ETH Library

Active semantic segmentation on a time budget

Doctoral Thesis

Author(s):
Roig Noguera, Gemma

Publication date:
2014

Permanent link:
https://doi.org/10.3929/ethz-a-010401989

Rights / license:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

DISS. ETH NO. 22248

ACTIVE SEMANTIC SEGMENTATION ON A TIME BUDGET

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

GEMMA ROIG NOGUERA

Màster Universitari en Enginyeria de Xarxes i Telecomunicacions,
Universitat Ramon Llull

born on *24.02.1984*

citizen of Catalonia, Spain

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Bastian Leibe, co-examiner

2014

TO XAVIER

Abstract

Efficient algorithms for object recognition are crucial for the newly robotics and computer vision applications that demand real-time and on-line methods. Some examples are autonomous systems, navigating robots, autonomous driving. In this work, we focus on efficient semantic segmentation, which is the problem of labeling each pixel of an image with a semantic class.

Our aim is to speed-up all of the parts of the semantic segmentation pipeline. We also aim at delivering a labeling solution on a time budget, that can be decided *on-the-fly*. For this purpose, we analyze all the components of the semantic segmentation pipeline, and identify the computational bottleneck of each of them. The different components of the pipeline are over-segmenting the image with local regions, extracting features and classify the local regions, and the final inference of the image labeling with semantic classes. We focus on each of these steps.

First, we introduce a new superpixel algorithm to over-segment the image. Our superpixel method runs in real-time and can deliver a solution at any time budget. Then, for feature extraction, we focus on the framework that computes descriptors and encodes them, followed by a pooling step. We see that the encoding step is the bottleneck, for computational efficiency and performance. We present a novel assignment-based encoding formulation, that allows for the design of a new, very efficient, encoding. Finally, the image labeling output is obtained modeling the dependencies with a Conditional Random Field (CRF). In semantic image segmentation, the computational cost of instantiating the potentials is much higher than MAP inference. We introduce Active MAP inference to *on-the-fly* select a subset of potentials to be instantiated in the energy function, leaving the rest as unknown, and to estimate the MAP labeling from such incomplete energy function.

We perform experiments on all proposed methods for the different parts of the semantic segmentation pipeline. We show that our superpixel extraction achieves higher accuracy than state-of-the-art on standard superpixel benchmark, while it runs in real-time. We test our feature encoding on standard image classification and segmentation benchmarks, and we show that our method achieves competitive results with the state-of-the-art, and requires less time and memory. Finally, results for semantic segmentation benchmark show that Active MAP inference achieves similar levels of accuracy but with major efficiency gains.

Zusammenfassung

Effiziente Algorithmen zur Objekterkennung sind essentiell für neue Anwendungen im Bereich der Robotik und Bildererkennung, die Echtzeit- und Online-Methoden verlangen. Einige Beispiele sind autonome Systeme, mobile Robotik, und autonomes Autofahren. In dieser Arbeit konzentrieren wir uns auf effiziente semantische Segmentierung, das ein Problem der Kennzeichnung jedes Pixel eines Bildes mit einer semantischen Klasse ist.

Unser Ziel ist die Beschleunigung aller Komponenten des Ablaufs der semantischen Segmentierung. Zudem wollen wir eine zeit-effiziente Lösung zur Pixel-Kennzeichnung bieten, die spontan abgerufen werden kann. Hierzu analysieren wir alle Komponenten des Segmentierungs-Ablaufes, und identifizieren die rechnerischen Engpässe jedes einzelnen. Die verschiedenen Komponenten dieses Ablaufs sind Segmentierung des Bildes in Superpixel, Extrahierung von Merkmalen und Klassifizierung der Superpixel, und letztendlich die Ableitung von den Bildmarkierungen zu semantischen Klassen. Wir gehen auf jeden dieser Schritte genauer ein.

Zunächst führen wir einen neuen Super-Pixel Algorithmus zur Segmentierung des Bildes ein. Unser Superpixel Verfahren läuft in Echtzeit ab und ist in der Lage zu jedem Zeitpunkt eine Lösung zu liefern. Beim Extrahieren der Merkmale konzentrieren wir uns auf das Programmiergerüst, das die Deskriptoren berechnet und codiert, gefolgt von einem Bündelungs-Schritt. Wir erkennen, dass der Schritt der Kodierung den Engpass für Recheneffizienz und Leistung darstellt. Wir präsentieren hier eine neue Zuordnungs-basierte Formulierung, das die Konstruktion einer neuartigen sehr effizienten Codierung ermöglicht. Die Ausgabe der Bilder-Kennzeichnung schliesslich wird durch Modellierung der Abhängigkeiten mit Conditional Random Field (CRF) erhalten. In semantischer Bildsegmentierung ist der Rechenaufwand der Instanziierung der Potentiale viel höher als MAP Inferenz. Wir führen eine Active MAP Inferenz ein, die zu jedem Zeitpunkt eine Teilmenge der Potentiale in der Energiefunktion instanziert, während der Rest als unbekannt angenommen wird, und zugleich die MAP Kennzeichnung von solch unvollständigen Energiefunktionen abschätzt.

Wir testen jede einzelne der vorgeschlagenen Methoden zu den verschiedenen Komponenten des semantischen Segmentierungsvorgangs und zeigen, dass unsere Superpixel Extrahierung höhere Genauigkeit erzielt als Stand der Technik Superpixel Methoden, abgesehen davon, dass sie noch dazu in Echtzeit ausgeführt wird. Wir testen unsere Codierung an Standard Bildklassifikations- und Segmentierungsmethoden, und zeigen, dass unsere Methode wettbewerbsfähige Ergebnisse erzielt, während sie Zeit- und Speicherplatz-effizienter ist. Zuletzt können wir zeigen, dass Active MAP Inferenz ähnliche Genauigkeiten erzielt wie andere Stand der Technik Segmentierungsmethoden, aber mit wesentlichem Gewinn an Effizienz.

Acknowledgements

First of all, I would like to thank Prof. Luc Van Gool for accepting to supervise me in the first place, and bringing me the opportunity to pursue the PhD under his mentorship. Also, for having his office door always open to my inquiries and concerns from the beginning.

A big thank you is for all co-authors of the work done during my PhD, without them this thesis would have not been possible. Not to forget the Bachelor and Master students who I had the pleasure to collaborate with, and who definitively had a great impact in my learning process.

My gratitude goes also to the BIWI members who made this years more enjoyable. Specially the 'BIWI sports club members' for encouraging me to do sports to keep a good mind-and-body balance.

My most special and sincere thanks is to Xavier, who always encouraged me to follow this path, and follow our dreams. Even though I have fallen many times, he has always given me his hand for helping me to stand up again. He has been, and still is, not only someone by my side, by also a supervisor when I need him to be, and most importantly a friend and my love.

My parents and family have been always a great support during this years. Also my friends who I have met during this journey. They have become part of my family in those years, and made me feel home.

Finally, I would like to acknowledge fundings from the European Commission, which financially supported this thesis through the IURO (FP7 ICT 248314) and RADHAR (FP7 ICT 248873) projects. Also all the partners from other institutions that collaborated with the projects.

Contents

List of Figures

List of Tables

Acronyms

AC: Assignment-based Coding

BoW: Bag-of-Words

CRF: Conditional Random Fields

HA: Hard-Assignment

LLC: Locality-constrained Linear Coding

MAP: *Maximum a Posteriori*

MRF: Markov Random Fields

NSQ: Nested Sparse Quantization

PM: Perturb and MAP Random Fields

SA: Soft-Assignment

SEEDS: Superpixels Extracted via Energy-driven Sampling

SC: Spectral Clustering

SQ: Sparse Quantization

VQ: Vector quantization

1 Introduction

Scene understanding has been one of the central goals in Computer Vision for many decades, and it is considered a key for the success of applications, such as autonomous driving, robot navigation, autonomous systems. Recognizing and localizing both objects and background is important for robotic systems, which may require a full understanding of their surrounding, for instance, to compute which is a good path to navigate. The computational resources in robotic applications are usually limited, and may vary in time. For these reason, efficient algorithms for scene understanding that are able to deliver an output on a time budget, which may be decided *on-the-fly*, are of crucial importance.

In this work, we focus on efficient semantic image segmentation, which aims at labeling each pixel of an image with a semantic class, from a predetermined set of classes. Semantic segmentation allows recognizing and localizing in the scene objects, such as people, cars, bicycles, animals; and also background, such as road, building, vegetation, water. The background usually maps groups of pixels in the image with undefined shape, and not necessarily with closed boundaries. This motivates us to focus on semantic segmentation, because the recognition is at pixel level, which gives information about the boundaries of the objects and background. Note that, object detection frameworks are more limited and not particularly suited for localizing background. This is because object detection is generally based on rectangular bounding-boxes to localize objects in the image.

Our semantic segmentation framework is able to deliver a solution in a time budget. In scenarios in which the resources may vary, such as as in autonomous navigation systems and mobile platforms, where an output solution may be needed at any time, and therefore, algorithms that are able to deliver a solution given a time budget are essential. This kinds of algorithms are also called

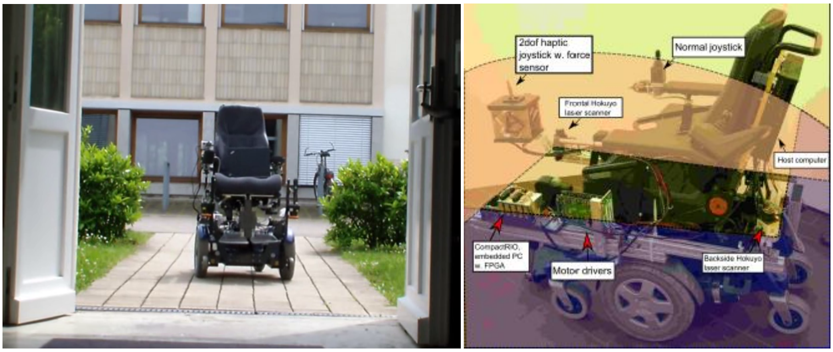

2dof haptic joystick w. force sensor
Normal joystick
Frontal Hokuyo laser scanner
Host computer
CompactRIO, embedded PC w. FPGA
Motor drivers
Backside Hokuyo laser scanner

***Figure 1.1**: RADHAR: Robotic ADaptation to Humans Adapting to Robots. It is an autonomous wheelchair to assist disable people and help them navigating.*

timely, as they are able to deliver a valid output solution given any time budget, which may be decided *on-the-fly*. Thus, the algorithms decide where to spend the computational resources to obtain the most accurate solution as possible for a restricted time budget. Such algorithms mostly rely on uncertainty measures and expected information gains of the desired solution, and are inspired from the decision making algorithms for robotics systems.

For this purpose, we identify the different steps involved in a typical semantic image segmentation pipeline, and analyze their computational bottlenecks. There are mainly three different steps: over-segmentation of the image, feature extraction and classification, and inference of the most likely image labeling. We propose new efficient algorithms for all of these steps, that are able to work on a time budget that may be decided *on-the-fly*. We introduce a new image over-segmentation method with superpixels in Chapter 3, an efficient feature extraction and classification in Chapter 4, and in Chapter 5 the Active MAP inference of the most likely image labeling, which actively decides where to spend computational efforts, on a time budget.

We have developed the contributions of this work in the framework of three robotic projects. The projects are:

- Robotic ADaptation to Humans Adapting to Robots (RADHAR) [Demeester et al., 2012; Commission, 2010–2013b].
- Interactive Urban Robot (IURO) [Commission, 2010–2013a].

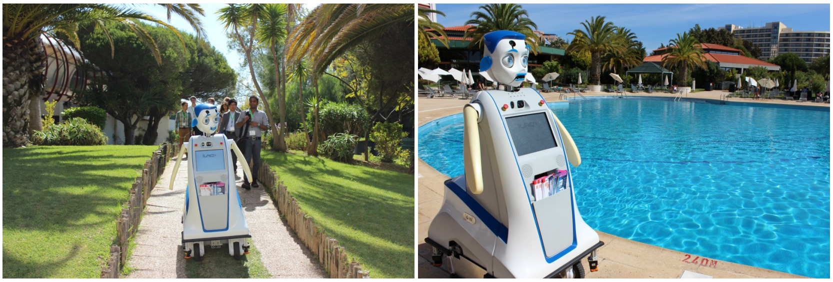


Figure 1.2: IURO: Interactive Urban Robot.

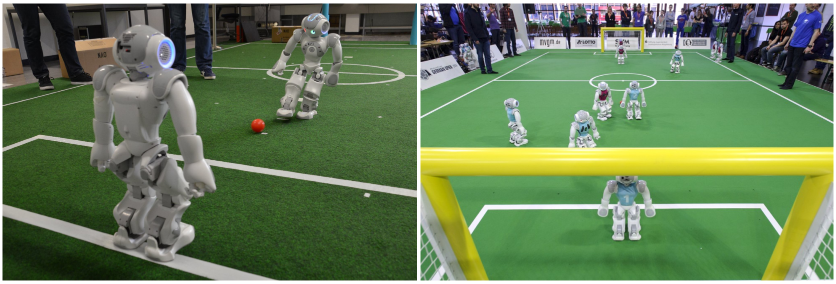


Figure 1.3: NAO robots used in the standard platform league of the Robocup, in which the robots play soccer against each other.

- Robocup team of ETH Zurich [Nguyen et al., 2011].

RADHAR project aimed at developing a driving assistance system in a smart wheel-chair, involving algorithms for environment perception, driver perception and modeling, and robot decision making. In Figure 1.1 we illustrate the RADHAR smart wheel-chair.

The main goal of the IURO project was to develop algorithms for enabling robots to navigate in the street and interact with people with robot-user interaction techniques. In Figure 1.2 we show a picture of the IURO robot.

The Robocup project at ETH aims at developing algorithms to give capabilities to the NAO robots to play soccer in the standard platform league of the Robocup tournament. We show some pictures in Figure 1.3 of the standard NAO robots from Aldebaran that are used to play soccer in the Robocup standard platform league [RoboCup, 2014].

Our contributions to these robotic projects are on the environment perception, providing semantic information of all locations in the environment that the robots are perceiving. This allows inferring the semantics of the environment and localization of people and objects, which is used for robot navigation as well as robot-human interaction purposes in the RADHAR and IURO projects. Also, the algorithms that we present in this work, have been integrated and adapted to the framework of the NAO robots for the ETH Robocup team. In the Robocup project, we use the superpixel algorithm and binary features to detect other robots in the soccer field, and the active MAP inference for deciding where to compute features and classifiers for self-localization and detection of other robots and the ball in the field.

In the following Chapter we revisit the related work and describe our semantic segmentation framework.

2

Overview of our Efficient Semantic Segmentation Framework

In this Chapter, we first revisit previous work on semantic image segmentation, and on efficient object recognition algorithms, both related to our work. Then, we present our efficient semantic segmentation framework, which also serves as outline of the remaining Chapters of this thesis.

2.1 Semantic Image Segmentation

Semantic image segmentation aims at assigning predefined class labels to every pixel in an image, and is a crucial step for automatic understanding of an image, *cf.* [Shotton & Kohli, 2014]. There are standard benchmarks to evaluate the performance of semantic image segmentation methods, and the most used in the computer vision community are the MSRC-21 and PASCAL VOC datasets. In Figure 2.1 and Figure 2.2 we show examples of images and their groundtruth of MSRC-21 and PASCAL VOC, respectively.

Semantic segmentation has been intensively studied in recent years. In this Section, we discuss the most relevant methods. We distinguish between approaches that propose candidate regions in the image to be an object, and then label the region as one entity, and approaches that first label local regions, either pixels or superpixels, and then obtain the object regions by grouping the labels of the same object instance.

Extracting regions for object candidates is mainly based on global knowledge of an object, which helps to lead to the semantic object boundaries. These methods usually use some *a priori* knowledge about the global object in order to guide the segmentation. This information can be extracted from different sources. For instance, Kumar et al. [2005] use the global shape of the object, which provides a meaningful approximation about boundaries of the object. In [Kumar et al., 2005], the shape of the object is encoded with the relative position of the parts. The method by Carreira et al. [2012b] proposes regions in the image to be object candidates, without using class-specific information. Then, those regions are ranked to do a pruning of object candidates, and only classify the most likely ones to be an object. This work was later extended by the same authors with the use of more powerful features for semantic segmentation in [Carreira et al., 2012a]. A similar approach by Arbelaez et al. [2012], proposes candidates building a hierarchy of regions. Then they classify the regions and combine the obtained semantics in the hierarchy to deliver the final labeling. In [Hariharan et al., 2014], they use the framework by Arbelaez et al. [2012] and propose to use more powerful features, and a refinement to also obtain bounding-boxes for object detection task. The main drawback of these approaches is the lack of a multi-class classification in the regions proposals, and the restricted variability of the object candidates. A similar idea, but more restrictive, is to use the bounding-box of a detection method, such as the method by Leibe et al. [2007] and by Lempitsky et al. [2009]. However, since these approaches can be understood as a refinement of the detection, the method is restricted to the detection accuracy. Other works are based on a coarse-to-fine approach based on hierarchical representation [Ladicky et al., 2009; Floros et al., 2011]. The main strengths of those methods are their ability of encoding the context of the region. However, it usually fails when background classes are not labeled for the training data and this semantic context can not be retrieved.

Other approaches label an image by only using information around each pixel or superpixel, and thus, they intrinsically do not take into account the whole structure of the object. Traditionally, the community has been divided into authors that encouraged to either use superpixels or pixels as the basic entity to be labeled. The generic state-of-the-art pixel-based approaches build upon the description of a patch around each pixel using several cues, and then apply a classifier to obtain the labeling [Jiang & Tu, 2009; Shotton et al., 2009]. In many state-of-the-art methods for semantic scene segmentation, the labeling of different regions in an image is usually done making use of contextual information [Oliva & Torralba, 2007]. A way of exploiting contextual information in such approaches is to include it at the feature level. This context can be encoded directly into the image descriptors by extending them with contextual cues. For instance, Pantofaru et al. [2008] proposed to use several segmentations and to combine them. Other authors propose to construct the model from a superpixel based approach , *e.g.* [Fulkerson et al., 2009; Jiang & Tu, 2009; Boix et al., 2012]. Another strand of research focused on combining the semantic labels estimated by different contextual features [Csurka & Perronnin, 2010; Munoz et al., 2010; Maire et al., 2011]. Similarly, prior knowledge can be provided from the use of a categorization method that infers whether the object is present in the image or not, *e.g.* [Plath et al., 2009]. In [Li et al., 2009] this information is extracted from the set of Flickr tags that the user had uploaded, and combined with a generative graphical model.

Many semantic image segmentation approaches use Markov Random Fields (MRF) and Conditional Random Fields (CRF), the success of which stems from their ability to infer the most likely scene configuration. Methods that are based on CRFs include the contextual information with the use of smoothness potentials between neighboring nodes, *e.g.* [Galleguillos et al., 2008; Gould et al., 2008], to exploit the inter-dependencies between regions. Since more sophisticated constraints are needed to encode higher level context, methods including interactions between regions at different scales have appeared. Verbeek & Triggs [2007] pioneered including the global scale information in a CRF to improve estimation at local scales. Then, a number of recent approaches have introduced sophisticated graphical models that include constraints on both local spatial smoothness and global consistency with the use of hierarchical CRFs and high-order potentials, which interact with local variables either directly or through intermediate hierarchies, *e.g.* [Kohli et al., 2009; Ladicky et al., 2010a,b; Boix et al., 2012]. In [Ion et al., 2011] a probabilistic model is used to find an agreement of labeled objects obtained by the method of Carreira et al. [2012b] to obtain the complete labeling of the image, and not only of an object.

