - [(u, v) \in \hat{\mathbf{y}}^i] \tag{6.18}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}^i, \mathbf{x}^i, \omega)}{\partial \alpha_\omega(\mathbf{x}^i)} = \left| \frac{\partial \mathbf{y}^i(u, v)}{\partial s} \right|^2 [(u, v) \in \mathbf{y}^i] - \left| \frac{\partial \hat{\mathbf{y}}^i(u, v)}{\partial s} \right|^2 [(u, v) \in \hat{\mathbf{y}}^i]$$

$$\frac{\partial \mathcal{L}(\mathbf{y}^i, \mathbf{x}^i, \omega)}{\partial \beta_\omega(\mathbf{x}^i)} = \left| \frac{\partial^2 \mathbf{y}^i(u, v)}{\partial s^2} \right|^2 [(u, v) \in \mathbf{y}^i] - \left| \frac{\partial^2 \hat{\mathbf{y}}^i(u, v)}{\partial s^2} \right|^2 [(u, v) \in \hat{\mathbf{y}}^i]$$

$$\frac{\partial \mathcal{L}(\mathbf{y}^i, \mathbf{x}^i, \omega)}{\partial \kappa_\omega(\mathbf{x}^i)} = [(u,v) \in \Omega(\mathbf{y}^i)] - [(u,v) \in \Omega(\hat{\mathbf{y}}^i)]. \tag{6.19}$$

In the above equations, $[\cdot]$ represents the Iverson bracket. Finally, we can get $\frac{\partial \mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega)}{\partial \omega}$ using the chain rule and modifying each CNN parameter ω applying:

$$\omega_{t+1} = \omega_t - \eta \frac{\partial \mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega)}{\partial \omega}, \tag{6.20}$$

which will simultaneously decrease $E(\mathbf{y}^i, \mathbf{x}^i; \omega)$ and increase $E(\hat{\mathbf{y}}^i, \mathbf{x}^i; \omega)$, thus making a better solution more likely when performing inference anew.

Task loss The task loss $\Delta(\mathbf{y}, \mathbf{y}^i)$ defines the actual objective we want to solve with the SSVM loss. Since it's the most common metric in instance segmentation, we employ the Intersection over Union (IoU) between the prediction $\mathbf{y}$ and the ground truth $\mathbf{y}^i$. Note that optimizing for IoU can be split into maximizing the intersection while minimizing the union. During training, this allows us to simply add a negative value during training to the κ map at the locations within the ground truth and a positive outside to obtain a loss-augmented inference (see Fig. 6.4).

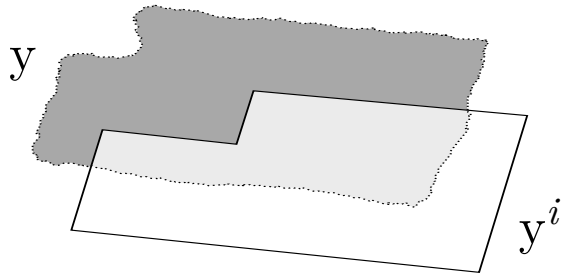


Figure 6.4: When training we encourage a high task loss (low IoU) by modifying the balloon term E_κ, adding a negative constant to κ at the nodes of the prediction $\mathbf{y}$ inside the ground truth $\mathbf{y^i}$ (light gray), and a positive constant to those outside (dark gray).

6.4 Experiments

We test the proposed DSAC method for building footprint extraction from overhead images. We consider two settings: *manual initialization*, where the user provides a single click near the center of the building and *automatic initialization*, where an instance segmentation algorithm is used to generate the initial polygons. The first setting is tested

in two datasets, *Vaihingen* and *Bing Huts*, while the second is tested in the *TorontoCity* dataset (Wang et al., 2016). The three datasets are detailed in the respective sections.

6.4.1 CNN architecture and general setup

To learn the ACM energy terms, we use a CNN architecture similar to the Hypercolumn model in (Hariharan et al., 2015). The input consists of a patch cropped around each initialization polygon and resized an image of fixed size for each dataset. The first layer consists of 7×7 convolutions, the second of 5×5 and all subsequent layers are of size 3×3. All the convolutional layers are followed by ReLU, batch normalization and 2×2 max-pooling. The number of filters is increased with the depth: 32, 64, 128 ,128, 256 and 256 for the six blocks. The output tensors of all the layers are then upsampled to the output size and concatenated. After this, a two-layer MLP with 256 and 64 hidden units is used to predict the four output maps: $D(\mathbf{x})$, $\alpha(\mathbf{x})$, $\beta(\mathbf{x})$ and $\kappa(\mathbf{x})$. We use this architecture for all datasets, with the exception of the *Bing huts* dataset, for which we skip the last two convolutional layers. In all cases, we use the Adam optimizer with a learning rate of 10^{-4}. We augment the data with random rotations. The number of ACM iterations is set to 50 in all the experiments, and the number of nodes is set to $L = 60$ in *Vaihingen* and *TorontoCity* and $L = 20$ in *Bing huts*.

6.4.2 Manual initialization

In this setting, the detection step is done manually by visual inspection. The only input required from the user is a single click to indicate the approximate center of the building. Two datasets are considered:

Vaihingen buildings The dataset consists of 168 buildings extracted from the training set of the ISPRS "2D semantic labeling contest"[4]. The images have three bands, corresponding to near infrared, red and green wavelengths, and a resolution of 9 cm. We used 100 buildings to train the models and the remaining 68 as a test set.

Bing huts The dataset consists of 605 individual huts visible on Bing maps aerial imagery at a resolution of 30 cm, over a rural area in Tanzania. See Figure 6.5 for an overview of the study area and Figure 6.7 for a full resolution subset. The ground truth building footprints have been obtained from OpenStreetMap[5]. A total of 335 images of size 80×80 pixels are used to train the models and the remaining 270 to test. The lower

[4] `http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html`
[5] `http://www.openstreetmap.org`

spatial resolution, low contrast between the buildings and the surrounding soil, as well as the high level of label noise make *Bing huts* a very challenging dataset.

We compare DSAC against a baseline where we train a CNN with the same architecture used by DSAC, but with a 3-class cross entropy loss with classes: building, building boundary, background. The boundary class is added to help the model focus on learning the shapes of the buildings. In this case, the click from the user is used to select the nearest connected region that has been labeled as building and treat it as the instance prediction.