Using a CRF allows modeling the dependencies between object and region classifiers. Yet, the uncertainty measure of the final labeling, which may be important to have in some applications, *e.g.* in robotic applications, is not straightforward to compute, since the solution is obtained with the *Maximum a Posteriori* (MAP) labeling. For estimating the uncertainty of the labeling there are various alternatives that can be used. For instance, one possibility is directly taking the scores of individual semantic classifiers, without the CRF. A higher score means a higher confidence about the object being detected, allowing to associate a confidence value to the detection. However, the individual semantic classifiers are not able to encode the structure of the labeling, which is necessary to encode contextual dependencies. Another way is to use methods that estimate the CRF density function, e.g., max-sum loopy belief propagation, which is able to approximate the marginals [Yedidia et al., 2001]. Also, Markov Chain Monte Carlo (MCMC) sampling could be used, which in theory is able to deliver any distribution of interest. However, these techniques are often avoided because belief propagation on loopy, densely connected, graphs has no guarantee on convergence nor quality of the results, and MCMC sampling in a CRF needs many iterations to guarantee a good mixing of the chain. Remarkably, Kohli & Torr [2006] introduced a method to estimate the uncertainty of the max-marginals with graph-cuts, but these do not correspond to the marginals.

In our work, we use a CRF to model the neighboring interaction of the image regions, similarly to previous works based on CRFs. Since we focus on efficient semantic image segmentation on a time budget, we not only aim at speeding-up the final inference, but all the components of the semantic segmentation pipeline. The bottleneck of the semantic segmentation pipeline is usually extracting the features and computing the classifiers in all the image regions. Thus, we do not compute the features and classifiers everywhere in the image. The CRF allows for propagating the label information based on similarity of adjacent regions in the image. We employ a Perturb-and-MAP Random Field, that allows performing fast approximate sampling from its probability density function. This allows to effectively compute the uncertainty of the solution, indicating the uncertainty of the most likely labeling in each region of the image. We exploit the uncertainty by putting more computational effort on the regions of the image that are less certain, and that we expect that will provide more information for understanding the whole scene.

After describing in the next section previous works on efficient object recognition methods and other approaches that work on a time budget, in Section 2.3 we describe our semantic segmentation framework, and how we speed it up to make it efficient and able to deliver an output, on a time budget.

2.2 Towards Efficient Object Recognition on a Time Budget

The increase of computer vision applications that demand real-time or anytime capabilities, has moved the interest of researches to study efficient object recognition algorithms. Timely applications are also becoming very popular. They consist on methods that are able to deliver an output solution on a time budget, which can be decided *on-the-fly*. The estimated output is expected to be the best possible solution for that time budget.

Among the initiating methods that emphasized efficiency, are the systems for object detection with early rejection. Such algorithms are based on the use of classifiers in cascades and use very efficient feature descriptors in all possible locations of the image. The most well-know method was presented by Viola & Jones [2004], in which they propose a cascade of classifiers based on Haar features for object detection. Their framework discards in very early stages of the cascade the candidates of being an instance of an object if it is classified as not being such object. There have been many follow-up works, in which authors propose more robust and efficient cascades of classifiers and adding more discriminant descriptors, *e.g.* [Chen & Yuille, 2005; Luo, 2005; Brubaker et al., 2008].

Methods based on cascades of classifiers imply classifying all images or possible regions of the images taking all the different classes into account. This does not scale well in general object detection problems, in which all possible regions of rectangle shape in the image are taken into account. Usually, this is of orders of millions of bounding-boxes per image. To address this scalability problem, there have been many attempts to alleviate the search. Some methods for efficient object detection are based on implicit shape models with voting schemes [Leibe et al., 2007; Maji & Malik, 2009; Gall et al., 2011; Lehmann et al., 2011]. Some tackle it with a coarse-to-fine approach, *e.g.* [Pedersoli et al., 2011; Lehmann et al., 2014], by exploring the image from broader regions and refining the area of search.

Another way to address the scalability issue has been with the object proposals using the so called objectness measure. The objectness measure was introduced by Alexe et al. [2012a]. It ranks regions of the image to be object candidates, without specifying their semantic class. The goal is to keep the best candidate regions, which it is shown to be sufficient with about thousands of candidates, and discard the rest. This allows to classify fewer regions of the image, which otherwise would be of orders of millions of image regions. There have been some follow-ups of this work which introduced new cues to boost the performance of the objectness, *e.g.* [Chang et al., 2011], but that demand more computational time. All this methods still demand on the orders of seconds to obtain the candidate regions, which is still not enough for real-time applications. Recently, there have appeared very efficient objectness methods, *e.g.* by Cheng et al. [2014] which uses binary representations and efficient computations. We also proposed an objectness measure in [Van den Bergh et al., 2013] based on a real-time superpixel algorithm.

Lately, there have been more attention on methods that are able to deliver an output on a time budget. This kind of algorithms are specially suited for robotic applications, in which in a certain amount of time, an output solution might be necessary, even if it is not the best one that it would be obtained with the full computation. The amount of time available can also be decided *on-the-fly*. One of the first works in this lines, was presented by Gao & Koller [2011]. They select which classifiers to compute in an ensemble of classifiers in a time budget, taking into account the expected gain in classification accuracy. They applied this method on image classification. In the same spirit, Karayev et al. [2014] introduced a method with decision making processes for learning a policy for selecting the features and classifiers on a time budget. The same authors, introduced in [Karayev et al., 2012] a system for object detection which selects which classes to compute in all possible bounding-boxes of the image. They use Reinforcement Learning to learn from data which classes to discriminate. A similar approach by Alexe et al. [2012b] on object detection, selects in which region of the image to compute the classifiers, taking into account the previous image locations already visited. It is based on a decision making system with a voting scheme.

The focus of this work is on efficient image semantic segmentation on a time budget. In image semantic segmentation we need a structured labeling output, which is in contrast to the previous methods described above, which aim is at predicting a single output at a time: which feature to use, which classifier to compute, or in which location of the image.

2.3 Efficient Semantic Segmentation Pipeline

We use the typical framework for semantic segmentation modeled with a Conditional Random Field (CRF). Using a CRF allows naturally encoding dependencies without the need of proposing a pool of candidate regions as a first step. Extracting the candidate regions might be a very tedious task and expensive to compute. Also, a CRF facilitates the propagation of the information from neighboring regions, with cheap to compute cues, and thus, the computation of features and classifiers everywhere in the image may not be necessary. Not computing features and classifiers everywhere in the image yields dramatic speed-ups, because computing features and classifiers is usually more expensive than doing MAP inference of the labeling. There have been many attempts in the literature to speed-up semantic segmentation by making the inference algorithms faster. In this work, we focus on all the different steps of the semantic segmentation pipeline to make it more efficient, and not only on the labeling inference, which is the last step of the pipeline.

The semantic segmentation pipeline that we use consist in three steps:

1. Over-segmentation of the image with superpixels (see Figure 2.3a).
2. Computation of features and classifier of all the local regions (see Figure 2.3b).
3. MAP inference to obtain the final labeling (see Figure 2.3c).

Our aim is to build an efficient semantic segmentation framework, that is able to deliver an output within a time budget. We make a contribution in each of the above three steps of the semantic segmentation pipeline with the aim of efficiency and computing it on a time budget: over-segmentation with superpixels, feature extraction and classification and the Active MAP inference of the most likely image labeling. In Figure 2.4 we depict our semantic segmentation pipeline.

As first step we use an over-segmentation of the image. We introduce an algorithm for efficient image over-segmentation, which is a pre-processing step that is usually used before image semantic segmentation, in which pixels of the image of similar appearance, in color and/or texture, are grouped together. The groups of perceptually similar pixels are called superpixels. Using superpixels allows reducing the number of instances to label from hundreds of thousands of pixels to few hundreds of superpixels. Since superpixels are used for efficiency reasons, they should not be computational demanding. We introduce an efficient superpixel algorithm that runs in real-time, which is based on a simple hill-climbing optimization. Starting from an initial superpixel partitioning, it continuously refines the superpixels by modifying the boundaries. We define a robust and fast to evaluate energy function, based on enforcing a similarity between the descriptors of the boundaries and the superpixel descriptor histogram. We also use new descriptors that encode both texture and color, and that are learned with spectral clustering techniques. We use a binary quantization based on sparse quantization, to reduce the computational cost of the descriptors distance to negligible, compared to the cost of using some color space. Thus, our approach avoids initial color conversions, reducing the computational cost to half, and achieves better performance than state-of-the-art on standard superpixel benchmarks.

In the second step of the pipeline, features and classifiers are computed in the superpixels to obtain a local probability for each class. We explore feature encoding methods and classification for visual object recognition, putting special focus on efficiency. We present an extremely efficient image encoding with binary representations. Using efficient image representation and classification methods is of crucial importance because it may become the bottleneck of the semantic image segmentation framework. Many state-of-the-art methods in object recognition extract features from an image and encode them, followed by a pooling step and classification. At the heart of our formulation lies a quantization into a set of k-sparse vectors, which we denote as sparse quantization. We design the new encoding as two nested, sparse quantizations. Its efficiency stems from leveraging bit-wise representations. We successfully apply our new feature encoding to image classification and semantic segmentation. We perform experiments and show that our feature encoding is competitive to state-of-the-art methods in terms of accuracy, and our method requires orders of magnitude less time and memory.

Finally, in the third step we do MAP inference by building a graph with a node per superpixel, and edges between neighboring superpixels that encode the similarity between such superpixels. We introduce a method that allows actively deciding where to put more emphasis on the calculation of the features and classifiers in the image, and then infer the labeling of the whole image. We compute features and classifiers in some regions of the image, and not in all image, depending on the amount of time available. Computing the features and classifiers in fewer places of the image may cause the accuracy to suffer. The effect of computing less features and classifiers is illustrated in Figure 2.5. We select which places are best to compute features and classifiers, such that the semantic segmentation does not suffer from a significant decrease in accuracy performance. Our framework is based on a CRF that encodes the relations of neighboring superpixels based on appearance similarity. Most *Maximum a*

Posteriori (MAP) inference algorithms for CRFs optimize an energy function knowing all the potentials. In semantic image segmentation, instantiating the potentials by classifiers fed with features is computationally more expensive than doing MAP inference. We introduce Active MAP inference 1) to on-the-fly select a subset of potentials to be instantiated in the energy function, leaving the rest of the parameters of the potentials unknown, and 2) to estimate the MAP labeling from such incomplete energy function. Results for semantic segmentation benchmarks show that Active MAP inference achieves similar levels of accuracy as instantiating all potentials of the CRF, but with major efficiency gains.

Our active semantic segmentation framework can also deliver the uncertainty of the labeling solution. Our work has been developed in the context of robotic projects, in which it is often desirable to have a measure of uncertainty of the perceptual solution. Awareness of the uncertainty of perception is fundamental for proper high level decision making systems, for fusing computer vision algorithms and robotics systems. This is because computer visual perception capabilities are still highly unreliable in unconstrained settings, and solutions might not be accurate in all image regions.

2.4 Outline of the Thesis

We first introduce in Chapter 3 a method for real-time over-segmentation of images, also known as superpixels, that will provide the local regions in which to compute the features and classifiers, which correspond to the unary potentials of the graphical model. Then, in Chapter 4 we introduce a method to speed up the computation of the features to obtain the semantic evidence of the superpixels of the image. Then, in Chapter 5, we introduce active MAP inference, which selects which regions of the image should be observed to obtain the most accurate semantic segmentation of the image given a time budget and how to do inference on a incomplete energy function. We depict the pipeline of our efficient semantic segmentation framework in Figure 2.6, and their respective Chapter in this thesis. Finally, we conclude and give future directions of this work in Chapter 6 and 7, respectively.

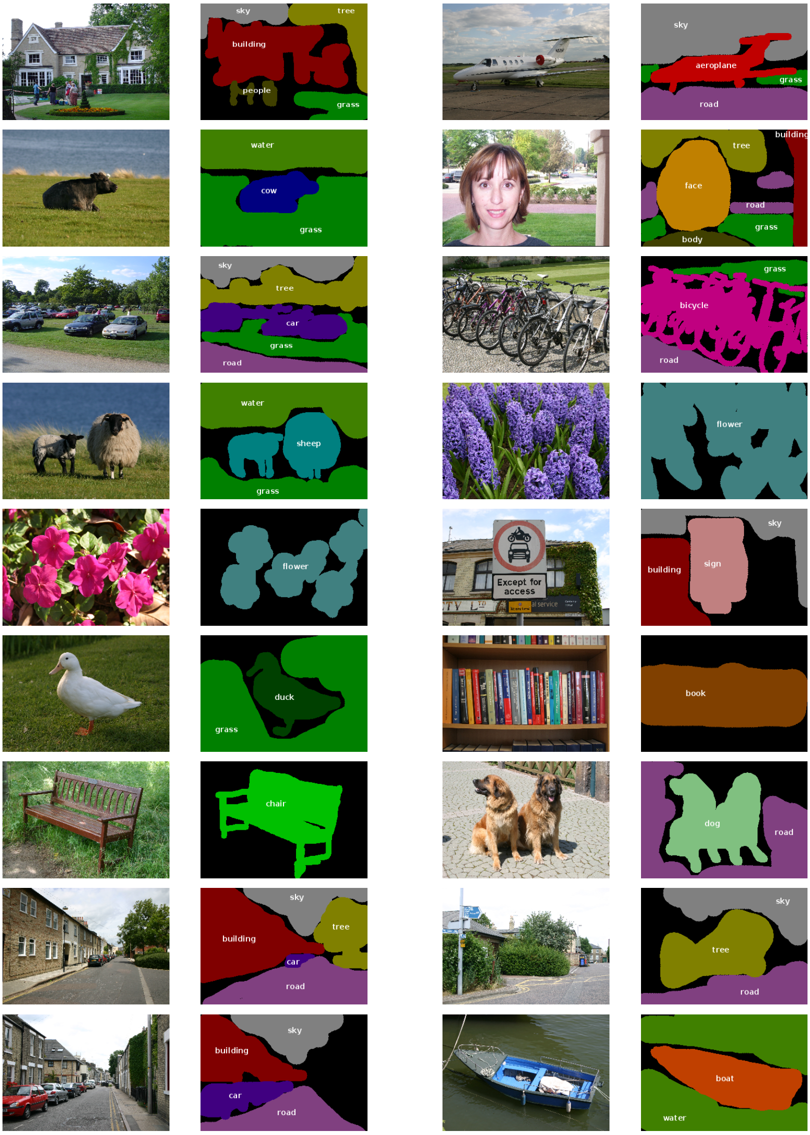


Figure 2.1: *Examples of images and groundtruth of MSRC-21 for semantic image segmentation. Each color of the groundtruth corresponds to a semantic class.*

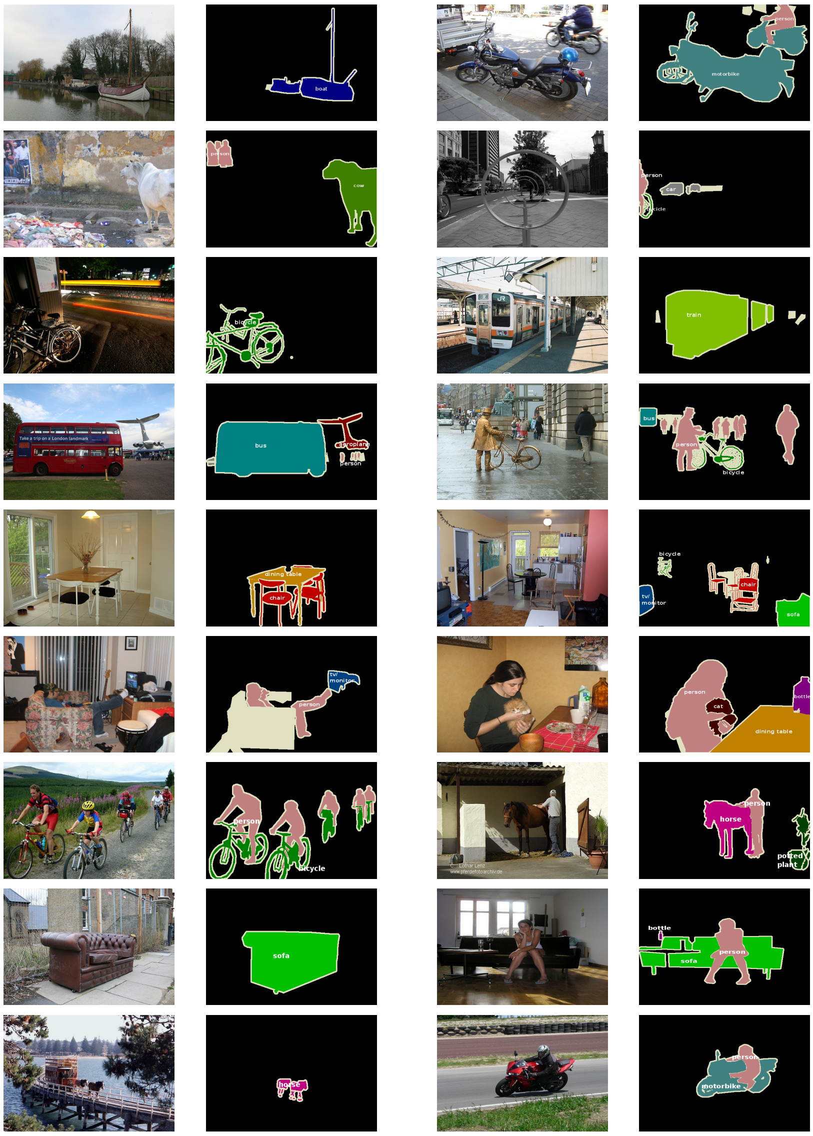
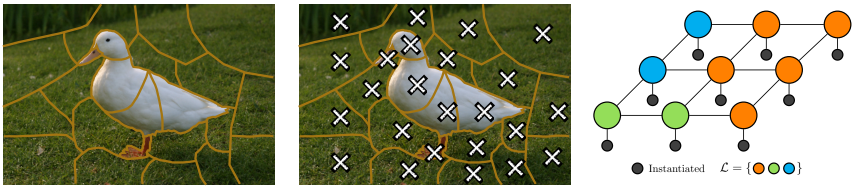
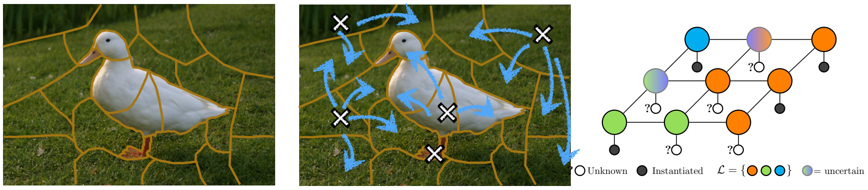


Figure 2.2: *Examples of images and groundtruth of PASCAL VOC for semantic image segmentation. Each color of the groundtruth corresponds to a semantic class.*



(a) (b) (c)

Figure 2.3: *Pipeline of image semantic segmentation with CRF: (a) superpixels, which boundaries are in yellow, (b) compute features and classifiers for each superpixel, marked with the white cross on the left image, and the gray circle in the graph, and (c) MAP inference.*



(a) (b) (c)