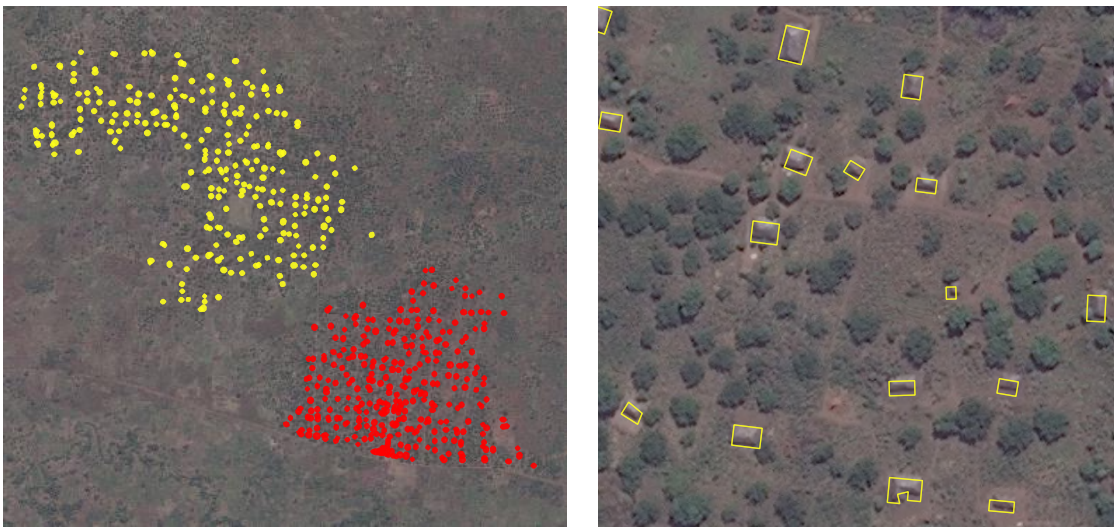


Figure 6.5: Left: Overview of the 4 km^2 area covered by the Bing huts dataset. The training instances are highlighted in red and the test ones in yellow. Right: detail of the test set.

6.4.3 Automatic initialization

Although the manual initialization only requires a single click from the user, it can still be a tedious task for large scale datasets. Existing instance segmentation algorithms, such as the recently proposed Deep Watershed Transform (Deep Watershed Transform (DWT)) (Bai and Urtasun, 2017), can be used instead to initialize the active contours. These methods have a good recall, but tend to undersegment the objects and to lose detail near to the boundaries. To compensate for this effect, the authors of (Bai and Urtasun, 2017) apply a morphology-based post-processing step. We test the possibility of initializing the ACM within DSAC with the results obtained by (Bai and Urtasun, 2017) on the TorontoCity building instance segmentation dataset (Wang et al., 2016), with around 28000 instances for training and 12000 for testing. The ACM contours are initialized with the output of the DWT (Bai and Urtasun, 2017), the current state-of-the-art in terms of IoU. Two initialization polygon types are considered: the raw DWT output and the post-processed versions used in (Wang et al., 2016). We also consider a

third variant, where the raw DWT is used at train time and the post-processed one for inference at test time: this variant is based on the intuition that making the problem harder at train time, in addition to using the loss augmentation, helps learning a better energy function.

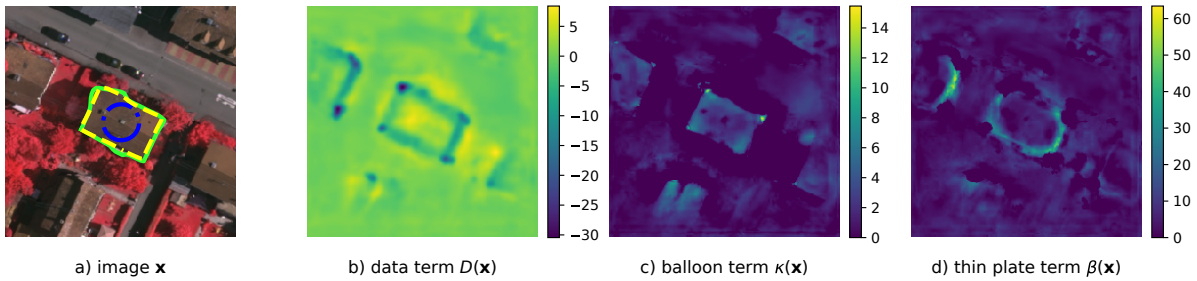


Figure 6.6: a) Image from the Vaihingen test set. The initial contour is in blue and the result in yellow, with the ground truth in green. b) Data term $D(\mathbf{x})$, where we can observe regions of lower energy along the boundary of the building. c) The balloon term $\kappa(\mathbf{x})$ has learned to produce positive values only inside the building, especially next to corners. d) In the thin plate term $\beta(\mathbf{x})$, we see that the curvature tends to be less penalized close to the building's corners. The membrane term provided by the model in this example was $\alpha(\mathbf{x}) = 0.74$

Figure 6.7: Examples of test set buildings in the Vaihingen (top row), Bing huts (middle row) and TorontoCity (bottom row) datasets. Ground truth in solid green line, baseline result in dash-dot blue and our active contour result in dashed yellow. Note that some of the ground truth polygons in the TorontoCity dataset are shifted (red arrows).

6.5 Results and discussion

6.5.1 Manual initialization

Table 6.1 reports the average IoU for the two datasets. Since the ground truth shift noise in the Bing huts dataset makes the IoU assessment untrustworthy, the root mean square error (Root Mean Square Error (RMSE) in m^2) committed when estimating the area of the building footprints is also reported. DSAC significantly improves the baseline in terms of IoU for both datasets. This ablation study confirms the need to allow κ and β to vary locally (as opposed to having a single value for the whole image), while α can be treated as a single value without loss of performance. It also highlights the importance of the balloon term for the convergence of the contour.

Table 6.1: Results on the test set for the manual initialization experiments, reported as average intersection over union (IoU, left) and area estimation (*Bing huts* only), with RMSE in m^2 (right).

	Average IoU		RMSE
	Vaihingen	Bing huts	Bing huts
CNN Baseline	0.78	0.56	23.9
DSAC (ours)	**0.84**	**0.65**	**13.4**
DSAC (scalar κ, β)	0.64	0.60	19.1
DSAC (no κ)	0.63	0.42	31.2
DSAC (local α)	0.83	**0.65**	**13.4**

Examples of segmentation results for the Vaihingen dataset (Figure 6.7, top row) show that the learned priors do indeed promote smooth, straight edges while often allowing for sharp corners. By looking at the predicted energy terms in Figure 6.6 we observe that the model focuses on the corners by producing very low D values close to them, while predicting high κ inside the building next to the corners and a sharp drop to 0 on the outside. Moreover, the smoothness term β is close to 0 at the corners and high along the edges.

In the Bing huts dataset results (Figure 6.7, bottom row), the biggest jump in performance can be seen in the area estimation metric. DSAC still tends to oversmooth the shapes, probably since it is unable to learn the location of corners due to the ground truth shift noise inherent to OpenStreetMap data, but manages to converge to polygons of the correct size, most probably because it learns to balance the balloon (κ, promoting large areas) and the membrane (α, promoting short contours) terms.

6.5.2 Automatic initialization

Table 6.2 reports the results obtained on the TorontoCity dataset using two metrics: the IoU-based weighted coverage ("WeighCov") and the shape similarity PolySim (Wang et al., 2016). Besides DWT, we also compare DSAC against the results of building footprint segmentation with FCN and ResNet, as reported by Wang et al. (2016). We observe an improvement with respect to DWT of both metrics. DSAC obtains the best weighted coverage scores irrespectively of the initialization strategy. Interestingly, the best results are obtained by the hybrid initialization using raw DWT at training time and post-processed DWT polygons at test time. This suggests that our intuition about making the model work harder at train time is correct and seems to complement the use of a task loss in the SSVM loss. Finally, segmentation examples are shown in the last row of Figure 6.7: DSAC (in yellow) consistently returns a more desirable segmentation with respect to DWT (in blue), closer to the ground truth polygon (in green). Although we can still see oversmoothing in our results, note how an important amount of shift noise is also present in some instances, making the DSAC result more plausible than the ground truth in a few cases (red arrows).

Table 6.2: Results of the proposed DSAC and the methods reported in (Wang et al., 2016) on the validation set of the TorontoCity dataset, containing over 12000 detected building instances. Two ACM initializations, `RW` ((Bai and Urtasun, 2017)) and `PP` ((Bai and Urtasun, 2017) post-processed), are compared.

	WeighCov	PolySim
FCN (Long et al., 2015)	0.46	**0.32**
ResNet (He et al., 2016)	0.40	0.29
DWT, raw (Bai and Urtasun, 2017) (`RW`)	0.42	0.20
DWT, postproc. (`PP`)	0.52	0.24
DSAC (init.: train `RW` / test `RW`)	0.55	0.26
DSAC (init.: train `PP` / test `PP`)	0.57	0.26
DSAC (init.: train `RW` / test `PP`)	**0.58**	0.27

6.6 Conclusion

We have shown the potential of embedding high-level geometric processes into a deep learning framework for the segmentation of object instances with strong shape priors, such as buildings in overhead images. The proposed Deep Structured Active Contours (DSAC) uses a CNN to predict the energy function parameters for an Active Contour Model (ACM) such as to make its output close to a ground truth set of polygonal footprints. The model is trained end-to-end by bringing the ACM inference into the CNN training schedule and using the ACM's output and the ground truth polygon to assess a structured loss that

can be used to update the CNN's parameters using back-propagation. DSAC opens up the possibility of using a large collection of energy terms encoding for different priors, since an adequate balance between them is learned automatically. The main limitation of our model is that the initialization is assumed to be given by some external method and is therefore not included in the learning process.

Results in three different datasets, which include a 10% relative improvement over the state-of-the-art on the *TorontoCity* dataset, show that combining the bottom-up feature extraction capabilities of CNNs with the high-level constraints provided by ACMs is a promising path for instance segmentation when strong geometric priors exist.

Chapter 7

Synthesis

7.1 Conclusions and outlook

This thesis has explored the use of domain specific prior knowledge in EO together with data driven methods. Focusing on overhead imagery, these priors are: 1) the geospatial prior, related to the knowledge about the approximate location on the Earth's surface of the observed objects and to 2) the overhead-view prior, the fact that we can assume to be viewing these objects from the top, which means that stronger geometric constraints can be applied than in the general imaging case. In all the study cases, the performances of the methods that make use of this prior knowledge is substantially improved. This highlights the importance of understanding which constraints naturally arise from the problem we are tackling and how to encode them, instead of directly applying methods developed for less or differently constrained settings.

7.1.1 The geospatial prior

The first explored constraint, presented in Chapter 3, is based on the geospatial nature of overhead images, which links observed pixels to approximate locations on the Earth's surface. Since certain temporal autocorrelation can be expected in the nature of objects at a given geographic location, this prior generates the assumption that images taken over the same area must be related, even if they are taken with different sensors and years apart. This was shown to be useful for change-aware, multi-modal registration, which in turn can be applied to transferring land-cover maps across images while taking into account the possibility of land-cover changes. In particular, pixels in different images of the same area are considered to be similar if they relate similarly to coarse regions within their image. Since these coarse regions have dimensions of the order of a hundred meters, they correspond to the same approximate geographical location in all the images of the same area, even in cases where registration errors of several meters exist. Therefore, the coarseness of the regions used for the similarity operation allows for robustness against the typical positioning errors of a few meters. The results in terms of multi-modal registration, change detection and multi-modal land-cover class transfer are substantially improved with respect to using other domain invariant features (such as SIFT, see Section 2.3.1).

The question that opens up after this work is whether a less constrained version of this prior, dropping the requirement of the images to be taken over the same geographical area, would also help understanding correspondences between image regions. The hypothesis is that the spatial arrangement of similar pixels can be specific to the class of the underlying objects. Therefore, if in two different images we observe similar patterns in terms of self-similarity, that may be a clue to be used to transfer knowledge such as class labels from one image to the other. This would indeed move from exploiting the geospatial prior to

the overhead-view prior, since the hard assumption about geographical location, where the same objects are observed in different images, is changed to the softer assumption that similar objects tend to organize spatially in comparable ways when seen from above.

7.1.2 The overhead-view prior

The rest of the thesis turned its attention towards the second prior, based on the knowledge that images are obtained from a overhead-view perspective. This constrained viewpoint results in the appearance of simpler symmetries than in other image domains, such as natural images. This overhead-view over the Earth's surface results in RS images being considered isotropic, with no dominant direction, for most purposes. On the one hand, this means that the absolute orientation of the image should not affect its interpretation, making the injection of rotation equivariance particularly suitable. On the other hand, we can expect a smaller variety of perceived shapes in the objects, since their orientation is typically constrained to a rotation around their vertical axis, reducing the diversity of possible appearances. This allows to represent object outlines as functions of simpler shapes.

Rotation equivariance

Inspired from the fact that absolute orientation in overhead imagery is arbitrary, Chapter 4 focused on the injection of rotation equivariance into CNN models, proposing a method named RotEqNet, allowing to control what kind of behavior is expected from the model output when the input image is subject to a change in orientation. This chapter showed that it is advantageous to use a convolution operator modified to provide equivariance to both translation and rotation when the task at hand has a known expected behavior with respect to a rotation of the input. Although other domains, including that of natural images, can also benefit from this constraints, Chapter 5 showed the advantages of injecting rotation equivariance into a CNN applied to land-cover mapping of aerial imagery. In particular, it resulted in a much more compact model, requiring one order of magnitude less parameters, without a drop in performance.

The main limitation of the RotEqNet method exposed in Chapter 4 is the assumption that activations that have angular responses differing by more than 90° have zero similarity. This might not be true when there are symmetries either in the filter or in the input's pattern. For instance, if the input is an almost perfectly homogeneous image, the maximally activated orientations are going to pick up on spurious signals and take arbitrary directions, exacerbating this noise. Therefore, an extension that takes such symmetries into account when computing the similarity between activations and filter vectors would be a natural next step.

Additionally, the encouraging results obtained in Marcos et al. (2018a) suggest that a simultaneous encoding of rotation and scale equivariance, in addition to translation, could be beneficial. In the case of RS imagery, this would allow to share the same features across modalities with different resolutions, potentially helping the performance in those modalities with the least training samples.

Shape priors

The overhead view reduces the diversity of viewpoints from which objects can be observed and the complexity of the occlusions that occur. Chapter 6 explored how these priors can be leveraged in the case of building footprint extraction by constraining the output to be a polygon and using a CNN to learn where to allow the formation of corners and where to prevent it. This resulted in a substantial improvement in the results on three building footprint extraction datasets of very different characteristics.