Figure 2.4: *Pipeline of image semantic segmentation for efficiency, with Active MAP inference for CRF: (a) superpixels, which boundaries are in yellow, (b) compute features and classifiers for some superpixels, marked with the white cross on the left image, and the gray circle in the graph, and then, (c) Active MAP inference, in which the graph is fully labeled, but the labels have an uncertainty associated. In the graph, if the features and classifiers are not computed is denoted with a '?'.*

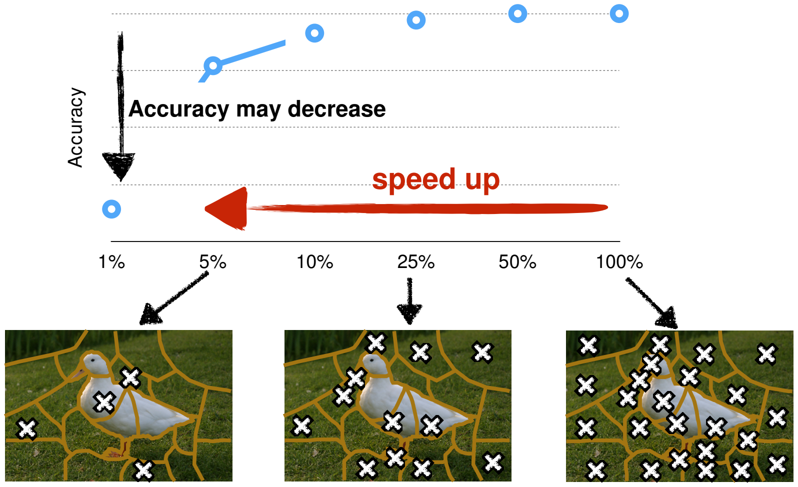


***Figure 2.5**: The performance accuracy may decrease if less superpixels are instantiated. The goal is to select the superpixels that are best such that the accuracy decreases the minimum amount, given a budget of time.*

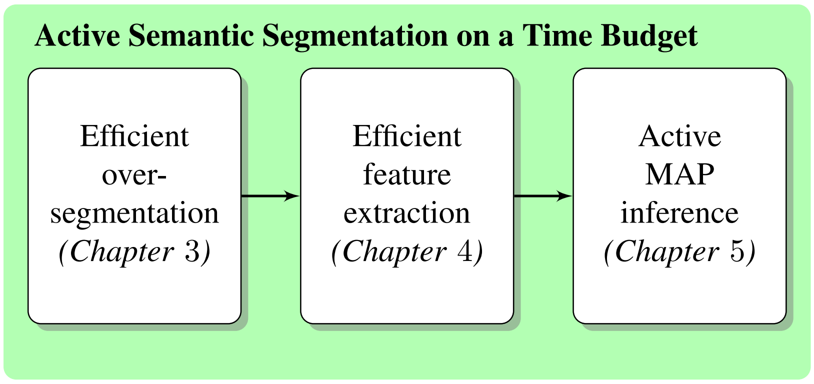

Active Semantic Segmentation on a Time Budget
Efficient over-segmentation (Chapter 3)
Efficient feature extraction (Chapter 4)
Active MAP inference (Chapter 5)

***Figure 2.6**: Semantic Segmentation Pipeline, and outline of this work.*

3

Efficient Image Over-Segmentation

3.1 Introduction

Many computer vision applications benefit from working with superpixels instead of just pixels, *e.g.* [Fulkerson et al., 2009; Wang et al., 2011; Alexe et al., 2012a; Boix et al., 2012]. Superpixels reduce the number of entities to be processed by the computer vision algorithms, and enable feature computation on bigger, more meaningful regions than pixels. Superpixels are of special interest for semantic segmentation, in which they are reported to bring major advantages. They reduce the number of entities to be labeled semantically and enable feature computation on bigger, more meaningful regions.

There is a surfeit of superpixel algorithms by now. Many superpixel algorithms are posed as a clustering problem with some additional constraints, that enforce that the superpixels form a continuous blob of pixels. Since superpixels are often used as a pre-processing step to reduce the computational cost of computer vision applications, some authors stressed the need that the computational cost of the superpixel extraction should be orders of magnitude lower than the final application [Felzenszwalb & Huttenlocher, 2004; Levinshtein et al., 2009]. It was not until recently that a superpixel algorithm running in real-time in a single CPU without the need of additional hardware, achieved state-of-the-art performance accuracy. This is the SEEDS superpixels, which we proposed in [Van den Bergh et al., 2012, 2014].

In SEEDS, we start from a complete superpixel partitioning, and we iteratively refine it. The refinement is done by moving the boundaries of the superpixels, or equivalently, by exchanging pixels between neighboring superpixels. The objective function of SEEDS can be maximized efficiently, and is based on enforcing homogeneity of the color distribution of the superpixels, plus an optional term that encourages smooth boundary shapes. The optimization is based on a hill-climbing algorithm, in which a proposed movement for refining the superpixels is accepted if the objective function increases. The SEEDS superpixel algorithm can be easily extended to work on videos and take into account spatio-temporal information [Van den Bergh et al., 2013], delivering an on-line partitioning.

A crucial step for the success of superpixel extraction are the pixel descriptors, because they determine the similarity between pixels, and may have a non-negligible cost. Typically, the pixel descriptors are simply a 3-dimensional vector that represents the pixel color, in a color space selected at hand, and may become the computational bottleneck (*e.g.* in SEEDS the color conversion takes half of the computational cost). Thus, we also investigate the learning of pixel descriptors for superpixel extraction, which has received little attention in the Computer Vision literature. We use SEEDS superpixels, and Spectral Clustering techniques [Rahimi & Recht, 2004; Zhang & Jordan, 2008] to learn descriptors from patches extracted around each pixel. We coin our method Spectral SEEDS. The learned descriptors allow to circumvent the color conversion used in SEEDS, and exploit the texture around the pixels. Also, we introduce binary quantization that allow to compute the pixel descriptors with a negligible cost compared to the rest of SEEDS algorithm.

In the experiments, we show that Spectral SEEDS reduces the computational cost of SEEDS to half, and both, SEEDS and Spectral SEEDS, achieve higher accuracy than state-of-the-art on the BSD500 dataset [Martin et al., 2001].

3.2 Towards Efficiently Extracted Superpixels

In this Section, we revisit the literature on superpixel extraction. The concept of superpixels as a pre-processing step was first introduced by Ren & Malik [2003]. They defined the superpixels as an over-segmentation of the image based on the principles of grouping developed by the classical Gestalt theory by Wertheimer [1938]. We divide the existing superpixel methods in two families, putting special emphasis on their compromise between accuracy and run-time. In the first one, the methods are based on graphs and work by gradually adding cuts. In the other, they gradually grow superpixels starting from an initial set. We add a third approach, which we introduced in [Van den Bergh et al.,

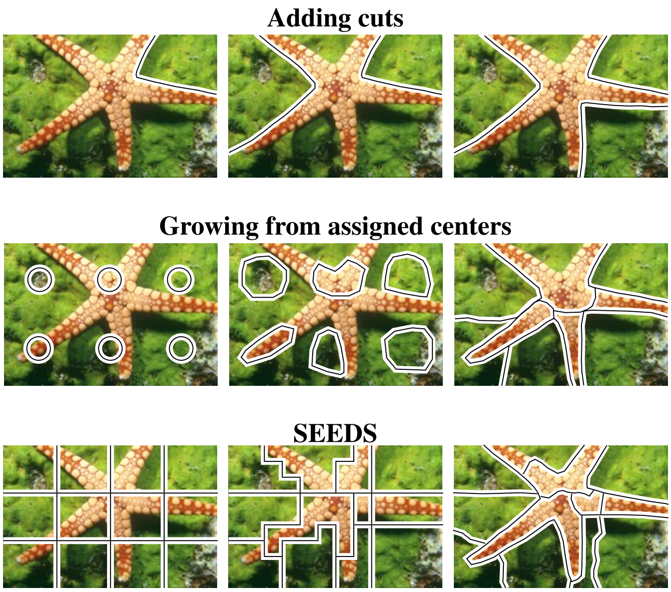


Figure 3.1: *Comparison of different strategies to build superpixels. Top: the image is progressively cut. Middle: the superpixels grow from assigned centers. Bottom: the presented method (SEEDS) proposes a novel approach: it initializes the superpixels in a gird, and continuously exchanges pixels on the boundaries between neighboring superpixels.*

2012], which moves the boundaries from an initial superpixel partitioning. We illustrate the different methods in Figure 3.1.

3.2.1 Gradual Addition of Cuts

Typically, superpixel methods based on a gradual addition of cuts are built upon an objective function that takes the similarities between neighboring pixels into account, and use a graph to represent them. Usually, the nodes of the graph represent pixels, and the edges their similarities. Shi & Malik [2000] introduced the seminal Normalized Cuts algorithm. It is based on the earlier work by Wu & Leahy [1993], which globally minimizes a graph-based objective function, by finding the optimal partition in the graph recursively. In [Shi & Malik, 2000], the cut cost is improved by normalizing it, taking into account all the nodes in the graph. In this way, they avoid favoring the cuts in small sets of nodes in the graph. Normalized Cuts is computationally demanding, and there have been attempts to speed it up by adding constraints [Eriksson et al., 2007; Xu et al., 2009], or by decomposing the graph in multiple scales [Cour et al., 2005].

Another strategy to improve the efficiency of graph-based methods was introduced by Felzenszwalb & Huttenlocher [2004]. They presented an agglomerative clustering of the nodes of the graph, which is faster than Normalized Cuts. However, Levinshtein et al. [2009] and Veksler & Boykov [2010] showed that it produces superpixels of irregular size and shapes which might not be desirable. The algorithm by Moore et al. [2008, 2010] finds the optimal cuts by using pre-computed boundary maps. Yet, the performance of this algorithm depends on the quality of such boundary maps. Veksler & Boykov [2010] place overlapping patches over the image and assign each pixel to one of those by inferring a solution with graph-cuts. Based on this work, Zhang et al. [2011] proposed an efficient algorithm that uses a pseudo-boolean optimization and achieves 0.5 seconds per image.

Recently, Liu et al. [2011b] introduced a new graph-based energy function and surpassed the previous results in terms of quality. Their method maximizes the entropy rate of the cuts in the graph, plus a balancing term that encourages superpixels of similar size. They show that maximizing the entropy rate favors the formation of compact and homogeneous superpixels, and they optimize it using a greedy algorithm. However, they also report that the algorithm takes about 2.5 s to segment an image of size 480×320.

3.2.2 Growing superpixels from assigned centers

There are methods not based on graphs. Watersheds is among the pioneers [Vincent & Soille, 1991; Meyer & Maragos, 1999]. It uses the gradient image, which is seen as a topological surface, and the superpixels are created by flooding the gradient image. A more recent method based on similar principles is Turbopixels [Levinshtein et al., 2009], which grows regions following geometric flows, until the superpixels are formed.

Achanta et al. [2012] introduced SLIC algorithm, which substantially improves the efficiency of superpixel extraction. SLIC starts from a regular grid of centers or segments, and grows the superpixels by clustering pixels around the cen-

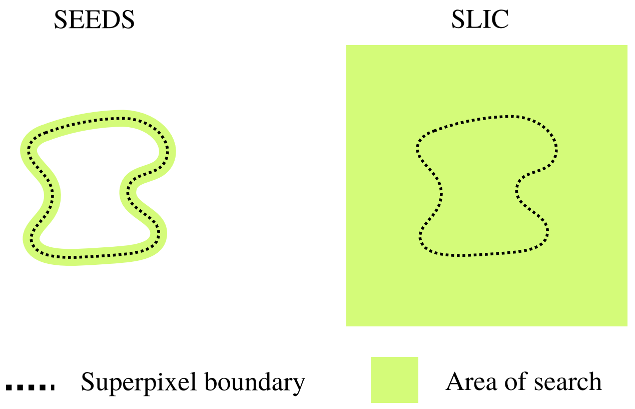


***Figure 3.2**: SEEDS vs SLIC. Area of search for the superpixel updates, which have a great impact on the computational time. SEEDS is much more efficient because it only updates superpixels at the boundary of superpixels.*

ters. At each iteration, the centers are updated, and the superpixels are grown again. Zeng et al. [2011] formulates this algorithm taking into account the geodesic distances between pixels, and accepts adding new superpixel centers. Consistent Segmentation by Zitnick et al. [2005] is based on similar principles, but it also estimates the optical flow jointly with the segmentation in video sequences using appearance and motion constraints.

A different strategy is followed by Quick-Shift [Vedaldi & Soatto, 2008]. It performs fast mean-shift, which was introduced by Comaniciu & Meer [2002], with a non-parametric clustering and with a non-iterative algorithm.

Even though all these methods are more efficient than graph-based alternatives, they do not run in real-time, and in most cases they obtain inferior performance. SLIC, being the fastest among them, is able to run at 5Hz.

3.2.3 Update the Boundaries

SEEDS is related to the methods that grow superpixels from an initial set in the sense that it also starts from a regular grid. Yet, it does not share their bottleneck of needing to iteratively grow superpixels. Growing might imply computing some distance between the superpixel and all surrounding pixels in each iteration, which comes at a non-negligible cost. SEEDS bypasses growing superpixels from a center, because it directly exchanges pixels between superpixels by moving only the boundaries. In Figure 3.2 we illustrate the updates of SEEDS compared to SLIC. While SEEDS updates only on the superpixel's boundary, SLIC updates all pixels in a predefined area around the center of the superpixel. In the experiments section, we show that SEEDS superpixels achieve state-of-the-art accuracy performance, and superpixels are extracted in real-time on a single CPU with no additional hardware.

After introducing SEEDS in more detail, in the sequel, we show that the pixel descriptors for SEEDS can be learned with spectral clustering techniques. Also, SEEDS superpixels extraction can be further speeded-up by using the learned descriptors and binary quantizations.

3.3 SEEDS Superpixels

A superpixel segmentation is an over-segmentation that usually enforces a consistent appearance inside superpixels and a regular shape of the superpixel boundaries. The quality of a superpixel is measured by its property of grouping similar pixels that belong to the same object, and by how well it follows object boundaries. We introduce the superpixel segmentation as an energy maximization problem where each superpixel is defined as a region with a color distribution and a shape of the boundary.

Let N be the number of pixels in the image, and K the number of superpixels that we want to obtain. We represent a partitioning of the image into superpixels with the mapping

$$p : \{1, \ldots, N\} \to \{1, \ldots, K\}, \tag{3.1}$$

where $p(i)$ denotes the superpixel to which pixel i is assigned. Also, we can represent an image partitioning by referring to the set of pixels in a superpixel, which we denote as $\mathcal{A}_k$: $\mathcal{A}_k = \{i : p(i) = k\}$, and thus, $\mathcal{A}_k$ contains the pixels in superpixel k. The whole partitioning of the image is represented with the sets $\{\mathcal{A}_k\}$. Since a pixel can only be assigned to a single superpixel, all sets $\mathcal{A}_k$ are restricted to be disjoint, and thus, the intersection between any pair of superpixels is always the empty set: $\mathcal{A}_k \cap \mathcal{A}_{k'} = \emptyset$. In the sequel, we interchangeably use p or $\{\mathcal{A}_k\}$ to represent a partitioning of the image into superpixels.

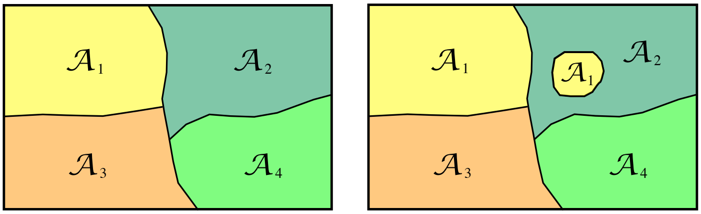


***Figure 3.3**: Left: an example partitioning in $\mathcal{P}$, where the superpixels are connected. Right: the partitioning is in $\mathcal{C}$ but not in $\mathcal{P}$ as it is an invalid superpixel partitioning.*

A superpixel is valid if it is spatially connected as an individual blob. We define $\mathcal{P}$ as the set of all partitionings into valid superpixels, and $\bar{\mathcal{P}}$ as the set of invalid partitionings, as shown in Figure 3.3. Also, we denote $\mathcal{C}$ as the more general set that includes all possible partitions (valid and invalid).

The superpixel problem aims at finding the partitioning $p \in \mathcal{P}$ that maximizes an objective function, or so called energy function. We denote the energy function as $H(s, I)$, where I is the input image. In the following, we will omit the dependency of the energy function on I for simplicity of notation. Then, we define $p^\star$ as the partitioning that maximizes the energy function:

$$p^\star = \arg\max_{p \in \mathcal{P}} H(p). \tag{3.2}$$

This optimization problem is challenging because the cardinalities of $\mathcal{P}$ and $\mathcal{C}$ are huge. In fact, $|\mathcal{C}|$ is the Stirling number of the second kind, which is of the order of $\frac{K^n}{K!}$ [Sharp, 1968]. What also renders the exploration of $\mathcal{P}$ difficult, is how $\mathcal{P}$ is embedded into $\mathcal{C}$. For each element in $\mathcal{P}$ there exists at least one element in $\bar{\mathcal{P}}$ which only differs in one pixel. This means that from any valid image partitioning, we are always one pixel away from an invalid solution.

In the following we introduce the energy function and the optimization algorithm of SEEDS superpixels. Then, we describe the extension to obtain SEEDS superpixels on videos, that are consistent over time.

3.3.1 Energy Function

The energy function, $H(p)$, evaluates the color distribution of the superpixels. We could also add an optional term for the prior of the shape of the superpixel boundaries. Yet, we found that the prior of the shape does not have an impact on the accuracy performance, and hence, we do not use it for the development of our algorithm. Superpixels should be perceptually consistent, in particular, they should be as homogeneous in color as possible. Nonetheless, it is unclear which is the best mathematical way to evaluate the homogeneity of color in a region. Almost each paper on superpixels in the literature introduces a new energy function to maximize, but none of them systematically outperforms the others. We introduce a novel measure on the color density distribution in a superpixel, that allows for efficient maximization with the hill-climbing approach.

Our energy function is built upon evaluating the color density distribution of each superpixel. We assume that the color distribution of each superpixel is independent from the rest. Let $\Psi(c_{\mathcal{A}_k})$ be a quality measure of a color distribution, and we define $H(p)$ as an evaluation of such quality in each superpixel k, *i.e.*

$$H(p) = \sum_k \Psi(c_{\mathcal{A}_k}). \tag{3.3}$$

$\Psi(c_{\mathcal{A}_k})$ is a function that enforces that the color distribution is concentrated in one or few colors. A common way to approximate a density distribution is discretizing the space into bins and building a histogram. Let λ be an entry in the color space, and $\mathcal{H}_j$ be a closed subset of the color space. $\mathcal{H}_j$ is a set of λ's that defines the colors in a bin of the histogram. We denote $c_{\mathcal{A}_k}(j)$ as the color histogram of the set of pixels in $\mathcal{A}_k$, and it is

$$c_{\mathcal{A}_k}(j) = \frac{1}{Z} \sum_{i \in \mathcal{A}_k} \delta(I(i) \in \mathcal{H}_j). \tag{3.4}$$

$I(i)$ denotes the color of pixel i, and Z is the normalization factor of the histogram. $\delta(\,\cdot\,)$ is the indicator function, which in this case returns 1 when the color of the pixel falls in the bin j of the color histogram, which has B bins in total. The pixels are represented using the Lab color space, which we found that gives better accuracy results in practice. We can also build a histogram based on learned texture filters, as we show in the sequel.

Algorithm 1: Hill-climbing in SEEDS

```
$p_t = initialize()$;
while $t < t_{stop}$ do
    $p = Propose(p_t)$;
    if $H(p) < H(p_t)$ then
        $p_t = p$;
    end
end
$p^\star = p_t$;
```

We define $\Psi(c_{\mathcal{A}_k})$ to enforce that the histogram is concentrated in few bins. A valid measure could be the entropy of the histogram. Yet, we found that the following measure is advantageous:

$$\Psi(c_{\mathcal{A}_k}) = \sum_{\{\mathcal{H}_j\}} (c_{\mathcal{A}_k}(j))^2. \tag{3.5}$$

Next, we show that this objective function can be optimized very efficiently by a hill-climbing algorithm, as histograms can be evaluated and updated efficiently. Observe that $\Psi(c_{\mathcal{A}_k})$ in Equation (3.5) encourages homogeneous superpixels, since the maximum of $\Psi(c_{\mathcal{A}_k})$ is reached when the histogram is concentrated in one bin, which gives $\Psi(c_{\mathcal{A}_k}) = 1$. In all the other cases, the function is lower, and it reaches its minimum in case that all bins take the same value. The main drawback of this energy function is that it does not take into account whether the colors are placed in bins far apart in the histogram or not. However, this is alleviated by the fact that we aim at over-segmenting the image, and each superpixel might tend to cover an area with a single color (or texture).

3.3.2 Superpixels via Hill-Climbing Optimization

We introduce a hill-climbing optimization for extracting superpixels. Hill-climbing is an optimization algorithm that iteratively updates the solution by proposing small local changes at each iteration. If the energy function of the proposed partitioning increases, the solution is updated. We denote $p \in \mathcal{P}$ as the proposed partitioning, and $p_t \in \mathcal{P}$ the lowest energy partitioning found at the instant t. A new partitioning p is proposed by introducing local changes

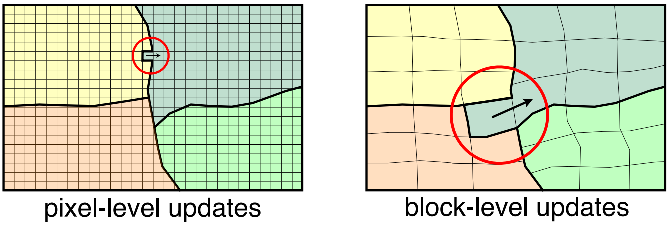


Figure 3.4: *Illustration of the movements at pixel-level and at block-level in the hill-climbing algorithm of SEEDS.*

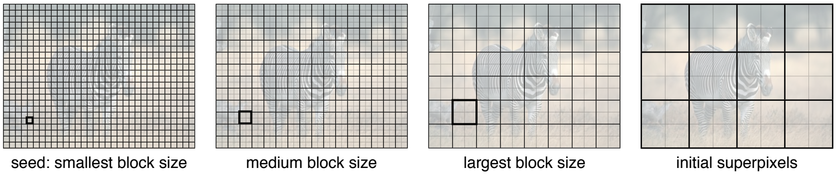


Figure 3.5: Initialization. *Example of initialization with 12 superpixels and blocks of different sizes. The initialization occurs from left to right: first the smallest blocks are initialized, and then concatenated* 2×2 *to form larger blocks. The largest blocks are concatenated* 2×2 *to create the initial superpixels. This rectangular grid (in this case* 4×3*) is the starting point of the SEEDS algorithm.*

at p_t, which in our case consists of moving some pixels from one superpixel to its neighbors. An iteration of the hill-climbing algorithm can be extremely efficient, because small changes to the partitioning can be evaluated very fast in practice.

An overview of the hill-climbing algorithm is shown in Algorithm 1. After initialization, the algorithm proposes new partitionings at two levels of granularity: pixel-level and block-level. Pixel-level updates move a superpixel boundary by 1 pixel, while block-level updates move a block of pixels from one superpixel to another. We illustrate the pixel-level and block-level updates in Figure 3.4. We will show that both types of updates can be seen as the same operation, at a different scale.

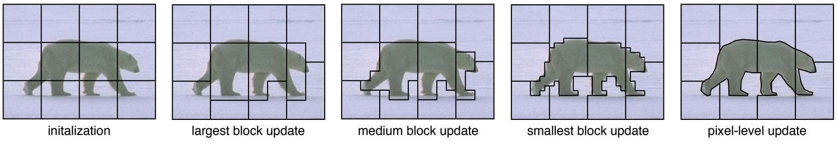


Figure 3.6: Block and pixel movements. *This figure shows an example of the evolution of the superpixel boundaries while going through the iterations of the SEEDS algorithm (in the case of 12 superpixels). From left to right: The first image shows the initialization as a grid. The subsequent images show the block updates from large to small. The last image shows the pixel-level update of the superpixel boundaries.*

Initialization

In hill-climbing, in order to converge to a solution close to the global optimum ($p^\star$), it is important to start from a good initial partitioning. We propose a regular grid as a first rough partitioning, which obeys the spatial constraints of the superpixels to form a partitioning in $\mathcal{P}$. We found that when evaluating a grid against the standard evaluation metrics, the performance is respectable: the grid achieves a reasonable over-segmentation, but of course fails at recovering the object boundaries. Observe that object boundaries are maximally half of the grid size away from the grid boundaries. This justifies using hill-climbing optimization for extracting superpixels, since the initialization is relatively close to the optimal solution.

Besides, we initialize the blocks of pixels (for the block movements) at different sizes, and compute the color histogram for each block. First, we generate the smallest block size, which is a block of 2×2 or 3×3 pixels. In order to generate larger block sizes, the small blocks are hierarchically joined in a 2×2 fashion. The corresponding histograms can be obtained by summing the histograms of the composing blocks, as shown in Figure 3.5.

The largest block size in the algorithm is a quarter of the target superpixel size. Thus, the superpixels are initialized as the concatenation of 2×2 blocks of the largest block size. This results in superpixels of a consistent size, independent from the size of the input image. The desired number of superpixels can be obtained by choosing the initial block size and number of block levels accordingly.

Proposing Pixel-level and Block-level Movements

In each iteration, the algorithm proposes a new partitioning p based on the previous one, p_t. The elements that are changed from p_t to p are either single pixels or blocks of pixels that are moved to a neighboring superpixel. We denote $\mathcal{A}_k^l$ as a candidate set of one or more pixels to be exchanged from the superpixel $\mathcal{A}_k$ to its neighbor $\mathcal{A}_n$. In the case of pixel-level updates $\mathcal{A}_k^l$ contains one pixel, and in the case of block-level updates $\mathcal{A}_k^l$ contains a small set of pixels, as illustrated in Figure 3.4. At each iteration of the hill-climbing, we generate a new partitioning by randomly picking $\mathcal{A}_k^l$ from all boundary pixels or blocks with equal probability, and we assign the chosen $\mathcal{A}_k^l$ to a random superpixel neighbor $\mathcal{A}_n$. In case it generates an invalid partitioning, which can only happen when a boundary movement splits a superpixel in two parts, it is discarded.

Block-level updates are used for reasons of efficiency, as they allow for faster convergence, and help to avoid local maxima. Note that block-level updates are more expensive, but move more pixels at the same time. Therefore, it is better to do large block-level updates at the beginning of the algorithm, and then smaller blocks, and finish the algorithm with pixel-level tuning of the boundaries. Thus, the algorithm first starts updating the larger blocks in the hierarchy, and then goes down one level in the hierarchy to update smaller blocks, and so on, until the pixel level. Once the pixel level is reached, there are not further updates in upper levels of the hierarchy. This is illustrated in Figure 3.6. The longer the individual pixel updating is running, the more accurate the resulting superpixels will be.

Evaluating Pixel-level and Block-level Movements

The proposed partitioning p is evaluated using the energy function, $H(p)$. In the following we describe the efficient evaluation of $H(p)$, and the efficient updating of the color distributions in case p is accepted.

We introduce an efficient way to evaluate $H(p)$ based on the intersection distance. Recall that the intersection distance between two histograms is

$$\mathbf{int}(c_{\mathcal{A}_a}, c_{\mathcal{A}_b}) = \sum_j \min\{c_{\mathcal{A}_a}(j), c_{\mathcal{A}_b}(j)\}, \tag{3.6}$$

where j is a bin in the histogram. Observe that it only involves $|\{\mathcal{H}_j\}|$ comparisons and sums, where $|\{\mathcal{H}_j\}|$ is the number of bins of the histogram. Recall that $\mathcal{A}^l_k$ is the set of pixels that are candidates to be moved from the superpixel $\mathcal{A}_k$ to $\mathcal{A}_n$. We base the evaluation of $H(p) > H(p_t)$ on the following observation. More details of the observation are provided in the appendix.

Observation 1. *Let the sizes of $\mathcal{A}_k$ and $\mathcal{A}_n$ be similar, and $\mathcal{A}^l_k$ much smaller,* i.e. $|\mathcal{A}_k| \approx |\mathcal{A}_n| \gg |\mathcal{A}^l_k|$. *If the histogram of $\mathcal{A}^l_k$ is concentrated in a single bin, then*

$$\mathbf{int}(c_{\mathcal{A}_n}, c_{\mathcal{A}^l_k}) \geq \mathbf{int}(c_{\mathcal{A}_k \backslash \mathcal{A}^l_k}, c_{\mathcal{A}^l_k}) \iff H(p) \geq H(p_t). \tag{3.7}$$

Observation 1 can be used to evaluate whether the energy function increases or not by simply computing two intersection distances. However, it makes two assumptions about the superpixels. The first is that the size of $\mathcal{A}^l_k$ is much smaller than the size of the superpixel, and that both superpixels have a similar size. When $\mathcal{A}^l_k$ is a single pixel or a small block of pixels, it is reasonable to assume that this is true for most cases. The second assumption is that the histogram of $\mathcal{A}^l_k$ is concentrated in a single bin. This is always the case if $\mathcal{A}^l_k$ is a single pixel, because there is only one color. In the block-level case it is reasonable to expect that the colors in each block are concentrated in few bins. When running the algorithm on the standard superpixels benchmark, these assumptions hold in 93% of the cases, and likely this is not true when the largest block updates are made.

Interestingly, in the case of evaluating a pixel-level update, the computation of the intersection can be achieved with a single access to memory, as depicted in Figure 3.7. This is because the histogram of a pixel has a single bin activated with a 1, and hence, the intersection distance is the value of the histogram of the superpixel. Note that without Observation 1, the evaluation of $H(p) \geq H(p_t)$ would be much more expensive to compute: it would require evaluating Equation (3.5) for superpixel n and k with and without the update, which involves for each side of the inequality $|\{\mathcal{H}_j\}|$ products and $|\{\mathcal{H}_j\}|$ sums.

Updating the Color Distributions

Once a new partition has been accepted, the histograms of $\mathcal{A}_k$ and $\mathcal{A}_n$ have to be updated efficiently. In the pixel-level case, this update can be achieved with

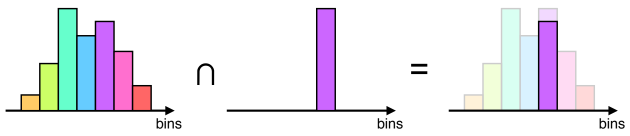


***Figure 3.7**: The intersection between two histograms, when one is the color distribution of a single pixel, can be computed with a single access to memory.*

a single increment and decrement of bin j of the the respective histograms. In the block-level case, this update is achieved by subtracting $c_{\mathcal{A}_k^l}$ from $c_{\mathcal{A}_k}$ and adding it to $c_{\mathcal{A}_n}$.

Termination

When stopping the algorithm, one obtains a valid image partitioning with a quality depending on the allowed run-time. The longer the algorithm is allowed to run, the higher the value of the objective function will get. The algorithm will usually be terminated during pixel-level updating of the boundaries. However, should one choose to terminate the algorithm very early on in the algorithm during the block-level updates, the algorithm still returns a valid partitioning.

We can set t_{stop} depending on the application, or we can even assign a time budget *on-the-fly*. We believe this to be a crucial property for on-line applications, but nonetheless one that has received little attention in the context of superpixel extraction so far. In graph-based superpixel algorithms, one has to wait until all cuts have been added to the graph, and in methods that grow superpixels, one has to wait until the growing is done, the cost of which is not negligible. The hill-climbing approach uses more iterations than previous methods, but each iteration is done extremely fast. Since the time to finish the current iteration in the hill-climbing is negligible, the algorithm can be stopped at any given time.

3.3.3 SEEDS for Videos

SEEDS algorithm can be easily extended to work on video, taking into account the temporal consistency, as we showed in [Van den Bergh et al., 2013]. Our video approach propagates superpixels over multiple frames to build 3D spatio-temporal constructs. As time goes on, new video superpixels can appear and others may terminate. In the literature, this is controlled by constraining the number of superpixel tubes in the sequence. For online applications this is not possible however, since the upcoming length and content of the sequence are unknown. Thus, we use alternative constraints defined through 2 parameters:

- *Superpixels per frame:* number of superpixels in which each single frame is partitioned.
- *Superpixel rate*: the rate of creating/terminating superpixels over time.

In order to fulfill both constraints, the termination of a superpixel implies the creation of a new one in the same frame.

We now define $\mathcal{P}$ as the set of valid partitions of a video. These are the partitions for which the superpixels are contiguous blobs in all frames and that exhibit the correct superpixel-per-frame and superpixel-rate behavior. We denote $\mathcal{A}_k^t$ as the set of pixels that belong to superpixel k, at frame t. To indicate all pixels of the video superpixel up to frame t, we use $\mathcal{A}_k^{t:0}$.

Similarly to SEEDS superpixels, the energy function encourages color homogeneity within the 3D superpixels. We use a color histogram of each superpixel to evaluate this. The color histogram of $\mathcal{A}_k^{t:0}$ is written as $c_{\mathcal{A}_k^{t:0}}$. Recall that $\mathcal{H}_j$ is a subset of the color space which determines the colors in a bin of the histogram. Then the energy function is

$$H(p) = \sum_k \sum_{\{\mathcal{H}_j\}} (c_{\mathcal{A}_k^{t:0}}(j))^2, \tag{3.8}$$

which is maximal when the histograms have only one non-zero bin for each video superpixel.

The optimization algorithm is designed to maximize the energy function in an online fashion (*i.e.* only using past frames and at video rate). It computes the partition of the current frame, starting from an approximation of the last partition. Once the partition of the current frame is delivered, it remains fixed. We use a similar hill-climbing algorithm as in SEEDS for stills, that runs in real-time. See Fig. 3.8 for an overview of the algorithm. We now describe the differences between the hill-climbing optimization in SEEDS in still and in Videos, which are the pixel and block-level updates, creating and terminating video superpixels, and the propagation of the video superpixels over time.

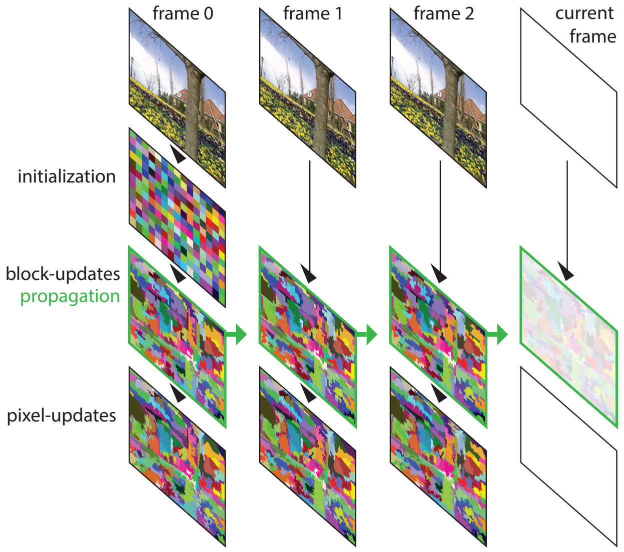


***Figure 3.8**: Overview of the Video SEEDS algorithm: The superpixel labels are propagated at an intermediary step of block-level updates. The result is fine-tuned for each frame individually.*

Pixel and block-level updates

The hill-climbing optimization proposes block-level updates to pixel-level updates, as described in SEEDS for stills, with the difference that takes into account the temporal information. Let $\mathcal{B}^t_n$ be a block of pixels of the current frame that belongs to the superpixel n, *i.e.* $\mathcal{B}^t_n \subset \mathcal{A}^t_n \subset \mathcal{A}^{t:0}_n$. To evaluate whether exchanging the block $\mathcal{B}^t_n$ from superpixel n to m increases the objective function, we can use one histogram intersection computation, rather than evaluating the complete energy function. This is $\mathbf{int}(c_{\mathcal{B}^t_n}, c_{\mathcal{A}^{t:0}_m}) \geq \mathbf{int}(c_{\mathcal{B}^t_n}, c_{\mathcal{A}^{t:0}_n \setminus \mathcal{B}^t_n})$. Thus, if the intersection of $\mathcal{B}^t_n$ to the video superpixel $\mathcal{A}^{t:0}_m$ is higher than the intersection to the superpixel it currently belongs to, the exchange is accepted, otherwise it is discarded, similarly to the optimization of SEEDS for stills.

The inequality of intersection distances, maximizes the energy under the assumptions that $|\mathcal{A}^{t:0}_m| \approx |\mathcal{A}^{t:0}_n|$, $|\mathcal{B}^t_n| \ll |\mathcal{A}^{t:0}_n|$, where $|\cdot|$ is the cardinality of the set. Also, it assumes that the histogram of $\mathcal{B}^t_n$ is concentrated in a single bin. The first assumption is that video superpixels are of similar size and that the blocks are much smaller than the video superpixels. This holds most of the time, since superpixels indeed tend to be of the same size, and the blocks are defined to be at most one fourth of a superpixel in a frame, and hence, are much smaller than superpixels extending on multiple frames in the video. The second assumption is that the block of pixels have homogeneous color histograms.

Creating and terminating video superpixels

According to the superpixel rate, some frames are selected to terminate and create superpixels. When a frame is selected, we first terminate a superpixel, and then we create a new one. To this aim, we introduce inequalities with intersection distances. They allow to evaluate which termination and creation of superpixels yield higher energy using efficient intersection distances.

When a superpixel is terminated, its pixels at frame t are incorporated to a neighbor superpixel. Let $\mathcal{A}_n^t \subset \mathcal{A}_n^{t:0}$ and $\mathcal{A}_m^t \subset \mathcal{A}_m^{t:0}$ be two candidates of superpixels to terminate at frame t. Let $\mathcal{A}_p^{t:0}$ and $\mathcal{A}_q^{t:0}$ be the superpixel candidate to incorporate $\mathcal{A}_n^t$ and $\mathcal{A}_m^t$, respectively. The superpixel with larger intersection with its neighbor is the one selected to terminate, *i.e.* $\mathbf{int}(c_{\mathcal{A}_n^t}, c_{\mathcal{A}_p^{t:0}}) \geq \mathbf{int}(c_{\mathcal{A}_m^t}, c_{\mathcal{A}_q^{t:0}})$. We terminate the superpixel with higher intersection similarity to its neighbor among all superpixels in the frame. The above equation leads to the highest energy state, under the assumptions that $|\mathcal{A}_p^{t:0}| \approx |\mathcal{A}_q^{t:0}|$, $|\mathcal{A}_n^t| \ll |\mathcal{A}_p^{t:0}|, |\mathcal{A}_m^t| \ll |\mathcal{A}_q^{t:0}|$, and that both $\mathcal{A}_n^t$ and $\mathcal{A}_m^t$ have histograms concentrated into one bin. These are similar to the assumptions for the previous inequalities with intersection distances. Additionally, it is also assumed that $c_{\mathcal{A}_n^{t:0}} \approx c_{\mathcal{A}_n^{(t-1):0}}$ and $c_{\mathcal{A}_m^{t:0}} \approx c_{\mathcal{A}_m^{(t-1):0}}$. This is, the color histogram of the temporal superpixel remains approximately the same including and excluding the pixels at the current frame. This holds most of the time, given the fact that $|\mathcal{A}_n^t| \ll |\mathcal{A}_n^{t:0}|$.