The results obtained in Chapter 6 show that restricting the feasible space of the output segmentation masks for building footprint extraction helps obtaining a substantially better performance. It would therefore be interesting to extend this concept to the multi-class segmentation case. One possible approach would be to combine the jigsaw puzzle solving proposed in Marcos et al. (2016d) with the structured learning approach of Chapter 6. Instead of keeping a large collection of pieces as in Marcos et al. (2016d), a more compact representation could be learned in a sparsifying CNN, which could simultaneously learn the pieces themselves and the compatibility between them.

7.2 Reflection

Prior knowledge on the problem at hand was the main resource for building traditional CV models before the popularization of DL. Although other data driven approaches had been widely used before, such as SVM, RF or boosting, part of the pipeline of CV models using them, such as feature extraction, was generally handcrafted. SIFT, for instance, was designed to be invariant to scale, rotation and absolute intensity while keeping information about shape based on relative intensity. In the case of complex objects, part-based models can look for part templates in an image, based on the similarity of their SIFT features, and check whether they relate to each other in a way that is compatible with our prior knowledge about the complex object. If we are instead looking for known simple geometries in an image, the approach could include edge detection, in case we know that there is often a strong gradient along the object boundaries, and some form of Hough transform, because we have an idea of the shape we are looking for but not a very precise one about its location or size. The use of MRFs, ACMs and other energy minimization

methods requires the careful design of an energy function that will provide satisfactory results when minimized.

In order to take the right design decisions it is important to understand not just the natural constraints of the problem and how to express them, but also the relationship between input and output at each step, since this allows the expert to plug prior knowledge into this relationship and make use of it. For instance, we might have a hypothesis on how a certain operator, such an edge detector, might react to the presence of the object's boundaries and base on that the design of the following step, which might use this information to improve the detection of objects. If the operator's nature is not easily interpretable, it will be more difficult for the expert to inject this prior knowledge about the problem.

With the adoption of the DL paradigm, there has been a trend of using the same models across a myriad of applications: the same architectures, the scaffoldings that contain the learnable parameters, and the same cost functions that measure the goodness of the current solution and drive the learning process that tries to improve them. It seemed like a GPU and some knowledge in DL is all that is required to surpass the field experts in public competitions, from galaxy identification to skin cancer detection. However, this paradigm has two important drawbacks: 1) it is limited by the data: without enough and good data, an off-the-shelf DL model will probably fail to generalize well to new data and 2) it is hard to understand how the intermediate feature representations, and therefore the output, react to a modification in the input (which is why DL models are often called "back boxes"). This hampers our ability to interpret the features and output of a DL model, preventing us from establishing the trustworthiness of the output and injecting prior knowledge.

Chapters 4 to 6 of this thesis deal with methods that make certain aspects of deep CNNs more interpretable and thus suitable to the injection of prior knowledge: Chapter 4 is about making the intermediate features react in a predictable way to rotations in the input image and Chapter 6 uses a CNN to generate an intermediate output that can be interpreted and thereafter used by another algorithm to produce the final result. This is part of a growing trend in the direction of DL models in which it is possible to disentangle the factors of variation in the input images (such as into number of objects, size, color, pose, *etc.*) (Chen et al., 2016b; Mathieu et al., 2016; Narayanaswamy et al., 2017), rather than simply providing a final result. Making the dimensions of some intermediate feature spaces interpretable by humans opens up the possibility of injecting prior knowledge into the problem. Hinton et al. (2000) proposed a framework for representing individual objects in an image as multidimensional vectors, or capsules, that are hierarchically related to each other forming a tree structure. Capsules in one layer follow a voting scheme in order to associate and build a higher level capsule in the next layer. The values in the capsules have been shown to become easily interpretable, with the addition that they can be analyzed at different levels of the object hierarchy (Sabour et al., 2017). The concept

of disentanglement has also been applied to the output space. For a user of a fine-grained classification model, with hundreds of very similar classes such as bird species, it might be easier to asses the trustworthiness of the model if the output consists, not just of the species' name, but also a list of easily identifiable attributes, such as beak shape, color, tail characteristics, *etc.*, that corresponds to that species (Lampert et al., 2014). Not only does this help judging the validity of the result, but also allows to make use of prior knowledge for tasks such as zero-shot learning (Akata et al., 2013), in which a never seen class can be identified if we hold some information about its attributes.

Another concept, closely linked to interpretability, is explainability. Sometimes used as a synonym of interpretability (Zhang et al., 2018), some authors (Miller, 2017; Samek et al., 2017) consider a ML model to be explainable if it can offer the user, along with the prediction, answers about why that prediction has been produced. For instance, it could generate a textual explanation describing the image features that are relevant to the decision (Hendricks et al., 2016) or allow to visualize what regions of an image trigger certain output. This lets the user judge whether the model is "looking" at the expected place or has only found some artifact in the data. If both the where and the why became simultaneously interpretable, this would also open the door for the users to learn how to solve the visual problem themselves (Mac Aodha et al., 2018).

Having a DL model fully built of interpretable elements and able to point at the location on the image causing each element to activate would not only make it more trustworthy and easy to debug, but also more adequate to the injection of prior knowledge at every level: at the geometrical composition level thanks to the localization; at the feature level; and at the output level. I believe that the CV community should strive for such DL models that 1) allow the experts to make use their knowledge and 2) inspire trust, even teach new things, to the final users.

References

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

Akata, Z., Perronnin, F., Harchaoui, Z., and Schmid, C. (2013). Label-embedding for attribute-based classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Altun, Y., Hofmann, T., and Tsochantaridis, I. (2007). Support vector learning for interdependent and structured output spaces. In Bakir, G., Hofmann, T., Schölkopf, B., Smola, A. J., and Vishwanathan, S., editors, *Predicting Structured Data*, pages 85–105. MIT press.

Arganda-Carreras, I., Turaga, S. C., Berger, D. R., Cireşan, D., Giusti, A., Gambardella, L. M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J. M., et al. (2015). Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9.

Audebert, N., Le Saux, B., and Lefèvre, S. (2016). Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Proceeding of the Asian Conference on Computer Vision (ACCV)*.

Audebert, N., Le Saux, B., and Lefèvrey, S. (2017). Fusion of heterogeneous data in convolutional networks for urban semantic labeling. In *Proceedings of the Joint Urban Remote Sensing Event (JURSE)*. IEEE.

Bai, M. and Urtasun, R. (2017). Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Beier, T., Andres, B., Köthe, U., and Hamprecht, F. A. (2016). An efficient fusion move algorithm for the minimum cost lifted multicut problem. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

Belanger, D. and McCallum, A. (2016). Structured prediction energy networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 983–992.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W.

(2010). A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Benedek, C. and Szirányi, T. (2009). Change detection in optical aerial images by a multilayer conditional mixed Markov model. *IEEE Transactions on Geoscience and Remote Sensing*, 47(10):3416–3430.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302.

Blaschke, T. (2010). Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(1):2–16.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the annual meeting of the association of computational linguistics*.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239.

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

Brooks, R., Nelson, T., Amolins, K., and Hall, G. B. (2015). Semi-automated building footprint extraction from orthophotos. *Geomatica*, 69(2):231–244.

Bruzzone, L. and Marconcini, M. (2010). Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):770–787.

Bruzzone, L. and Serpico, S. B. (1997). An iterative technique for the detection of land-cover transitions in multitemporal remote-sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 35(4):858–867.

Campos-Taberner, M., Romero-Soriano, A., Gatta, C., Camps-Valls, G., Lagrange, A., Saux, B. L., Beaupère, A., Boulch, A., Chan-Hon-Tong, A., Herbin, S., Randrianarivo, H., Ferecatu, M., Shimoni, M., Moser, G., and Tuia, D. (2016). Processing of extremely high resolution LiDAR and RGB data: Outcome of the 2015 IEEE GRSS Data Fusion Contest. Part A: 2D contest. *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, 9(12):5547–5559.

Cardona, A., Saalfeld, S., Preibisch, S., Schmid, B., Cheng, A., Pulokas, J., Tomancak, P., and Hartenstein, V. (2010). An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. *PLoS Biology*, 8(10):e1000502.

Castelluccio, M. P. G., Sansone, C., and Verdoliva, L. (2015). Land use classification in

remote sensing images by convolutional neural networks. *arXiv*, 1508:1–11.

Castrejon, L., Kundu, K., Urtasun, R., and Fidler, S. (2017). Annotating object instances with a polygon-RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen, H., Qi, X. J., Cheng, J. Z., and Heng, P. A. (2016a). Deep contextual networks for neuronal structure segmentation. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Chen, L.-C., Schwing, A., Yuille, A., and Urtasun, R. (2015). Learning deep structured models. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016b). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*.

Cheng, G., Zhou, P., and Han, J. (2016a). Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12):7405–7415.

Cheng, G., Zhou, P., and Han, J. (2016b). RIFD-CNN: Rotation-Invariant and Fisher Discriminative convolutional neural networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cireşan, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338.

Cohen, L. D. (1991). On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218.

Cohen, T. S. and Welling, M. (2016a). Group equivariant convolutional networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Cohen, T. S. and Welling, M. (2016b). Steerable CNNs. *arXiv preprint arXiv:1612.08498*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cula, O. G. and Dana, K. J. (2001). Compact representation of bidirectional texture functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dai, J., He, K., and Sun, J. (2016a). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dai, J., Li, Y., He, K., and Sun, J. (2016b). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007). Transferring naive bayes classifiers for text classification. In *Proceedings of the National Conference on Artificial Intelligence*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dalla Mura, M., Atli Benediktsson, J., Waske, B., and Bruzzone, L. (2010). Morphological attribute profiles for the analysis of very high resolution images. *IEEE Transactions on Geoscience and Remote Sensing*, 48(10):3747–3762.

Daume III, H. and Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, pages 101–126.

Deselaers, T. and Ferrari, V. (2010). Global and efficient self-similarity for object classification and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

DigitalGlobe (2000a). QuickBird datasheet. `http://global.digitalglobe.com/sites/default/files/QuickBird-DS-QB-Prod.pdf/`.

DigitalGlobe (2000b). WorldView 2 datasheet. `http://global.digitalglobe.com/sites/default/files/DG_WorldView2_DS_PROD.pdf/`.

Doe, J. (2014). Commercial satellite imaging market - global industry analysis, size, share, growth, trends, and forecast, 2013 - 2019. *Transparency Market Research*.

Dragomir Anguelov, P. S., Pang, H.-C., Koller, D., and Sebastian Thrun, J. D. (2005). The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Advances in Neural Information Processing Systems (NIPS)*.

Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., and Pal, C. (2016). The importance of skip connections in biomedical image segmentation. In *Proceeding of the Deep Learning and Data Labeling for Medical Applications Workshop (DLMIA)*, pages 179–187.

Duan, L., Tsang, I. W., Xu, D., and Chua, T.-S. (2009). Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Fakhry, A., Peng, H., and Ji, S. (2016). Deep models for brain em image segmentation: novel insights and improved performance. *Bioinformatics*, 32(15):2352–2358.

Fathi, A., Wojna, Z., Rathod, V., Wang, P., Song, H. O., Guadarrama, S., and Murphy, K. P. (2017). Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*.

Fauvel, M., Tarabalka, Y., Benediktsson, J., Chanussot, J., and Tilton, J. (2013). Advances in spectral-spatial classification of hyperspectral images. *Proceedings of the IEEE*, 101(3):652–675.

Franke, J., Gebreslasie, M., Bauwens, I., Deleu, J., and Siegert, F. (2015). Earth observation in support of malaria control and epidemiology: MALAREO monitoring approaches. *Geospatial health*, 10(1).

Ganin, Y. and Lempitsky, V. (2014). Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741.

Gens, R. and Domingos, P. M. (2014). Deep symmetry networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Gerke, M. (2015). Use of the stair vision library within the ISPRS 2D semantic labeling benchmark (Vaihingen). Technical report, ITC, Univ. of Twente.

Glocer, K., Eads, D., and Theiler, J. (2005). Online feature selection for pixel classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, Bonn, German.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU Press.

Gong, B., Grauman, K., and Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Book in preparation for MIT Press.

Gopalan, R., Li, R., and Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 999–1006. IEEE.

Gueguen, L. (2015). Classifying compound structures in satellite images: A compressed representation for fast queries. *IEEE Transactions on Geoscience and Remote Sensing*, 53(4):1803–1818.

Gueguen, L. and Hamid, R. (2015). Large-scale damage detection using satellite imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1321–132.

Gunn, S. R. and Nixon, M. S. (1997). A robust snake implementation; a dual active contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):63–68.

Gupta, S., Girshick, R., Arbeláez, P., and Malik, J. (2014). Learning rich features from RGB-D images for object detection and segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

Hansen, M. C., Potapov, P. V., Moore, R., Hancher, M., Turubanova, S., Tyukavina, A., Thau, D., Stehman, S., Goetz, S., Loveland, T., et al. (2013). High-resolution global maps of 21st-century forest cover change. *Science*, 342(6160):850–853.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Harvey, J. (2002). Estimating census district populations from satellite imagery: some approaches and limitations. *International Journal of Remote Sensing*, 23(10):2071–2095.

Harvey, N. R., Theiler, J., Brumby, S. P., Perkins, S., Szymanski, J. J., Bloch, J. J., Porter, R. B., Galassi, M., and Young, A. C. (2002). Comparison of GENIE and conventional supervised classifiers for multispectral image feature extraction. *IEEE Trans. Geosci. Remote Sens.*, 40(2):393–404.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hendricks, L. A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., and Darrell, T. (2016). Generating visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

Henriques, J. F. and Vedaldi, A. (2016). Warped convolutions: Efficient invariance to spatial transformations. *arXiv preprint arXiv:1609.04382*.

Hinton, G. E., Ghahramani, Z., and Teh, Y. W. (2000). Learning to parse images. In *Advances in Neural Information Processing Systems (NIPS)*.

Hu, G., Yang, Y., Yi, D., Kittler, J., Christmas, W., Li, S. Z., and Hospedales, T. (2015). When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks.

In *Advances in Neural Information Processing Systems (NIPS)*.

Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449.

Kaiser, P., Wegner, J. D., Lucchi, A., Jaggi, M., Hofmann, T., and Schindler, K. (2017). Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*.