If a superpixel is terminated, a new one should be created to fulfill the constraint of number of superpixels per frame. The candidates to form a new superpixel are blocks of pixels that belong to an existing video superpixel. Let $\mathcal{B}_n^t \subset \mathcal{A}_n^{t:0}$ and $\mathcal{B}_m^t \subset \mathcal{A}_m^{t:0}$ be blocks of superpixels candidates to create a new superpixel. We select the block of pixels whose histogram minimally intersects with its current superpixel. *I.e.*, $\mathbf{int}(c_{\mathcal{B}_m^t}, c_{\mathcal{A}_m^{t:0}\setminus\mathcal{B}_m^t}) \leq \mathbf{int}(c_{\mathcal{B}_n^t}, c_{\mathcal{A}_n^{t:0}\setminus\mathcal{B}_n^t})$. We select the block of pixels with minimum intersection in the frame. This yields the highest energy, assuming that $|\mathcal{A}_m^{t:0}| \approx |\mathcal{A}_n^{t:0}|$, $|\mathcal{B}_n^t| \ll |\mathcal{A}_n^{t:0}|, |\mathcal{B}_m^t| \ll |\mathcal{A}_m^{t:0}|$, and that both $\mathcal{B}_n^t$ and $\mathcal{B}_m^t$ have histograms concentrated into one bin.

Propagation

In the first frame of the video, the superpixels are initialized along a grid using the hierarchy of blocks. In the subsequent frames, the block hierarchy is exploited to initialize the superpixels. Rather than re-initializing along a grid, the new frame is initialized by taking an intermediary block-level result from the previous frame (see Figure 3.8). Like this, the superpixel structure can be propagated from the previous frame while discarding small details. In principle, the algorithm can run for an infinitely long video, since it generates the partition online, and in memory we only need the histograms of the video superpixels that propagate to the current frame.

3.4 Spectral SEEDS

In this Section, we introduce a method for learning the descriptors used in SEEDS, such that color and texture are taken into account. We first revisit Spectral Clustering, because the descriptors are learned using Spectral Clustering techniques, and we also introduce an equivalent notation of SEEDS that is convenient to incorporate the learned descriptors with Spectral Clustering, as we will show later in this Section. We coin Spectral SEEDS the method with SEEDS and learned descriptors.

3.4.1 Preliminaries

Spectral Clustering

We revisit Spectral Clustering (SC), which we use in Spectral SEEDS. For notation convenience we denote the set of column vectors of a matrix $\mathbf{M} \in \mathbb{R}^{P \times Q}$ as $\{\mathbf{m}_q\}_Q$, where $q \in \{1, \ldots, Q\}$. The sub-index in the set denotes its cardinality, *i.e.* $|\{\mathbf{m}_q\}_Q| = Q$. Also, m_{qj} is the j entry of the vector $\mathbf{m}_q \in \mathbb{R}^P$.

Spectral Clustering (SC) is a popular approach to clustering, initially introduced for graph partitioning problems [Donath & Hoffman, 1973; Fiedler, 1973]. SC received a lot of attention for region extraction during the last decade, *e.g.* [Malik et al., 2001; Maire et al., 2011; Shi & Malik, 2000; Yu & Shi, 2003].

Let $\{\mathbf{v}_i\}_N$, in which $\mathbf{v}_i \in \mathbb{R}^D$, be the set of N vectors to be clustered into K groups. We use the $N \times N$ similarity matrix, denoted as $\mathbf{A}$, to characterize the similarities between all pairs of vectors in $\{\mathbf{v}_i\}_N$. The matrix $\mathbf{A}$ is normalized to obtain the Laplacian matrix, which we call $\mathbf{L}$. There are many ways we can define $\mathbf{L}$, *cf.* [Von Luxburg, 2007]. A common way is to use

$$\begin{aligned} \mathbf{L} &= \mathbf{D} - \mathbf{A}, \\ \text{where } \mathbf{D} &= \text{diag}(\mathbf{A}\mathbf{1}_N), \end{aligned} \tag{3.9}$$

diag$(\mathbf{s})$ is the diagonal matrix with $s_1, s_2, \ldots$ as its diagonal entries, and $\mathbf{1}_N$ is a column vector with N ones. We use the $N \times K$ indicator matrix $\mathbf{E}$ to denote the assignment of the N input vectors to one of the K clusters. Each row of $\mathbf{E}$ has one entry set to 1 to indicate the assignment, and the rest to 0. Thus, $\mathbf{E}$ fulfills the following constraints: $\mathbf{E} \in \{0,1\}^{N\times K}$ and $\mathbf{E}\mathbf{1}_K = \mathbf{1}_N$. The *multiclass* clustering criterion optimizes [Yu & Shi, 2003; Bach & Jordan, 2006]:

$$\begin{aligned} \min_{\mathbf{E}} \quad & tr(\mathbf{E}^t\mathbf{L}\mathbf{E}) \\ s.t. \quad & \mathbf{E} \in \{0,1\}^{N\times K} \\ & \mathbf{E}\mathbf{1}_K = \mathbf{1}_N, \end{aligned} \tag{3.10}$$

where the super-index t is the transpose operator, and $tr(\,\cdot\,)$ is the trace of a matrix.

SC relaxes the problem in Equation (3.10) to a continuous optimization by dropping the constraint that forces $\mathbf{E}$ to be a binary matrix. The result of the spectral relaxation, which we denote as $\mathbf{Y}$, can take real values, but the constraint that the columns of $\mathbf{Y}$ form an orthonormal basis, still applies. Also, it is common to introduce some user-defined weights for each input vector, denoted by π_i, which form a diagonal matrix of weights $\vec{\mathbf{\Pi}} = diag(\pi_1, \pi_2, \ldots, \pi_N)$. The constraints of the relaxed problem become $\vec{\mathbf{\Pi}}\mathbf{Y}\mathbf{1}_K = \mathbf{1}_N$ and $\mathbf{Y}^t\vec{\mathbf{\Pi}}\mathbf{Y} =$

Algorithm 2: SC

```
Input: ${\mathbf{v}_i}_N$, $K$
Output: $\mathbf{E}$
1. Laplacian:
$\mathbf{A} \leftarrow$ Similarity from $\{\mathbf{v}_i\}_N$
$\mathbf{L} \leftarrow$ Laplacian from $\mathbf{A}$
2. Spectral Relaxation:
$\{\mathbf{y}_i\}_N \leftarrow$ Eigenvectors($\mathbf{L}$)
3. Rounding:
$\mathbf{E} \leftarrow$ Rounding($\{\mathbf{y}_i\}_N, K$)
```

$\mathbf{I}_K$ where $\mathbf{I}_K$ is the $K \times K$ identity matrix. Thus, the clustering objective in Equation (3.10) becomes

$$\begin{aligned} \min_{\mathbf{Y}} \quad & tr(\mathbf{Y}^t\mathbf{L}\mathbf{Y}) \\ s.t. \quad & \mathbf{Y}^t\vec{\mathbf{\Pi}}\mathbf{Y} = \mathbf{I}_K \\ & \vec{\mathbf{\Pi}}\mathbf{Y}\mathbf{1}_K = \mathbf{1}_N. \end{aligned} \tag{3.11}$$

This relaxed optimization problem can be solved with a generalized eigenvector problem, *i.e.* $\mathbf{L}\mathbf{y} = \lambda\vec{\mathbf{\Pi}}\mathbf{y}$, where $\mathbf{y}$ is a column of $\mathbf{Y}^t$ and λ an eigenvalue. The name "spectral" comes from the fact that SC is an eigenvector problem.

Finally, a rounding heuristic is used to recover the solution, $\mathbf{E}$, of the original discrete problem from the eigenvalues, $\mathbf{Y}$; *cf.* [Von Luxburg, 2007]. In Algorithm 2 there is a summary of the SC algorithm.

Spectral Clustering in the Primal Form

The formulation in the primal form of SC was introduced by Rahimi & Recht [2004], and further developed by Zhang & Jordan [2008]. It establishes a connection between the representation of the input vectors, $\{\mathbf{v}_i\}_N$, and the relaxed solution to SC. Recall that $\mathbf{V}$ is the matrix which columns are the vectors to be clustered, *i.e.* $\{\mathbf{v}_i \in \mathbb{R}^D\}_N$, and let $\mathbf{S}$ be a $D \times K$ matrix which corresponds to the variables to optimize in the primal form of SC. $\mathbf{S}$ is related to $\mathbf{Y}$ through the input data, *i.e.* $\mathbf{Y} = \mathbf{V}^t\mathbf{S}$. The columns of $\mathbf{S}$ can be seen as hyperplanes that separate the different clusters, and the entries of $\mathbf{Y}$ the signed distance to

```
Algorithm 3: SEEDS
Input: ${\mathbf{v}_i}_N, K$
Output: $p \in \mathcal{P}$
1. Adaptative binning:
$\{\mathcal{H}_j\}_B \leftarrow$ center color bins
2. Assignment:
$y_{ij} \leftarrow$ similarity $(\mathbf{v}_i, \mathcal{H}_j)\ \forall i, j$
$\{\mathbf{u}_i\}_N \leftarrow \max_j y_{ij}\ \forall i, j$
3. Hill-climbing:
$p \leftarrow$ optimize Equation (3.12)
```

the hyperplanes [Zhang & Jordan, 2008]. The optimization is the primal form can be derived by replacing $\mathbf{Y} = \mathbf{V}^t\mathbf{S}$ in Equation (3.11).

Observe that depending on the number of clusters, it may be advantageous to do the optimization in the primal form of SC, which involves $D \times K$ variables, rather than in the original form of SC, which scales quadratically with the number of input vectors. Next, we introduce a new notation of SEEDS that allows for a relation to Spectral Clustering. Then, in the sequel, we show that the primal formulation of SC can be used to achieve speed-ups in the computation of SC.

New notation of SEEDS Superpixels

We re-write the notation of SEEDS superpixels, which will be convenient for our purposes, as we will see later in this Chapter. Recall that $\mathcal{H}_j$ defines the colors in a bin that form the histogram, and B is the number of bins of the histogram. Each pixel of the image is assigned to one bin of the histogram. Let $\mathbf{u}_i$ be the vector that encodes the assignment of the pixel descriptor or pixel color, $\mathbf{v}_i$, to the histogram bin. Thus, u_{ij} is equal to $\delta(\mathbf{v}_i \in \mathcal{H}_j)$, in which $\delta(x) = 1$ if x is true, and 0 otherwise. Note that $\mathbf{u}_i$ is a vector of 0's and one 1, that indicates the bin to which the pixel descriptor has been assigned.

Recall that $\mathcal{A}_k$ is the set of pixels inside a superpixel k. The objective function of SEEDS with the new notation is

$$\arg\max_{\{\mathcal{A}_k\}_K} \sum_{k<K} \sum_{j<B} \left(\frac{1}{Z} \sum_{i \in \mathcal{A}_k} u_{ij}\right)^2, \tag{3.12}$$

where Z is a normalization factor of the histogram. Observe that Equation (3.12) is equivalent to Equation (3.2). Note that $(\frac{1}{Z}\sum_{i\in\mathcal{A}_k} u_{ij})^2$ is maximum when all $\{\mathbf{u}_i\}_{i\in\mathcal{A}_k}$ are assigned to the same color bin. In Algorithm 3 there is an overview of the SEEDS algorithm using this notation.

3.4.2 Spectral SEEDS Algorithm

In the following, we first introduce the algorithm of Spectral SEEDS, and then, show how it is related to SEEDS and to SC. After that, we analyze its computational cost.

Recall that the original version of SEEDS represents the pixels using the Lab color space. We replace this color descriptor by 3×3 square patches extracted around each pixel, in RGB. This allows to better capture the color and the texture, and avoids the color conversion of SEEDS, which takes about half of the computational time of SEEDS, as we show in the experiments section. We could use bigger patch sizes, but were more expensive to compute and we did not observe any improvement in the results. We denote as $\mathbf{v}_i$ the descriptor vector for a single pixel, which is of length 27, *i.e.* the RGB value of the 3×3 patch around the pixel.

Algorithm

Recall that $\mathbf{Y}$ is the result of the the spectral relaxation of SC. $\mathbf{Y}$ gives an idea of how much each pixel belongs to the clusters inferred by SC. Also, recall that in SEEDS, $\mathbf{Y}$ is defined as the similarity between the pixel color and the color bins of the histogram, $\{\mathcal{H}_j\}_B$. We propose to compute $\mathbf{Y}$ as in SC, and then, continue as in SEEDS. We use SC to compute $\mathbf{Y}$ because it infers clusters of the input descriptors that we can use as the bins to build the histograms for SEEDS. In this way, we use an spectral embedding for extracting representations that may be richer than just pre-fixed color histogram bins, and use SEEDS for an efficient computation of the final superpixels. We could use another clustering technique different from SC to compute $\mathbf{Y}$. We show in the sequel that SC is convenient in our case. Spectral SEEDS is the following 3 step-algorithm:

1. *Spectral Embedding.* It consists of extracting the spectral relaxation, $\mathbf{Y}$, from all the pixel descriptors of the image, $\{\mathbf{v}_i\}_N$. SC extracts clus-

ters from $\{\mathbf{v}_i\}_N$, which in the following steps, are used as the bins for building the histogram for SEEDS. Note that these bins are built from the representation used in $\mathbf{v}_i$, which is more general than the Lab color value of a pixel used in SEEDS. $\mathbf{y}_i$ encodes how much $\mathbf{v}_i$ belongs to the color/texture bin extracted with SC.

2. *Unconstrained Rounding.* The vectors of the spectral embedding, $\mathbf{y}_i$, are rounded, such that each pixel $\mathbf{v}_i$ is assigned to only one bin, obtaining $\mathbf{u}_i$ in the same way as in SEEDS (*i.e.* $u_{ij} = 1$ when $\mathbf{v}_i$ is assigned to bin j, and 0 otherwise). The rounded matrix for all pixels, $\mathbf{U}$, is called unconstrained rounding because it does not constrain the superpixels to be continuous blobs of pixels.

3. *Hill-Climbing from SEEDS for Constrained Rounding.* We run the hill-climbing optimization from SEEDS, taking as input the rounded assignments, $\mathbf{U}$. The hill-climbing enforces homogeneous superpixels by maximizing the objective function of SEEDS in Equation (3.12). The hill-climbing of SEEDS guarantees that the resulting superpixels are continuous blobs, *i.e.* the final partitioning is $p \in \mathcal{P}$.

Since the algorithm of Spectral SEEDS is built from SC and SEEDS, it can be analyzed from both SC and SEEDS perspective.

From SC point of view, there is a spectral embedding, $\mathbf{Y}$, and then, the final rounding to obtain a valid superpixel partitioning, $p \in \mathcal{P}$, is done with SEEDS. Thus, SEEDS is used as the heuristic to recover the constrained solution of the discrete SC problem. In the literature, many sophisticated heuristics have been proposed, *e.g.* [Malik et al., 2001; Yu & Shi, 2003; Bach & Jordan, 2006; Zhang & Jordan, 2008]. Note that superpixel extraction is simpler than other more general clustering algorithms, and a simple heuristics such as SEEDS can achieve good performance for superpixel extraction, but may not be the case for any clustering problem.

In SEEDS, the spectral embedding, $\mathbf{Y}$, corresponds to the similarity between the pixel color and the colors of the histogram bins. In Spectral SEEDS, the pixel representation may be adapted to any descriptor using spectral embeddings. In the experiments we show that this allows to achieve higher performance.

Computational Cost Analysis

The bottleneck of the Spectral SEEDS algorithm is computing the spectral relaxation, $\mathbf{Y}$, of Step 1. The spectral relaxation in SC is usually obtained from the Laplacian matrix built from all the pixels of the image, which increases quadratically with the number of pixels, and computing the eigenvalues of this matrix may be very computationally expensive. In the following, we focus on speeding-up this step of the algorithm, and reduce it to negligible cost. In Subsection 3.4.3 we introduce a learning for the spectral embedding to fix it to constant values, and in Subsection 3.4.4 how to make more efficient the similarity computation in the spectral embedding using Sparse Quantization. The computation of $\mathbf{Y}$ could be done by other clustering techniques different from SC as well, but the optimisations that we present in the following are particular for SC.

Note that Steps 2 and 3 of the Spectral SEEDS algorithm are the same as in the original SEEDS algorithm. The assignment to obtain the unconstrained rounding, $\mathbf{U}$, is a trivial computation that involves the selection of the the maximum value for each column vector of $\mathbf{Y}$, which has negligible cost. The hill-climbing optimization is very efficient to compute, as reported in SEEDS [Van den Bergh et al., 2012], and in the experiments section.

3.4.3 Learning the Spectral Embedding

We now introduce a method for learning the spectral embedding, $\mathbf{Y}$, using the primal form of SC. This yields significant computational savings. In the following, we first introduce the primal form of SC for Spectral SEEDS, and then show how to reuse the same parameters for all images, and avoid computing them for each new image.

Primal Form of the Spectral Embedding

Computing the spectral relaxation, $\mathbf{Y}$, for superpixel extraction, may be prohibitive when the number of pixels is large. This is because $\mathbf{Y}$ are the eigenvalues of the Laplacian matrix, which increases quadratically with the number of pixels. A way around the Laplacian matrix is the primal form of SC, whose formulation does not depend on the Laplacian matrix. Recall that the variables to optimize in the primal form of SC, $\mathbf{S}$, are the hyperplanes that separate the clusters (see Subsection 3.4.1). The amount of variables to optimize is the number of clusters times the length of the pixel descriptor. Note that this is orders of magnitude smaller than in the original form of SC, which are the number of pixels times the number of clusters.

Another advantage of using the primal form of SC instead of the original form of SC, is that $\mathbf{S}$ can be easily computed from a subset of pixels of the image. Note that in the primal of SC, the spectral embedding, $\mathbf{Y}$, is obtained by projecting all descriptors of the pixels, $\mathbf{v}_i$, to $\mathbf{S}$, *i.e.* $\mathbf{Y} = \mathbf{V}^t\mathbf{S}$. In the original form of SC, computing $\mathbf{Y}$ using a subset of pixels is not straightforward, because in the original form of SC $\mathbf{Y}$ is directly obtained for all pixels. In an image, the colors and textures are highly redundant, and hence, it is reasonable to expect that a subset of pixels is representative enough to compute $\mathbf{S}$ for all image.

Learning the Parameters of the Spectral Embedding

To further speed-up the computation of $\mathbf{Y}$, we learn a fixed set of parameters $\mathbf{S}$ for all images, and avoid recomputing them for every image. We compute $\mathbf{S}$ *a priori* from a collection of pixel descriptors, $\{\mathbf{v}_i\}_N$, randomly extracted from a training set of images. Then, these parameters $\mathbf{S}$ can be used for any image since they are calculated from a set of representative pixels. In the experiments we show that this approach achieves similar accuracy than adapting $\mathbf{S}$ to each new image.