Kampffmeyer, M., Salberg, A.-B., and Jenssen, R. (2016). Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

Kellenberger, B., Marcos, D., and Tuia, D. (2018). Detecting mammals in uav images: Best practices to address a substantially imbalanced dataset with deep learning. *Remote Sensing of Environment*, 216:139–153.

Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A., and Yezzi, A. (1995). Gradient flows and geometric active contour models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Kivinen, J. J. and Williams, C. K. (2011). Transformation equivariant boltzmann machines. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*. Springer.

Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimizedvia graph cuts? *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (2):147–159.

Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T.-W., and Sejnowski, T. J. (2003). Dictionary learning algorithms for sparse representation. *Neural Computation*, 15(2):349–396.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Kroon, D.-J. and Slump, C. H. (2009). MRI modality transformation in demon registration. In *Proc. ISBI*, pages 963–966.

Kumar, A., Saha, A., and Daume, H. (2010). Co-regularization based semi-supervised domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*.

Lagrange, A., Le Saux, B., Beaupere, A., Boulch, A., Chan-Hon-Tong, A., Herbin, S., Randrianarivo, H., and Ferecatu, M. (2015). Benchmarking classification of earth-observation data: From learning explicit features to convolutional networks. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 4173 – 417.

Lampert, C. H., Nickisch, H., and Harmeling, S. (2014). Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465.

Laparra, V., Marcos, D., Tuia, D., and Camps-Valls, G. (2015). Large-scale random features for kernel regression. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*.

Laptev, D., Savinov, N., Buhmann, J. M., and Pollefeys, M. (2016). TI-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Le Cun, Y., Matan, O., Boser, B., Denker, J. S., Henderson, D., Howard, R., Hubbard, W., Jacket, L., and Baird, H. (1990). Handwritten zip code recognition with multilayer networks. In *Proceeding of the IEEE International Conference on Pattern Recognition (ICPR)*, volume 2. IEEE.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

Leen, T. K. (1995). From data distributions to regularization in invariant learning. *Neural Computation*, 7(5):974–981.

Lei, Z., Fang, T., Huo, H., and Li, D. (2012). Rotation-invariant object detection of remotely sensed images based on texton forest and hough voting. *IEEE Transactions on Geoscience and Remote Sensing*, 50(4):1206–1217.

Lenc, K. and Vedaldi, A. (2015). Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Leung, T. and Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44.

Leventon, M. E., Grimson, W. E. L., and Faugeras, O. (2002). Statistical shape influence in geodesic active contours. In *IEEE EMBS International Summer School on Biomedical Imaging*.

Li, S. Z. (1994). Markov random field models in computer vision. In *Proceedings of the*

European Conference on Computer Vision (ECCV). Springer.

Liu, C., Yuen, J., and Torralba, A. (2011). SIFT flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994.

Liu, Y., Piramanayagam, S., Monteiro, S. T., and Saber, E. (2017). Dense semantic labeling of very-high-resolution aerial imagery and lidar with fully-convolutional neural networks and higher-order crfs. In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Workshops, Earthvision*. IEEE.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Lu, D., Li, G., and Moran, E. (2014). Current situation and needs of change detection techniques. *International Journal of Image and Data Fusion*, 5(1):13–38.

Mac Aodha, O., Su, S., Chen, Y., Perona, P., and Yue, Y. (2018). Teaching categories to human learners with visual explanations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Maggiolo, L., Marcos, D., Moser, G., and Tuia, D. (2018). Improving maps from CNNs trained with sparse, scribbled ground truths using fully connected CRFs. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*.

Maggiori, E., Tarabalka, Y., Charpiat, G., and Alliez, P. (2016). High-resolution aerial image labeling with convolutional neural networks. *arXiv preprint arXiv: 1611.01962*.

Maggiori, E., Tarabalka, Y., Charpiat, G., and Alliez, P. (2017). Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):645–657.

Mairal, J., Ponce, J., Sapiro, G., Zisserman, A., and Bach, F. R. (2009). Supervised dictionary learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Malek, S., Bazi, Y., Alajlan, N., AlHichri, H., and Melgani, F. (2014). Efficient framework for palm tree detection in UAV images. *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, 7(12):4692–4703.

Marcos, D., de Morsier, F., Matasci, G., Tuia, D., and Thiran, J.-P. (2014). Hierarchical sparse representation for dictionary-based classification of hyperspectral images. In *Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE.

Marcos, D., Hamid, R., and Tuia, D. (2016a). Geospatial correspondences for multimodal registration. In *Proceedings of the CVF/IEEE Conference on Computer Vision and*

Pattern Recognition (CVPR).

Marcos, D., Kellenberger, B., Lobry, S., and Tuia, D. (2018a). Scale equivariance in cnns with vector fields. In *FAIM/ICML Workshop on Towards learning with limited labels: Equivariance, Invariance, and Beyond*.

Marcos, D., Lasser, T., López, A., and Bourquard, A. (2016b). Compressed imaging by sparse random convolution. *Optics Express*, 24(2):1269–1290.

Marcos, D., Tuia, D., Kellenberger, B., Zhang, L., Bai, M., Liao, R., and Urtasun, R. (2018b). Learning deep structured active contours end-to-end. In *Proceedings of the CVF/IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Marcos, D., Volpi, M., Kellenberger, B., and Tuia, D. (2018c). Land cover mapping at very high resolution with rotation equivariant cnns: Towards small yet accurate models. *ISPRS Journal of Photogrammetry and Remote Sensing*.

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. (2017). Rotation equivariant vector field networks. In *Proceedings of the CVF/IEEE International Conference on Computer Vision (ICCV)*.

Marcos, D., Volpi, M., and Tuia, D. (2016c). Learning rotation invariant convolutional filters for texture classification. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*.

Marcos, D., Volpi, M., and Tuia, D. (2016d). Solving structured segmentation of aerial images as puzzles. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*.

Marcos-Gonzalez, D., Camps-Valls, G., and Tuia, D. (2015). Weakly supervised alignment of multisensor images. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*.

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., and Stilla, U. (2016). Semantic segmentation of aerial images with an ensemble of cnss. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:473–480.

Mathieu, M. F., Zhao, J. J., Zhao, J., Ramesh, A., Sprechmann, P., and LeCun, Y. (2016). Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems (NIPS)*.

Miller, T. (2017). Explanation in artificial intelligence: insights from the social sciences. *arXiv preprint arXiv:1706.07269*.

Mnih, V. (2013). *Machine learning for aerial image labeling*. University of Toronto (Canada).

Montoya-Zegarra, J., Leistner, C., Schindler, K., et al. (2013). Semantic tie points. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 377–384. IEEE.

Montoya-Zegarra, J. A., Wegner, J. D., Ladickỳ, L., and Schindler, K. (2015). Semantic segmentation of aerial images in urban areas with class-specific higher-order cliques. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):127.

Moser, G., Serpico, S. B., and Benediktsson, J. A. (2013). Land-cover mapping by Markov modeling of spatial-contextual information. *Proceedings of the IEEE*, 101(3):631–651.

Narayanaswamy, S., Paige, T. B., Van de Meent, J.-W., Desmaison, A., Goodman, N., Kohli, P., Wood, F., and Torr, P. (2017). Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems (NIPS)*.