To learn $\mathbf{S}$ for all images, we optimize $\mathbf{S}$ with the original form of SC, rather than in the primal. This is because we found that it is enough to randomly select about 5000 pixels among all images, which makes the computation in the original form of SC feasible. Note that in this case is more adequate the original form of SC, because only 5000 pixels are involved, while SC for one image involves hundreds of thousands. Thus, we learn the primal parameters of SC by first calculating the Laplacian matrix from the randomly selected pixels from a training set of images. For computing the Laplacian we use the weights $\vec{\Pi} = \mathbf{D}$, see Equation (3.9), which is common in the literature for image region extraction, *e.g.* [Malik et al., 2001]. We calculate $\mathbf{Y}$ from the Laplacian, and we obtain $\mathbf{S}$ using its relation to $\mathbf{Y}$, *i.e.* $\mathbf{S} = (\mathbf{V}^t)^{+}\mathbf{Y}$, where $\mathbf{V}$ is a matrix which columns correspond to the randomly selected pixels, and

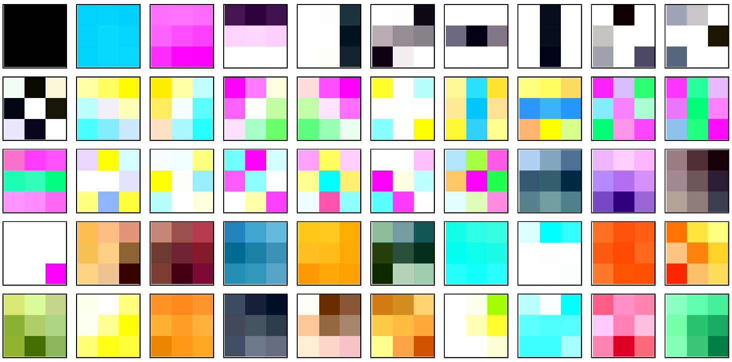


Figure 3.9: Learned patterns for Spectral SEEDS. *First* 50 *filters with highest corresponding eigenvalues.*

$+$ is the Moore-Penrose operator [Rahimi & Recht, 2004; Zhang & Jordan, 2008].

Note that we can choose the number of column vectors in $\mathbf{S}$, which corresponds to B (number of histogram bins in SEEDS), and we analyze its value in the experiments section. In Figure 3.9, we show 50 vectors from $\{\mathbf{s}_j\}_B$ with the highest corresponding eigenvalues from SC, learned from the training set of BSD Dataset. We can see that $\{\mathbf{s}_j\}_B$ resemble color filters of 3×3 pixels. From now on we refer to $\{\mathbf{s}_j\}_B$ as filters.

To calculate $\mathbf{Y}$ for a new image, we simply project the pixel descriptors of the new image to the learned filters, *i.e.* $y_{ij} = \mathbf{v}_i^t \mathbf{s}_j$, or equivalently $\mathbf{Y} = \mathbf{V}^t\mathbf{S}$. The resulting $\mathbf{Y}$ is highly sparse, because $\mathbf{S}$ has been learned in general for any image. Observe that the bottleneck of the algorithm becomes the projection of the pixels to the primal parameters of the SC, $\mathbf{S}$. In the next Subsection, we introduce a method to make the projection more efficient.

3.4.4 Fast Embedding with Sparse Quantization

In the previous Subsection, we showed that $\mathbf{Y}$ can be computed by $\mathbf{Y} = \mathbf{V}^t\mathbf{S}$, where $\mathbf{S}$ has been learned from all the dataset. Although this significantly reduces the computational cost of SC, computing the similarities between the

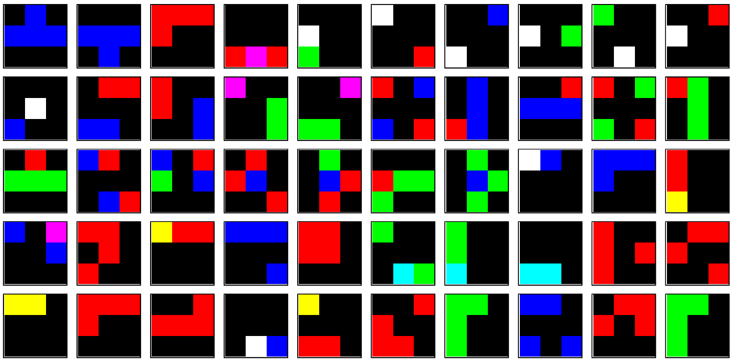


Figure 3.10: Learned patterns for Spectral SEEDS with SQ. *First* 50 *filters with highest corresponding eigenvalues of figure 3.9 quantised with SQ in* $\mathbb{B}_4^{27}$.

pixel descriptors and the leaned filters, $\mathbf{V}^t\mathbf{S}$, has a higher computational cost than computing similarities when using a color histogram. We propose to use Sparse Quantization (SQ) to binarize the descriptors and the filters, and use fast binary similarity computations. SQ is a quantization to the set of k-sparse vectors. SQ is fast to compute, does not increase the dimensionality, and results in binary representations that allow for fast computations of similarities. We refer to Section 4.2 for a detailed explanation of SQ.

Sparse Quantization for Efficient Computation of $\mathbf{V}^t\mathbf{S}$

Let $\mathbb{B}_k^m = \{0,1\}_k^m$ be the space of *binary* k-sparse vectors, with k ones and $(m-k)$ zeros. $\mathbb{B}_k^m$ is composed of a finite amount of elements, in particular, its cardinality is $|\mathbb{B}_k^m| = \binom{m}{k}$. Recall that $\{\mathbf{s}_j\}_B$ is the set of column vectors of $\mathbf{S}$ which have the same length as the pixel descriptors. We apply binary SQ to both the 3×3 patches, $\{\mathbf{v}_i\}_N$, and the learned filters, $\{\mathbf{s}_j\}_B$, in order to speed up the calculation of the similarity measure. Thus, instead of computing $\mathbf{v}_i^t\mathbf{s}_j$, we use the binary SQ version of $\mathbf{v}_i$ and $\mathbf{s}_j$, and exploit fast binary similarities. Recall that the length of the pixel descriptor vectors is 27. In the results, we found by cross-validation that SQ to the set of 4-sparse binary vectors, yields comparable performance as using the non-quantized version of the filters. In Figure 3.10, we show the SQ version of the learned filters showed in Figure 3.9.

The set of 4-sparse binary vectors is of relatively small cardinality, *i.e.* $|\mathbb{B}_4^{27}| = \binom{27}{4} = 17550$. In order to compute $\mathbf{v}_i^t \mathbf{s}_j$, we can store a look-up table in memory for the $\binom{27}{4}$ possible quantizations of $\mathbf{v}_i$. The look-up table, can relate the quantized $\mathbf{v}_i$ with $\mathbf{y}_i$ for each learned $\mathbf{s}_j$. To further speed-up the pipeline, note that we can directly relate $\mathbf{v}_i$ to $\mathbf{u}_i$ (Step 2 in the Spectral SEEDS algorithm in Section 3.4), instead of first relating $\mathbf{v}_i$ to $\mathbf{y}_i$ and then computing $\mathbf{u}_i$.

In practice, the look-up table with input the SQ of $\mathbf{v}_i$ and output which entry of the vectors $\mathbf{u}_i$ is activated, can be implemented with a list of arrays of pointers. The first array of pointers is of length equal to the feature length, 27, and each of the position of the array represents the index of the first feature dimension activated in the binary vector. Each position of the first array points to another array of again dimension 27, which represents the second dimension of the binary vector, and so on. For a set of binary vectors in $\mathbb{B}_4^{27}$, there are 3 layers of stack arrays of pointers, as described, and the fourth layer of arrays contains the value of the maximum of $\mathbf{y}_i$, *i.e.* the activated entry of $\mathbf{u}_i$. Note that there is no need to fill all the arrays. For instance, if the index selected in the previous array was the position 10, the following indexes can only be between the 11 and 27. This allows for great savings of memory, since rather than having a table of 27^4 we have $\binom{27}{4}$.

In summary, the calculation of the $\mathbf{V}^t\mathbf{S}$ can be avoided as well as Step 2 of the Spectral SEEDS algorithm by using a look-up table. This has a computational cost of a selection of the 4 higher elements in $\mathbf{v}_i$, and one memory access to the look-up table, which is faster than the Lab color conversion, as we show in the results. In Algorithm 4 we show the learning of Spectral SEEDS, and in Algorithm 5 the testing. In Figure 3.11, we show the results of convolving the image with some of the filters, $\{\mathbf{s}_j\}_B$, with and without using SQ. Finally, in Figure 3.12, we show the differences of $\{\mathbf{u}_i\}_N$ with and without SQ, using the same filters as in Figure 3.11. We can see that when using SQ, more details of the image are thrown away due to the quantization effects, and region extraction may even benefit from that.

3.5 Experiments

In this Section, we report results of SEEDS and Spectral SEEDS algorithms on the BSD500 [Martin et al., 2001] dataset. The BSD500 dataset consist on 481×321 images, and it is divided into 3 splits, in which there are 200

Algorithm 4: Learning Spectral SEEDS

```
**Input**: $\{\mathbf{v}_i\}_N$ from all train. set
**Output**: $\hat{\mathbf{S}}$, Look-up tables
1. Spectral Relaxation:
$\mathbf{L} \leftarrow$ Laplacian from $\{\mathbf{v}_i\}_N$
$\{\mathbf{y}_i\}_N \leftarrow$ Eigenvectors of $\mathbf{L}$
2. Filters:
$\mathbf{S} = (\mathbf{V}^t)^+\mathbf{Y}$
3. emphSQ of filters:
$\hat{\mathbf{S}} \leftarrow$ SQ of $\{\mathbf{s}_j\}_B$
Prepare Look-up tables from $\hat{\mathbf{S}}$
```

Algorithm 5: Testing Spectral SEEDS

```
**Input**: $\{\mathbf{v}_i\}_N$, $K$
**Output**: $p \in \mathcal{P}$
1. SQ of pixel descriptors:
$\{\hat{\mathbf{v}}_i\}_N \leftarrow$ SQ of $\{\mathbf{v}_i\}_N$
2. Assignment:
$u_{ij} \leftarrow$ Look-up table $(\hat{\mathbf{v}}_i, \hat{\mathbf{s}}_j)$ $\forall i, j$
3.Hill-climbing:
$p \leftarrow$ optimize Equation (3.12)
```

images for training, 100 for validation, and 200 for testing. After describing the evaluation metrics, we first evaluate the parameters of Spectral SEEDS, and then we compare it to state-of-the-art superpixel algorithms.

3.5.1 Evaluation Metrics

We compute the standard metrics used to evaluate the performance of superpixel algorithms. These are the Undersegmentation Error (UE), the Boundary Recall (BR) and the Achievable Segmentation Accuracy (ASA).

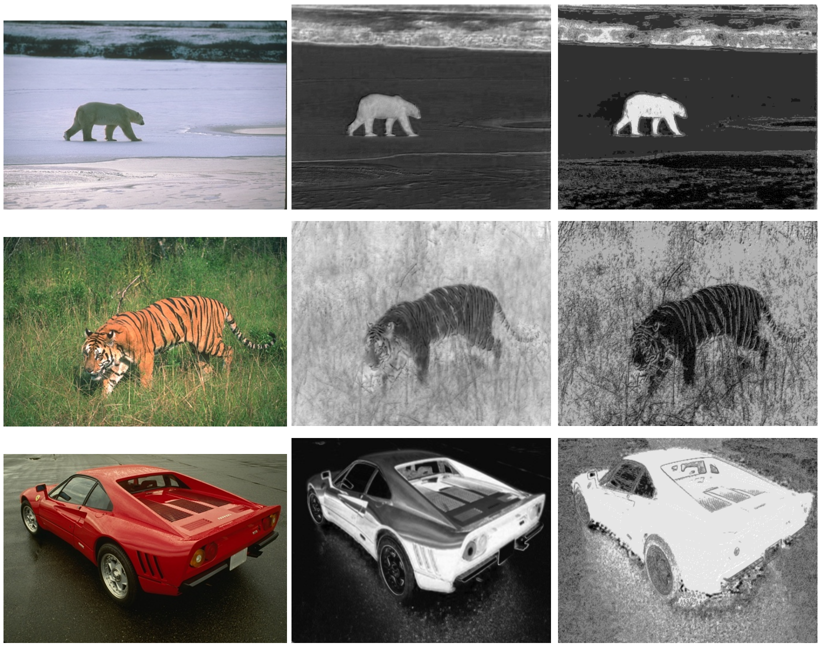
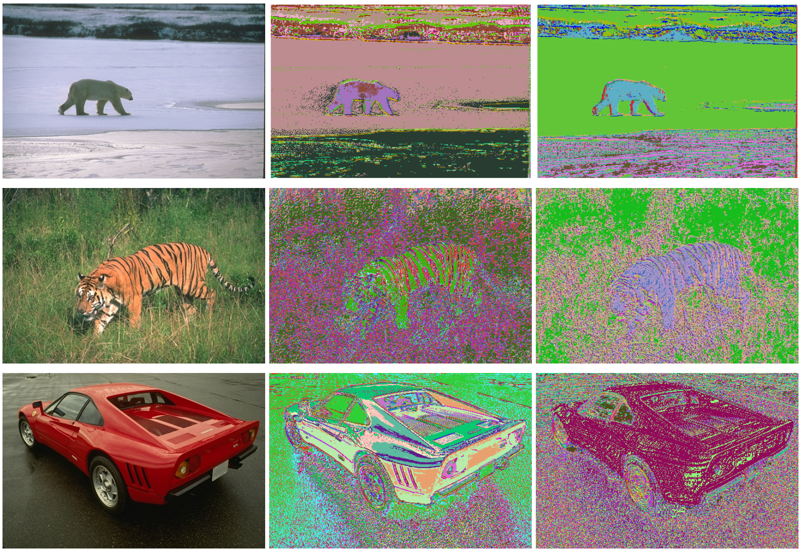


Figure 3.11: Spectral SEEDS. *(a) Original image. (b) Response to a learned filter. A different filter is used for each image. (c) is analogous to (b), but using SQ.*

Undersegmentation Error (UE) measures that a superpixel should not overlap more than one object. The standard formulation is

$$UE(p) = \frac{\sum_i \sum_{k:p_k \cap g_i \neq \emptyset} |s_k - g_i|}{\sum_i |g_i|}, \tag{3.13}$$

where g_i are the ground truth segments, p_k the output segments of the algorithm, and $|a|$ indicates the size of the segment. The borders of s_k are removed from the labeling before computing the UE.



(a) (b) (c)

***Figure 3.12**: Spectral SEEDS. (a) Original image. (b) Unconstrained rounding from the response to a learned filter. The filters are the same used in Fig. 3.11. The colors (randomly selected) represent a different dominant filter. (c) is analogous to (b), but using SQ.*

Boundary Recall (BR) evaluates the percentage of borders from the groundtruth that coincide with the borders of the superpixels. It is formulated as

$$BR(p) = \frac{\sum_{i \in \mathcal{B}(g)} \mathrm{I}[min_{j \in \mathcal{B}(p)} \|i - j\| < \epsilon]}{|\mathcal{B}(g)|}, \tag{3.14}$$

where $\mathcal{B}(g)$ and $\mathcal{B}(p)$ are the union sets of superpixel boundaries of the groundtruth and the computed superpixels, respectively. The function I[.], is an indicator function that returns 1 if a boundary pixel of the output superpixel is within ϵ pixels of tolerance of the groundtruth boundaries. We set $\epsilon = 2$, as in [Liu et al., 2011b; Van den Bergh et al., 2012].

Achievable Segmentation Accuracy (ASA) is an upper bound measure. It gives the maximum performance when taking superpixels as units for object segmentation, and is computed as

$$ASA(p) = \frac{\sum_k \max_i |p_k \cap g_i|}{\sum_i |g_i|}, \tag{3.15}$$

where the superpixels are labeled with the label of the groundtruth segment which has the largest overlap.

3.5.2 Number of filters

We report results of Spectral SEEDS learning the filters in the training set, with and without SQ (denoted in the plots as Learned filters and Learned filters + SQ, respectively). We learn the filters by randomly selecting 5000 pixels from all images. We also tried selecting more pixels, but we did not observe a significant improvement in the performance. We compare this to Spectral SEEDS computing the filters at each new image from a subset of 1000 pixels of the image (denoted as Adaptative filters), and to SEEDS using Lab color space (denoted as Lab space (SEEDS)). In all cases, the parameters of the hill-climbing optimization are the same.

In Figure 3.13, we show the performance, on the validation set of BSD500, of Spectral SEEDS for different number of filters, when using 400 superpixels, 200, 100 and 50 superpixels, respectively. Note that when using SQ, we need less filters than without for having the same accuracy. This is because one filter with SQ may capture the effect of multiple filters without SQ, due to the coarsening effect of SQ. Also, observe that the adapted set needs less filters to achieve the same performance as the learned set of filters. This is in accordance with what we observe when using the learned filters, that only a small subset of them are activated for the same image. Finally, note that with Spectral SEEDS, there is no need of color space conversions, but it achieves similar levels of performance as SEEDS with the Lab color space.

3.5.3 Computational Cost

We use one 2.8 GHz i7 CPU. In SEEDS, conversion to Lab space takes around 0.06 seconds, which is approximately the same as the cost of extracting the

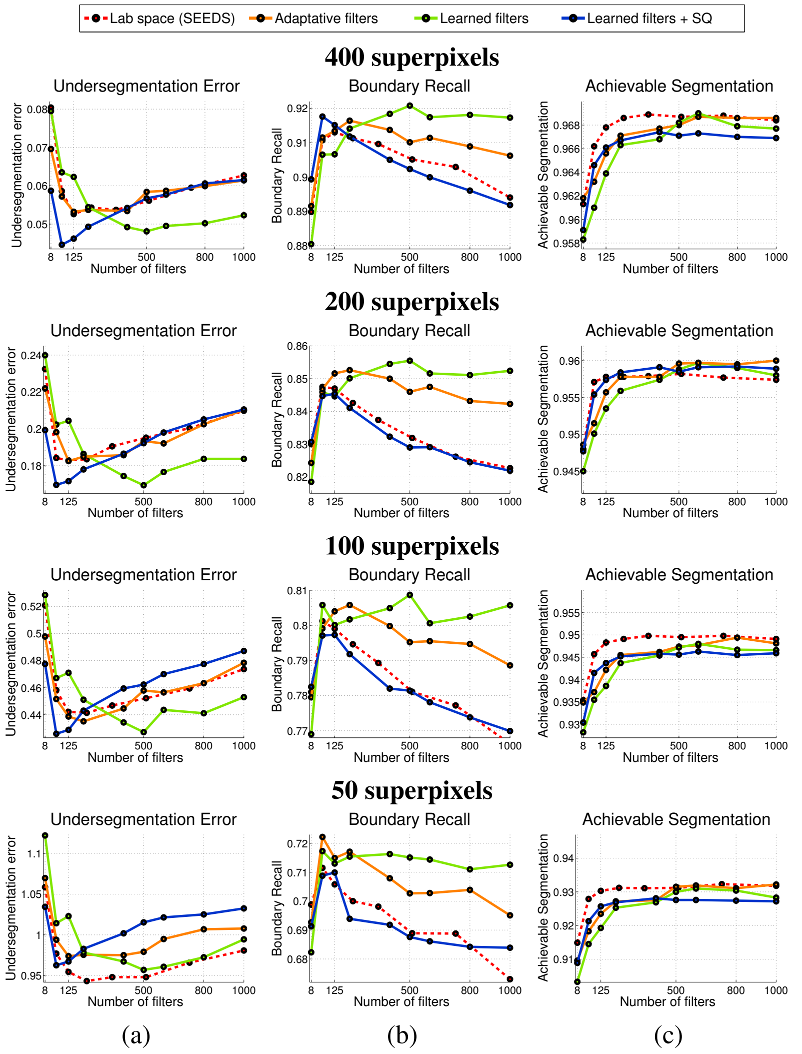
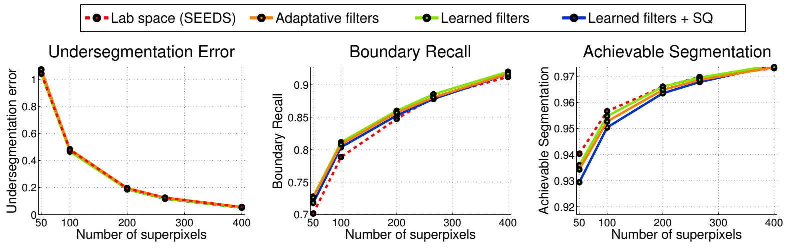
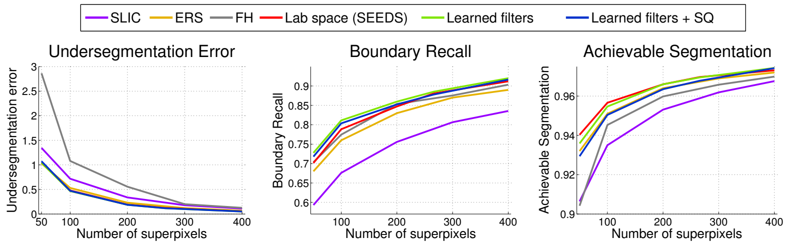


Figure 3.13: Results on BSD500 Dataset in the validation set. *We compare Spectral SEEDS with and without SQ, and learning the filters. Also, we compare it with normal SEEDS. (a) Undersegmentation error, (b) Boundary Recall, (d) Achievable segmentation.*



(a) (b) (c)

Figure 3.14: Results on BSD500 Dataset in the testing set. *We compare Spectral SEEDS with and without SQ, and learning the patterns. Also, we compare it with normal SEEDS, for different number of superpixels. (a) Undersegmentation error (*the lower the better*), (b) Boundary Recall (*the higher the better*), (d) Achievable segmentation (*the higher the better*).*



(a) (b) (c)

Figure 3.15: Results on BSD500 Dataset. *We compare Spectral SEEDS with and without SQ, and learning the patterns to state-of-the-art methods for different number of superpixels. We compare it to normal SEEDS Van den Bergh et al. [2012], Felzenszwalb and Huttenlocher (FH) Felzenszwalb & Huttenlocher [2004], Entropy Rate Superpixels (ERS) Liu et al. [2011b] and SLIC Achanta et al. [2012]. (a) Undersegmentation error, (b) Boundary Recall, (d) Achievable segmentation. Better seen in color.*

superpixels with the parameters that report state-of-the-art results. In Spectral SEEDS with SQ, there is no color conversion, and Step 1 and 2 are implemented with the look-up table take less than 0.01 sec per image. Since the number of filters is the same as in SEEDS, Step 3 has the same computational cost.

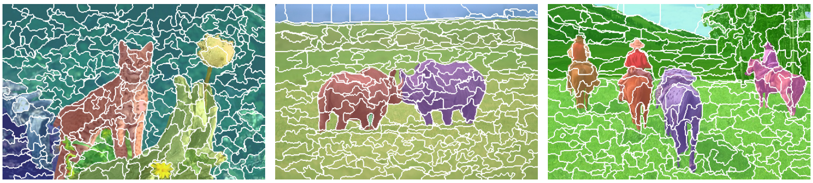


***Figure 3.16**: Example SEEDS segmentations with 200 superpixels. The ground-truth segments are color coded and blended on the images. The superpixel boundaries are shown in white.*

3.5.4 Comparison to state-of-the-art

In Figure 3.14 we report results on the testing set for different number of superpixels. We compare Spectral SEEDS with 500 learned filters, Spectral SEEDS with SQ with 64 filters, spectral SEEDS with 200 adapted filters for each image and SEEDS with Lab space and a color histogram of length 125. Spectral SEEDS with SQ with 64 filters achieves state-of-the-art performance, a bit better than SEEDS, with half of the computational cost. Spectral SEEDS with 500 learned filters performs better than SEEDS (at the expense of higher computational cost).

In Figure 3.15 we compare our methods to state-of-the-art superpixels methods, namely Felzenszwalb & Huttenlocher [2004], Entropy Rate Superpixels (ERS) [Liu et al., 2011b] and SLIC [Achanta et al., 2012]. Spectral SEEDS with SQ with 64 filters achieves state-of-the-art performance with half of the computational cost of the original SEEDS. Spectral SEEDS with 500 learned filters performs better than SEEDS (at the expense of higher computational cost). Finally, in Figures3.16 and 3.17 we show some examples of superpixels extracted with SEEDS using Lab color space and Spectral SEEDS, respectively. Note that the boundaries are shaky, because we do not use any boundary term in SEEDS, but we could include it as in [Van den Bergh et al., 2012] (for fair comparison, all SEEDS evaluated in this work have not used the boundary term).

3.6 Conclusion

We presented a superpixel algorithm based on SC and SEEDS superpixels, that can work on still images and also can take into account the spatio-temporal

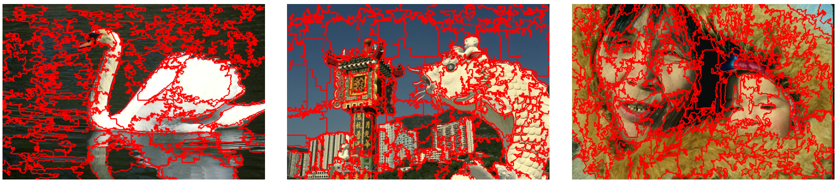


Figure 3.17: Examples of Spectral SEEDS with 200 superpixels. *The boundaries are in red.*

imformation on videos. Our algorithm is able to use color and texture filters, more complex than the color bins of the histogram used in the original SEEDS. We presented a method to speed-up the spectral embedding of the algorithm by using the primal form of SC, and by learning the pixel descriptors from a training set. To further speed-up our superpixel algorithm, we also introduced the use of sparse quantization. Our algorithm achieves state-of-the-art results and runs in real-time in a single CPU. Differents cues from color and texture could be used, and we plan to incorporate depth in future work.

4

Efficient Feature Extraction with an Efficient Feature Coding

4.1 Introduction

Recent research has led to important progress in visual object recognition and classification. Many state-of-the-art object recognition schemes consist of a feed-forward architecture, usually divided into feature extraction, feature encoding, spatial pooling, and a classifier [Boiman et al., 2008; Boureau et al., 2010a; Chatfield et al., 2011]. Recently, sophisticated encoding and pooling schemes have allowed for learning the different blocks in the pipeline [Zeiler et al., 2011; Yu et al., 2011].

Feature encoding is one of the most intriguing blocks in a feed-forward architecture. Encoding aims at partitioning the feature descriptor space into informative regions, in order to both generalize towards intra-class variances and discriminate between different categories. Many authors pointed out the significant influence feature encoding has on object recognition performance, and stressed the need for encodings with better generalization properties [Liu et al., 2011a; Coates & Ng, 2011]. The trend has been to introduce refined encodings that certainly generalize better and thus achieve higher levels of performance, but that come at the cost of increasing computational complexity. Indeed, feature encoding can be orders of magnitude more demanding than feature extraction and pooling, and it can become the bottleneck of the whole pipeline. For real-time and large scale applications, it is therefore crucial to design efficient feature encoding methods.

In the literature, one finds a plethora of feature encoding techniques. One of the most popular is assignment-based coding (AC). AC encodes a feature by assigning it to one or more codebook entries. When combined with average pooling, this corresponds to Bag-of-Words (BoW) [Csurka et al., 2004; Sivic & Zisserman, 2003]. Although simple in principle, it can be computationally demanding, since it requires the calculation of distances between the descriptors and the codebook entries. In particular, its computational cost increases with the size of the codebook and the number of patches [Nowak et al., 2006]. Therefore, several methods were devised to speed up the BoW approach, e.g. [Nister & Stewenius, 2006; Philbin et al., 2008; Moosmann et al., 2007].

Sparse Coding has emerged as a powerful alternative to AC, showing better results than AC, especially, when combined with max-pooling [Boureau et al., 2010a,b; Yang et al., 2009; Benoit et al., 2011]. In terms of efficiency, however, Sparse Coding optimizes a convex problem of the size of the codebook length, which makes it comparably slow. Locality-constrained Linear Coding (LLC) by Wang et al. [2010] is an approximation of Sparse Coding, which speeds it up without a significant loss in performance. The current state-of-the-art in visual classification makes use of super-vector coding [Zhou et al., 2010] and Fisher encoding [Perronin et al., 2010], *cf.* [Chatfield et al., 2011]. Yet, both methods are memory-demanding. Using compression methods, *e.g.* [Perronin et al., 2010], the Fisher kernel can be made tractable for large-scale applications, but the need for decompression before classification reduces the efficiency again.

In this work, we focus on AC because it is a promising compromise in terms of speed and performance [Liu et al., 2011a]. We introduce a new formulation for AC, and from that, we design a new efficient feature encoding scheme. At the heart of our formulation lies a quantization into a set of k-sparse vectors, which we denote as *sparse quantization*. It offers a novel viewpoint of AC, which, although algorithmically equivalent to AC, it allows for the unification of AC and sparse coding. In fact, our formulation allows for the design of a new efficient encoding. We coin it ‘Nested Sparse Quantization’ (NSQ), which consists of two feature encodings, in a nested architecture. We first encode the features, and the result is then fed to another feature encoder. Both encoders follow the novel formulation of AC. The first one is instantiated in a way that it is very fast to execute, and since it is build from a hard assignment encoder, it yields a binary vector. The second encoder is fed with this binary vector and thus also becomes very efficient to compute.

In the experimental part, we compare NSQ to state-of-the-art methods for object recognition. As we show on various datasets, *i.e.* Caltech 101, PASCAL

VOC 07 and ImageNet, our method demands orders of magnitude less time and memory than those previous approaches, while achieving competitive levels of performance. For instance, it is able to encode one million images in less than 24 hours using one laptop of 4 CPUs and 6GB of disk. We also report the accuracy performance in a standard semantic segmentation benchmark, MSRC-21, and we observe the same conclusions.

4.2 Sparse Quantization

In this Section, we introduce the mathematical tools that we use, *i.e.* quantization, sparsity and sparse quantization.

4.2.1 Quantization

Let $\mathbf{x} \in \mathbb{R}^q$ be a vector, and $\mathbf{B} \in \mathbb{R}^{q \times m}$ the so called codebook matrix with m entries $\mathbf{b}_i \in \mathbb{R}^q$, *i.e.* $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_m]$.

Definition 1. Quantization *is a mapping of a vector* $\mathbf{x}$ *into its closest vector of the codebook* $\{\mathbf{b}_i | i = 1, \dots, m\}$*, which we denote as* $\hat{\mathbf{x}}^\star = \arg\min_{\hat{\mathbf{x}} \in \{\mathbf{b}_i\}} \|\hat{\mathbf{x}} - \mathbf{x}\|^2$.

The purpose of quantization is to reduce the cardinality of the representation space. $\mathbf{x}$ has an infinite set of possible values, and when mapped to the codebook, it is restricted to a finite set of possible vectors. In general, the cost of computing a quantization is $\mathcal{O}(m\jmath)$, where $\jmath$ represents the cost of computing a single distance. In practice, the most common used is the squared Euclidean distance, which is expensive since it implies $2q$ float operations, and yields a total cost of $\mathcal{O}(mq)$.

4.2.2 Sparsity

A vector $\boldsymbol{\alpha}$ is k-sparse when it has at most k non-zero entries, *i.e.* $\|\mathbf{x}\|_0 \leq k$. Let $\mathbb{R}^m_k$ be the space of k-sparse vectors in $\mathbb{R}^m$, which is $\mathbb{R}^m_k = \{\boldsymbol{\alpha} \in \mathbb{R}^m : \|\boldsymbol{\alpha}\|_0 \leq k\}$. Sparsity is a highly non-linear model [DeVore, 1998]. Observe that the the sum of two k-sparse vectors does not necessarily result in another k-sparse vector, since their non-zero entries might not coincide. Thus, the result

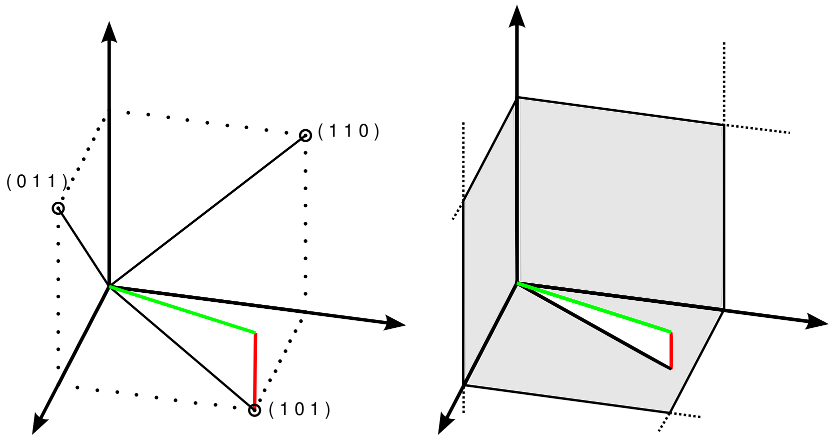


***Figure 4.1**: Representation of $\mathbb{B}_2^3$ (left) and $\mathbb{R}_2^3$ (right) space with the input vector* $\mathbf{x}$ *in green, and the codification error vector* $(\hat{\mathbf{x}} - \mathbf{x})$ *in red.*

is no longer in $\mathbb{R}_k^m$. We also introduce the subset of binary vectors in $\mathbb{R}_k^m$, which is the set of vectors with k ones and $(m-k)$ zeros. Let $\{0,1\}_k^m = \mathbb{B}_k^m$ be the space of *binary* k-sparse vectors, $\mathbb{B}_k^m = \{\boldsymbol{\alpha} \in \mathbb{B}^m : \|\boldsymbol{\alpha}\|_0 = k\}$, where $\mathbb{B}_k^m$ is a subset of $\mathbb{R}_k^m$, *i.e.* $\mathbb{B}_k^m \subset \mathbb{R}_k^m$, and it is composed of a finite amount of elements, in particular, its cardinality is $|\mathbb{B}_k^m| = \binom{m}{k}$.

4.2.3 Sparse Quantization

We consider the particular case of the quantization when the codebook is the set of k-sparse vectors.

Definition 2. Sparse Quantization *is a quantization into the codebook* $\mathbf{B} = \mathbb{R}_k^q$.

The formulation for the Sparse Quantization of $\mathbf{x} \in \mathbb{R}^q$ is

$$\hat{\mathbf{x}}^\star = \arg\min_{\hat{\mathbf{x}} \in \mathbb{R}_k^q} \|\hat{\mathbf{x}} - \mathbf{x}\|^2. \tag{4.1}$$

In the following, we assume that $\mathbf{x} \in \mathbb{R}_+^q$ and that $\mathbf{x}$ is normalized. For brevity, we use the term *Sparse Quantization* (SQ), but to be more precise, it should be called quantization into the codebook $\mathbb{R}^q_k$. Interestingly, Sparse Quantization can be done in a more efficient way than a quantization into an arbitrary codebook. In the following propositions, we show that SQ can be achieved with a sorting algorithm, and it does not require the computation of the distances to the codebook entries. The first proposition shows the case for the codebook $\mathbb{B}^q_k$, which is what we use in our approach. The second proposition is the derivation for Sparse Quantization in the case where the codebook is the set $\mathbb{R}^q_k$. We assume that $\mathbf{x} \in \mathbb{R}^q_+$, and that $\mathbf{x}$ is normalized to 1. In all cases where we use the propositions this is true.

Proposition 1. *Let* $\hat{\mathbf{x}}^\star = \arg\min_{\hat{\mathbf{x}} \in \mathbb{B}^q_k} ||\hat{\mathbf{x}} - \mathbf{x}||^2$ *be the quantization into* $\mathbb{B}^q_k$ *of* $\mathbf{x} \in \mathbb{R}^q_+$, $||\mathbf{x}||^2_2 = 1$. *We can obtain* $\hat{\mathbf{x}}^\star$ *by*

$$\hat{x}^\star_i = \left\{ \begin{array}{ll} 1 & \textit{if } i \in k\textit{-Highest}(\mathbf{x}) \\ 0 & \textit{otherwise} \end{array} \right. , \tag{4.2}$$

where k*-Highest*$(\mathbf{x})$ *is the set of dimensions indices that indicate which are the* k *highest values in the vector* $\mathbf{x}$.

Proof. We first rewrite $||\hat{\mathbf{x}} - \mathbf{x}||^2$ as $\sum_i (\hat{x}_i - x_i)^2$. Since $\hat{\mathbf{x}} \in \mathbb{B}^m_k$ has k elements set to 1 and $(m-k)$ set to 0, we can write the above summation as

$$\sum_i (\hat{x}_i - x_i)^2 = \sum_{i:\hat{x}_i = 1} (1 - x_i)^2 + \sum_{i:\hat{x}_i = 0} (x_i)^2. \tag{4.3}$$

We sort in descending order the set of values at each dimension of $\mathbf{x}$, *i.e.* we sort $\{x_i\}$, and we use a new indexing in this ordered set. We indicate so by using $\mathbf{x}'$, and we index it with s instead of i, such that $x'_{(s-1)} > x'_s$. To see when Equation (4.3) is minimum, note that

$$(x'_1)^2 > \ldots > (x'_{(s-1)})^2 > (x'_s)^2 > \ldots \tag{4.4}$$

$$(1 - x'_1)^2 < \ldots < (1 - x'_{(s-1)})^2 < (1 - x'_s)^2 < \ldots, \tag{4.5}$$

where (4.5) is due to the assumption $||\mathbf{x}||^2 = 1$. Therefore, to make both terms in (4.3) minimum, we set the k ones in $\hat{\mathbf{x}}$ such that $(1 - x'_s)^2$ in (4.5) are minimum, and we set $(m-k)$ zeros in $\hat{\mathbf{x}}$ such that $(x'_s)^2$ in (4.4) are minimum. Thus, we set the k ones in the k highest values, and 0 to the other $(m-k)$ values, which is what the Proposition says. □

Proposition 2. *Let* $\hat{\mathbf{x}}^\star$ *be* $\mathbf{x} \in \mathbb{R}^q_+$ *quantized into* $\mathbb{R}^q_k$, i.e. $\hat{\mathbf{x}}^\star = \arg\min_{\hat{\mathbf{x}} \in \mathbb{R}^q_k} \|\hat{\mathbf{x}} - \mathbf{x}\|^2$. *We can obtain* $\hat{\mathbf{x}}^\star$ *by setting to the* x_i *value the* k *highest dimensions in* $\mathbf{x}$, *and* 0 *otherwise,* i.e.

$$\hat{x}^\star_i = \begin{cases} x_i & \textit{if } i \in k\textit{-Highest}(\mathbf{x}) \\ 0 & \textit{otherwise} \end{cases}, \tag{4.6}$$

where k*-Highest*$(\mathbf{x})$ *is the set of dimensions indices that indicate which are the* k *highest values in the vector* $\mathbf{x}$.