Ngiam, J., Chen, Z., Chia, D., Koh, P. W., Le, Q. V., and Ng, A. Y. (2010). Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Ojala, T., Pietikäinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59.

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

Penatti, O., Nogueira, K., and dos Santos, J. A. (2015). Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Workshops, Earthvision*.

Pengra, B., Long, J., Dahal, D., Stehman, S. V., and Loveland, T. R. (2015). A global reference database from very high resolution commercial satellite data and methodology for application to landsat derived 30m continuous field tree cover data. *Remote Sensing of Environment*, 165:234–248.

Regniers, O., Bombrun, L., Lafon, V., and Germain, C. (2016). Supervised classification of very high resolution optical images using wavelet based textural features. *IEEE Transactions on Geoscience and Remote Sensing*, 54(6):3722–3735.

Romera-Paredes, B. and Torr, P. H. S. (2016). Recurrent instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer.

Rousson, M. and Paragios, N. (2008). Prior knowledge, level set representations & visual grouping. *International Journal of Computer Vision*, 76(3):231–243.

Royer, L. A., Richmond, D. L., Rother, C., Andres, B., and Kainmueller, D. (2016). Convexity shape constraints for image segmentation. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533.

Rupprecht, C., Huaroc, E., Baust, M., and Navab, N. (2016). Deep active contours. *arXiv preprint arXiv:1607.05074*.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NIPS)*.

Sahar, L., Muthukumar, S., and French, S. P. (2010). Using aerial imagery and GIS in automated building footprint extraction and shape recognition for earthquake risk assessment of urban inventories. *IEEE Transactions on Geoscience and Remote Sensing*, 48(9):3511–3520.

Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Schiller, J. and Voisard, A. (2004). *Location-based services*. Elsevier.

Schwing, A. G. and Urtasun, R. (2015). Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*.

Shechtman, E. and Irani, M. (2007). Matching local self-similarities across images and videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shekhovtsov, A., Kovtun, I., and Hlaváč, V. (2008). Efficient mrf deformation model for non-rigid image matching. *Computer Vision and Image Understanding*, 112(1):91–99.

Sherrah, J. (2016). Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. *arXiv preprint arXiv: 1606.02585*.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.

Sifre, L. and Mallat, S. (2013). Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Simard, P. Y., Steinkraus, D., Platt, J. C., et al. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*. Citeseer.

Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., and Moreno-Noguer, F. (2015). Discriminative learning of deep convolutional feature point descriptors. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sohn, K. and Lee, H. (2012). Learning invariant representations with local transformations. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Sun, X., Christoudias, C. M., and Fua, P. (2014). Free-shape polygonal object localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

Swisstopo (2018). Swiss Federal Office of Topography SWISSIMAGE FCIR. `http://www.swisstopo.admin.ch/internet/swisstopo/en/home/products/images/ortho/swissimage/SWISSIMAGE_FCIR.html`.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2006). A comparative study of energy minimization methods for Markov Random Fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. (2005). Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Tsochantaridis, I., Finley, T., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Tuia, D., Courty, N., and Flamary, R. (2015). Multiclass feature learning for hyperspectral image classification: sparse and hierarchical solutions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:272–285.

Tuia, D., Marcos, D., and Camps-Valls, G. (2016). Multi-temporal and multi-source remote sensing image classification by nonlinear relative normalization. *ISPRS Journal of Photogrammetry and Remote Sensing*, 120:1–12.

Tur, G. (2009). Co-adaptation: Adaptive co-training for semi-supervised learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Vakalopoulou, M., Karantzalos, K., Komodakis, N., and Paragios, N. (2015). Simultaneous registration and change detection in multitemporal, very high resolution remote sensing data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Vargas-Muñoz, J. E., Marcos, D., Lobry, S., dos Santos, J. A., Falcão, A. X., and Tuia, D. (2018). Correcting misaligned rural building annotations in open street map using convolutional neural networks evidence. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*.

Vedaldi, A. and Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`.

Vedaldi, A. and Lenc, K. (2015). MatConvNet – convolutional neural networks for MATLAB. In *Proceeding of the ACM International Conference on Multimedia*.

Volpi, M. and Ferrari, V. (2015a). Semantic segmentation of urban scenes by learning local class interactions. In *Proceedings of the IEEE Conference on Computer Vision*

and Pattern Recognition Workshops (CVPRW).

Volpi, M. and Ferrari, V. (2015b). Structured prediction for urban scene semantic segmentation with geographic context. In *Proceedings of the Joint Urban Remote Sensing Event (JURSE)*. IEEE.

Volpi, M. and Tuia, D. (2017). Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):881–893.

Wang, C. and Mahadevan, S. (2009). Manifold alignment without correspondence. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Wang, C. and Mahadevan, S. (2011). Heterogeneous domain adaptation using manifold alignment. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Wang, O., Lodha, S. K., and Helmbold, D. P. (2006). A bayesian approach to building footprint extraction from aerial lidar data. In *International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 192–199. IEEE.

Wang, S., Bai, M., Mattyus, G., Chu, H., Luo, W., Yang, B., Liang, J., Cheverie, J., Fidler, S., and Urtasun, R. (2016). TorontoCity: Seeing the world with a million eyes. *arXiv preprint arXiv:1612.00423*.

Winder, S. A. and Brown, M. (2007). Learning local image descriptors. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. (2016). Harmonic networks: Deep translation and rotation equivariance. *arXiv preprint arXiv:1612.04642*.

Xie, Y., Weng, A., and Weng, Q. (2015). Population estimation of urban residential communities using remotely sensed morphologic data. *IEEE Geoscience and Remote Sensing Letters*, 12(5):1111–1115.

Xing, D., Dai, W., Xue, G.-R., and Yu, Y. (2007). Bridged refinement for transfer learning. In *Knowledge Discovery in Databases: PKDD 2007*, pages 324–335. Springer.

Yang, C., Everitt, J. H., Du, Q., Luo, B., and Chanussot, J. (2013). Using high-resolution airborne and satellite imagery to assess crop growth and yield variability for precision agriculture. *Proceedings of the IEEE*, 101(3):582–592.

Yang, J., Jiang, Y.-G., Hauptmann, A. G., and Ngo, C.-W. (2007). Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM.

Zagoruyko, S. and Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016a). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, Q., Wu, Y. N., and Zhu, S.-C. (2018). Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, Z., Fidler, S., and Urtasun, R. (2016b). Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, Y., Ye, Q., Qiu, Q., and Jiao, J. (2017). Oriented response networks. *arXiv preprint arXiv:1701.01833*.

Acknowledgements

Although, according to the custom, my name is the only one to be read on the cover of this book, none of it would have been accomplished had I been on my own. Firstly, and indeed most importantly, my supervisor Devis Tuia has been the keystone holding together the unstructured results of my work and enthusiasm, allowing me to build a PhD out of them. From the very beginning, when he helped me organize my MSc thesis in Lausanne and then encouraged me to join him in Zurich, to the very end, in Wageningen. But he hasn't been the only one to contribute very substantially to the contents of this book. Michele Volpi and Benjamin Kellenberger deserve a special mention, not only for helping me out with conceptualizing, running experiments and writing, but also with making this thesis at all possible thanks to their 24/7 technical support. Many others have contributed, knowingly or not, to the ideas explored in this book. Raffay Hamid helped overcome my reluctance to send my first paper to CVPR and Raquel Urtasun helped me push my limits, both providing extremely valuable learning experiences. Discussions with Nikos Komodakis, who volunteered to evaluate my ongoing thesis, Gillian Milani, Gellért Máttyus, Renjie Liao, Lisa Zhang, Zeynep Akata, Taco Cohen, Sylvain Lobry, John Vargas, Shivangi Srivastava and many others have either found their way into this book or helped me tidy up my brain mess. And of course, on top of this, there's that handful of amazing human beings who make life worth it and provide the strength to face Mondays, in Switzerland, Toronto and Wageningen, but especially my family and my southern family in Montefrío.

About the author

Diego Marcos grew up in Pamplona, Spain, where he studied the first four years of a five year degree on Industrial Engineering, the last year of which he followed at the Technical University of Darmstadt, Germany. He then spent two years in Berlin, interning at the Fraunhofer Institute for Micro-technology, where he participated in the development of an elastic electrode designed to measure signals in the peripheral nervous system for the control of prosthetic limbs. The manual and time consuming nature of this project drove him to wonder how computer simulations could be used to help in such a process and moved to Lausanne, Switzerland, to follow a MSc on Computational Sciences and Engineering at the Swiss Technical Institute of Technology (EPFL). There he discovered the joys of signal processing, machine learning and remote sensing. In February 2015, to make the joy last, he started a PhD under the supervision of Prof. Devis Tuia at the University of Zurich, on a topic at the interface between remote sensing and computer vision. After three years of academic and alpine enjoyment, he followed his supervisor and transferred to Wageningen University, the Netherlands, where he wrapped up his PhD work, managed to produce the present document and plans to continue studying the injection of priors in Deep Learning models by making them more interpretable.

Peer-reviewed journal publications

Marcos, D., Volpi, M., Kellenberger, B., and Tuia, D. (2018c). Land cover mapping at very high resolution with rotation equivariant cnns: Towards small yet accurate models. *ISPRS Journal of Photogrammetry and Remote Sensing*

Marcos, D., Lasser, T., López, A., and Bourquard, A. (2016b). Compressed imaging by sparse random convolution. *Optics Express*, 24(2):1269–1290

Tuia, D., Marcos, D., and Camps-Valls, G. (2016). Multi-temporal and multi-source remote sensing image classification by nonlinear relative normalization. *ISPRS Journal*

of Photogrammetry and Remote Sensing, 120:1–12

Kellenberger, B., Marcos, D., and Tuia, D. (2018). Detecting mammals in uav images: Best practices to address a substantially imbalanced dataset with deep learning. *Remote Sensing of Environment*, 216:139–153

Main peer-reviewed conference publications

Marcos, D., Hamid, R., and Tuia, D. (2016a). Geospatial correspondences for multimodal registration. In *Proceedings of the CVF/IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. (2017). Rotation equivariant vector field networks. In *Proceedings of the CVF/IEEE International Conference on Computer Vision (ICCV)*

Marcos, D., Tuia, D., Kellenberger, B., Zhang, L., Bai, M., Liao, R., and Urtasun, R. (2018b). Learning deep structured active contours end-to-end. In *Proceedings of the CVF/IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Other peer-reviewed conference publications

Vargas-Muñoz, J. E., Marcos, D., Lobry, S., dos Santos, J. A., Falcão, A. X., and Tuia, D. (2018). Correcting misaligned rural building annotations in open street map using convolutional neural networks evidence. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*

Maggiolo, L., Marcos, D., Moser, G., and Tuia, D. (2018). Improving maps from CNNs trained with sparse, scribbled ground truths using fully connected CRFs. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*

Kellenberger, B., Marcos, D., and Tuia, D. (2018). Detecting mammals in uav images: Best practices to address a substantially imbalanced dataset with deep learning. *Remote Sensing of Environment*, 216:139–153

Marcos, D., Volpi, M., and Tuia, D. (2016c). Learning rotation invariant convolutional filters for texture classification. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*

Marcos, D., Kellenberger, B., Lobry, S., and Tuia, D. (2018a). Scale equivariance in cnns with vector fields. In *FAIM/ICML Workshop on Towards learning with limited labels:*

Marcos, D., Volpi, M., and Tuia, D. (2016d). Solving structured segmentation of aerial images as puzzles. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*

Marcos-Gonzalez, D., Camps-Valls, G., and Tuia, D. (2015). Weakly supervised alignment of multisensor images. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*

Laparra, V., Marcos, D., Tuia, D., and Camps-Valls, G. (2015). Large-scale random features for kernel regression. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*

Marcos, D., de Morsier, F., Matasci, G., Tuia, D., and Thiran, J.-P. (2014). Hierarchical sparse representation for dictionary-based classification of hyperspectral images. In *Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE

PE&RC Training and Education Statement

With the training and education activities listed below the PhD candidate has complied with the requirements set by the C.T. de Wit Graduate School for Production Ecology and Resource Conservation (PE&RC) which comprises of a minimum total of 32 ECTS (= 22 weeks of activities)

Review of literature (6 ECTS)

- Reduce, reuse and recycle: the 3 Rs of ground truth sustainability.

Writing of project proposal (4.5 ECTS)

- Reduce, reuse and recycle: the 3 Rs of ground truth sustainability.

Post-graduate courses (2.5 ECTS)

- Vision and sports summer school; Czech Tech. Univ. (2016)
- Training DART radiative transfer model; CESBIO, Toulouse (2016)

Laboratory training and working visits (4.5 ECTS)

- Investigating active contours for building footprint extraction; University of Toronto. (2017)

Invited review of (unpublished) journal manuscript (2 ECTS)

- IEEE TGRS: Image classification with convolutional networks (2016)
- IEEE TGRS: Image classification (2016)

Deficiency, Refresh, Brush-up courses (6 ECTS)

- - Retrieving geographic information; Uni Zürich (2016)

- High performance computing; Uni Zürich (2016)

Competence strengthening / skills courses (4.7 ECTS)

- PhD Seminar; Uni Zürich
- Project management; Uni Zürich
- Scientific writing; Uni Zürich
- How to review a scientific paper; WUR
- Career perspectives; WUR

PE&RC Annual meetings, seminars and the PE&RC weekend (1.8 ECTS)

- Department of Geography Graduate School Retreat (2015)
- Department of Geography Graduate School Retreat (2016)
- PE&RC PhD Weekend (2018)

Discussion groups / local seminars / other scientific meetings (4.5 ECTS)

- RSL colloquium seminar (2015-2017)

International symposia, workshops and conferences (11.2 ECTS)

- IGARSS; oral presentation (2015)
- CVPR; poster presentation (2016)
- ICPR; poster presentation (2016)
- ICCV; poster presentation (2017)

Lecturing / supervision of practicals / tutorials (2 ECTS)

- Exercises design (2015)
- Machine learning for geosciences (2016)

This research received funding from the Swiss National Science Foundation project PP00P2_150593.

Cover design by Dennis Hendriks / ProefschriftMaken