Proof. We can proceed as in the proof of Proposition 1, but instead, $\hat{\mathbf{x}}^\star$ is in the set $\mathbb{R}^m_k$. The minimum of $\sum_i (\hat{x}_i - x_i)^2$ is achieved when $i : \hat{x}_i = 0$ selects the lowest x_i. Thus,

$$\sum_i (\hat{x}_i - x_i)^2 = \sum_{i:\hat{x}_i \neq 0} (x_i - x_i)^2 + \sum_{i:\hat{x}_i = 0} (x_i)^2 = \sum_{i:\hat{x}_i = 0} (x_i)^2. \tag{4.7}$$

This is the same as selecting the k highest elements of $\mathbf{x}$, and setting $\hat{x}_i = x_i$, and 0 otherwise. □

Proposition 1 and Proposition 2 show that the Sparse Quantization can be done with a sorting algorithm of the set $\{x_i\}$, and selecting the k highest values. It has a computational cost of $\mathcal{O}(q)$, in contrast to a general quantization, which has a cost of $\mathcal{O}(mq)$, where in practice $m \gg q$. In Figure 4.1, we show an example of Sparse Quantization in $\mathbb{B}^3_2$ and in $\mathbb{R}^3_2$. We plot an input vector $\mathbf{x}$ in green, and the quantization error in red. Also, we show all $\binom{3}{2} = 3$ possible vectors for $\mathbb{B}^3_2$.

4.3 A New Formulation of Assignment-based Coding

In this Section, we introduce a new framework for Assignment-based Coding (AC), which is based on the principles of Sparse Quantization. Additionally, we show the relation to Sparse Coding.

4.3.1 Assignment-based Coding Revisited

We first review AC and its main variants and identify some common terminology in the literature. This will serve as the basis for our new formulation that we introduce in the subsequent Section.

Let $\mathbf{x} \in \mathbb{R}_+^q$ be the vector of a patch descriptor, which has been normalized, and $\mathbf{B} \in \mathbb{R}^{q \times m}$ the codebook matrix with m entries $\mathbf{b}_i \in \mathbb{R}^q$. AC aims at encoding $\mathbf{x}$ by selecting the codebook entries $\mathbf{b}_i$ with minimum distance to $\mathbf{x}$. Such codebook entries are called the k-Nearest Neighbors (k-NN) of $\mathbf{x}$. k is the number of selected codebook entries, and it is a predefined constant.

Vector quantization (VQ) is the simplest AC method. It only selects a single codebook entry, which is the closest to $\mathbf{x}$, with the so called 1-NN. It is in fact a quantization, since it maps $\mathbf{x}$ into a codebook entry. Let $\boldsymbol{\alpha} \in \mathbb{R}^m$ be the vector that encodes the assignment. In VQ, α_i is set to 1 when it is the element of the codebook that is used to quantize $\mathbf{x}$, and the rest is set to 0.

VQ has been extended to the popular Hard-Assignment (HA), which assigns more than one codebook entry to $\mathbf{x}$, *e.g.* [Philbin et al., 2008]. In this case, there are 1s placed in the elements of $\boldsymbol{\alpha}$ that correspond to the k-NN codebook entries, *i.e.*

$$\alpha_i = \begin{cases} 1 & \text{if } i \in k\text{-NN}(\mathbf{x}, \mathbf{B}) \\ 0 & \text{otherwise} \end{cases} . \tag{4.8}$$

k-NN$(\mathbf{x}, \mathbf{B})$ is the set that indicates which k codebook entries are closest to $\mathbf{x}$. According to Definition 1, HA is not a quantization anymore but an assignment, because more than one vector of the codebook is assigned to $\mathbf{x}$. In the case of quantization, $\mathbf{x}$ is always represented with only a single codebook vector.

In HA, $\boldsymbol{\alpha}$ results in a binary vector with k ones and $(m-k)$ zeros. Thus, the $\boldsymbol{\alpha}$'s obtained are k-sparse vectors, which are in $\mathbb{B}_k^m$. In particular, $\boldsymbol{\alpha}$ is in $\mathbb{B}_k^m$ when using HA, and in $\mathbb{B}_1^m$ for VQ.

There exists also the soft version of HA, denoted as Soft-Assignment (SA) [Liu et al., 2011a]. In the work by Liu et al. [2011a], $\boldsymbol{\alpha}$ is normalized such that $\|\boldsymbol{\alpha}\|_2^2 = 1$. We assume that the normalization of $\boldsymbol{\alpha}$ is done in the pooling stage, for the sake of simplicity in the codification. Also, we refer to SA for what [Liu et al., 2011a] calls localized SA. In this case, instead of indicating the k-NN with 1s, $\boldsymbol{\alpha}$ contains some notion of the relative distance to the selected codebook entries, and becomes

$$\alpha_i = \begin{cases} \exp(-\beta d(\mathbf{x}, \mathbf{b}_i)) & \text{if } i \in k\text{-NN}(\mathbf{x}, \mathbf{B}) \\ 0 & \text{otherwise} \end{cases} . \tag{4.9}$$

$d(\mathbf{x}, \mathbf{y})$ is a given distance function, β a learned constant, and $d(\mathbf{x}, \mathbf{b}_i)$ is the distance between $\mathbf{x}$ and $\mathbf{b}_i$. Thus, SA codifies a patch descriptor $\mathbf{x}$ with a k-sparse $\boldsymbol{\alpha}$ in $\mathbb{R}^m_k$. Note that when $\beta = 0$ we recover HA, and $\boldsymbol{\alpha} \in \mathbb{B}^m_k$.

AC and other codings are usually the computational bottleneck of the patch-based image classification, *cf.* [Chatfield et al., 2011]. Typically, the cost of coding scales with the number of visual words, the number of patch descriptors and the dimensionality of these descriptors. This can make the coding an order of magnitude slower than the patch description. The reason is mainly because the encoding requires computing the distances between the patch descriptor and all the codebook entries.

In the following, we introduce a new formulation for AC, that also leads up to a very efficient quantization scheme.

4.3.2 AC as a Sparse Quantization

We now introduce a new view of AC based on a non-linear mapping of $\mathbf{x}$, which we denote as $\phi : \mathbb{R}^q \rightarrow \mathbb{R}^m$.

Definition 3. *Let $\phi(\mathbf{x})$ be the following mapping:*

$$\phi(\mathbf{x}) = \frac{1}{Z}[\exp(-\beta d(\mathbf{x}, \mathbf{b}_1)) \; \ldots \; \exp(-\beta d(\mathbf{x}, \mathbf{b}_m))], \tag{4.10}$$

where β is a learned constant and $d(a, b)$ a given distance. Z normalizes the vector, and $\{\mathbf{b}\}$ parametrize the mapping.

Observe that in AC, $\mathbf{B}$ denotes the codebook matrix, but in our new formulation it is used to parametrize the mapping $\phi(\mathbf{x})$. In the following proposition, we rewrite HA as a *Sparse Quantization* of $\phi(\mathbf{x})$ (Section 4.2.3). Then, we show this fact for SA.

Proposition 3. *Let $\boldsymbol{\alpha}^\star \in \mathbb{B}^m_k$ be the result of Hard Assignment in Equation (4.8). Then, the same $\boldsymbol{\alpha}^\star$ can be obtained through the following Sparse Quantization of $\phi(\mathbf{x})$:*

$$\boldsymbol{\alpha}^\star = \arg \min_{\boldsymbol{\alpha} \in \mathbb{B}^m_k} ||\boldsymbol{\alpha} - \phi(\mathbf{x})||_2^2. \tag{4.11}$$

Sketch of the Proof. We apply Proposition 1 to Equation (4.11), and we obtain that

$$\alpha_i^\star = \begin{cases} 1 & \text{if } i \in k\text{-Highest}(\phi(\mathbf{x})) \\ 0 & \text{otherwise} \end{cases}. \tag{4.12}$$

Finally, note that k-Highest$(\phi(\mathbf{x}))$ is equivalent to k-NN$(\mathbf{x}, \mathbf{B})$, and we recover the HA formulation. □

Proposition 3 shows that HA is a Sparse Quantization of $\phi(\mathbf{x})$ into the codebook $\mathbb{B}_k^m$. HA is formulated as a non-linear transformation $\phi(\mathbf{x})$, which uses $\mathbf{B}$, and then the resulting vector is quantized into $\mathbb{B}_k^m$. This is in contrast to how HA has been usually interpreted. Previous HA formulations used $\boldsymbol{\alpha}$ in order to indicate to which codebook entries $\mathbf{x}$ is assigned. Both formulations obtain the same coding results, and have the same computational complexity.

We now rewrite SA [Liu et al., 2011a] as a *Sparse Quantization* of $\phi(\mathbf{x})$ in $\mathbb{R}_k^q$. Recall that we consider that it is the pooling stage that normalizes $\boldsymbol{\alpha}$.

Proposition 4. *Let $\boldsymbol{\alpha}^\star \in \mathbb{R}_k^m$ be the result of Soft-Assignment of $\mathbf{x}$ to the codebook $\mathbf{B}$, as in Equation* (4.9). *Then, $\boldsymbol{\alpha}^\star$ can also be obtained through the following Sparse Quantization of $\phi(\mathbf{x})$:*

$$\boldsymbol{\alpha}^\star = \arg \min_{\boldsymbol{\alpha} \in \mathbb{R}_k^m} ||\boldsymbol{\alpha} - \phi(\mathbf{x})||^2. \tag{4.13}$$

Sketch of the Proof. We apply Proposition 2 to Equation (4.13), and we obtain that

$$\alpha_i^\star = \begin{cases} \phi_i(\mathbf{x}) & \text{if } i \in k\text{-Highest}(\phi(\mathbf{x})) \\ 0 & \text{otherwise} \end{cases}. \tag{4.14}$$

Finally, note that k-Highest$(\phi(\mathbf{x}))$ is equivalent to k-NN$(\mathbf{x}, \mathbf{B})$, and together with Definition 3, we recover the SA formulation of Equation (4.9). □

This shows that SA can also be seen as a quantization. That is, SA is equivalent to quantizing $\phi(\mathbf{x})$ into $\mathbb{R}_k^m$ by selecting the vector in $\mathbb{R}_k^m$ closer to $\phi(\mathbf{x})$.

In the following, we make use of the re-interpretation of HA and SA, and we show that Proposition 3 and Proposition 4 are able to fuse Assignment-based Coding and Sparse Coding into a single formulation. Also, we show that we can use the new formulation to design a very efficient codification.

4.3.3 A Formulation for Sparse Coding and AC

When considering AC as a Sparse Quantization, we can analyze its relation to Sparse Coding. Sparse Coding is the following coding scheme [Olshausen & Field, 1997]:

$$\boldsymbol{\alpha}^\star = \arg \min_{\boldsymbol{\alpha} \in \mathbb{R}^m_k} ||\mathbf{C}\boldsymbol{\alpha} - \mathbf{x}||_2^2, \tag{4.15}$$

where $\mathbf{C} \in \mathbb{R}^{q \times m}$ is the codebook used in Sparse Coding. It is usually relaxed in the following way

$$\boldsymbol{\alpha}^\star = \arg \min_{\boldsymbol{\alpha} \in \mathbb{R}^m} ||\mathbf{C}\boldsymbol{\alpha} - \mathbf{x}||_2^2 + \lambda \|\boldsymbol{\alpha}\|_1, \tag{4.16}$$

where $\lambda\|\boldsymbol{\alpha}\|_1$ is the convex sparsity regularization term of $\boldsymbol{\alpha} \in \mathbb{R}^m_k$, and it can be tuned to enforce different degrees of sparsity. Note that in [Yang et al., 2009] it was already shown that Sparse Coding can be seen as an extension of VQ; however, their formulation does not extend to AC in general.

The general formulation for Sparse Coding and Assignment-based Codings is:

$$\boldsymbol{\alpha}^\star = \arg \min_{\boldsymbol{\alpha} \in \mathbb{R}^m_k} ||\mathbf{C}\boldsymbol{\alpha} - \phi(\mathbf{x})||_2^2, \tag{4.17}$$

where $\phi(\mathbf{x})$ represents a mapping from the original q-dimensional vector $\mathbf{x}$ to a space with a possibly different dimension, such as m in the earlier definition. We now recover Sparse Coding by just setting $\phi(\mathbf{x}) = \mathbf{x}$, and we recover AC by setting $\mathbf{C}$ to be the identity matrix $\mathbf{I}^{m \times m}$.

4.4 Nested Sparse Quantization

In this Section, we introduce an efficient encoding built upon the new formulation of HA in Proposition 3. It consists of two Sparse Quantizations placed in a nested architecture, coined Nested Sparse Quantization (NSQ). We define NSQ as the following optimization problem:

$$\boldsymbol{\alpha} = \arg \min_{\boldsymbol{\alpha} \in \mathbb{B}^m_k} \|\boldsymbol{\alpha} - \phi(\hat{\mathbf{x}})\|^2, \tag{4.18}$$

$$\text{where } \hat{\mathbf{x}} = \arg \min_{\hat{\mathbf{x}} \in \mathbb{B}^{m'}_{k'}} \|\hat{\mathbf{x}} - \theta(\mathbf{x})\|^2, \tag{4.19}$$

where Equation (4.18) is the *outer* Sparse Quantization and Equation (4.19) is the *inner* Sparse Quantization, respectively. The parameters in the inner quantization are m', k' and the mapping $\theta(\mathbf{x})$, and they are not necessarily the same as in the outer quantization. The inner quantization results in the binary $\hat{\mathbf{x}}$, as a result of the Sparse Quantization into $\mathbb{B}^m_k$. The mapping in the inner quantization $\theta(\mathbf{x})$ has the real-valued $\mathbf{x}$ as input, whereas the outer mapping $\phi(\hat{\mathbf{x}})$ gets the vector $\hat{\mathbf{x}} \in \mathbb{B}^{m'}_{k'}$ as input, which is a binary vector.

4.4.1 Implementation Advantages

Observe that the input to the outer quantization has a *finite* amount of possible values, because it receives one of the entries of the inner codebook. This allows to use a look up table for the outer quantization, by memorizing the quantization of all entries of the inner codebook. However, in practice, the cardinality of the inner codebook is huge. In NSQ, the outer quantization is a mapping from a set of $\binom{m'}{k'}$ elements to $\binom{m}{k}$ possible outputs, where in practice $\binom{m'}{k'} \gg \binom{m}{k}$, and they are too high to be memorized in a look up table.

Indeed, the advantage of NSQ results from an appropriately chosen nesting of two different quantizations. The inner quantization turns real-valued vectors $\mathbf{x}$ into one of the $\binom{m'}{k'}$ possible binary vectors $\hat{\mathbf{x}}$, and the non-linear mapping should be simple not to overload the system. The outer quantization starts from a vector in the set $\mathbb{B}^{m'}_{k'}$, which are binary vectors, and can therefore be used to very efficiently deploy a more complex mapping $\phi(\hat{\mathbf{x}})$.

Inner Quantization

For the inner quantization, we use the most efficient and simplistic mapping possible, which is $\theta(\mathbf{x}) = \mathbf{x}$; that is, the mapping $\theta(\mathbf{x})$ does not apply any transformation on $\mathbf{x}$. Thus, the inner quantization in Equation (4.19) directly quantizes the input $\mathbf{x}$ into the k'-sparse codebook for that step, without any previous non-linear mapping. It becomes

$$\hat{\mathbf{x}} = \arg \min_{\hat{\mathbf{x}} \in \mathbb{B}^q_{k'}} \|\hat{\mathbf{x}} - \mathbf{x}\|^2, \tag{4.20}$$

where m' is equal to q, and k' is the only parameter to be set. As stated in Proposition 1, optimizing this problem is equivalent to computing the k'-Highest values in $\{x_i\}$. This implies selecting the k' elements of $\mathbf{x}$ with the

Algorithm 6: Nested Sparse Quantization

```
Input: $\mathbf{x} \in \mathbb{R}^q_+$, $\hat{\mathbf{B}} \in \mathbb{B}^{q \times m}_{k'}$
Output: $\boldsymbol{\alpha}$
$\hat{\mathbf{x}} \leftarrow$ Set $k'$ highest values of $\mathbf{x}$ to 1 and the rest to 0
foreach $i = \{1, \ldots, m\}$ do
    $\phi_i(\hat{\mathbf{x}}) = \sum_{j=1}^{q} (\hat{x}_j \odot \hat{b}_{ij})$
end
$\boldsymbol{\alpha} \leftarrow$ Set $k$ highest values of $\phi(\hat{\mathbf{x}})$ to 1 and the rest to 0
```

highest values, which has a computational cost of $\mathcal{O}(q)$. One could investigate alternatives to $\theta(\mathbf{x}) = \mathbf{x}$ to improve on the classification results, but this would reduce the speed again and is therefore not studied.

Outer Quantization

The outer quantization now gets the binary output of the inner quantization as its input, which is a vector in $\mathbb{B}^{m'}_{k'}$. The use of binary vectors allows to reduce the complexity by using fast bit-wise comparisons, which most modern CPUs handle with dedicated instructions [Shakhnarovich, 2005; Calonder et al., 2010]. Thus, we can afford applying a more sophisticated mapping ϕ. Recall that the computational cost of such $\phi(\hat{\mathbf{x}})$ still is $\mathcal{O}(m\jmath)$, with $\jmath$ the cost of computing the distance. We can reduce the cost $\jmath$ by the very fact that $\hat{\mathbf{x}}$ is binary. Thus, the mapping is the same as in Equation (4.10) but changing the distance function to a Hamming distance, $\hat{d}(\hat{\mathbf{x}}, \hat{\mathbf{b}}_i)$. Recall that the $\hat{\mathbf{b}}_i$ are parameters used in the definition of the non-linear mapping, and do not correspond neither to the codebook vectors of the inner quantization nor to the codebook vectors of the outer quantization. In our case, these are obtained by piping a total of m vectors $\mathbf{b}_i$ through the inner quantization, thus yielding those $\hat{\mathbf{b}}_i$.

Next, the mapping $\phi(\hat{\mathbf{x}})$ can be further simplified by dropping the exponential functions, given the equivalence of $\arg\min_{\boldsymbol{\alpha}}$ with the mapping $\phi(\hat{\mathbf{x}}) = \frac{1}{Z}[\tilde{d}(\hat{\mathbf{x}}, \hat{\mathbf{b}}_1) \ \ldots \ \tilde{d}(\hat{\mathbf{x}}, \hat{\mathbf{b}}_m)]$. Now, suppose we take $\tilde{d}(\hat{\mathbf{x}}, \hat{\mathbf{b}}_i) = \sum_{j=1}^{q} (\hat{x}_j \odot \hat{b}_{ij})$, where $\odot$ is the negation of the exclusive OR operator (*nxor*). It returns 1 if both elements are equal, and 0 otherwise. This is again very efficient to compute.

In Algorithm 6 we depict the implementation of NSQ, which highlights the simplicity and efficiency of the method.

4.4.2 Discussion

Quantization vs. recognition accuracy

At this point, one may object that NSQ adds an extra quantization step in the feature encoding procedure, and that quantization errors may grow worse, leading to a deteriorated performance. Yet, in recognition, quantization should also allow for sufficient generalization. In the experiments, we show that the recognition performance is not impaired, but that the speed goes up dramatically.

Relation of the Inner Quantization to Binary Embeddings and Hashing

NSQ is not to be confused with Binary Embeddings or Hashing-based methods. The goal of the inner quantization is a first step of feature encoding, and it is not designed to be a fast nearest neighbor extractor nor a fast distance approximator. In fact, in the experiments we show that NSQ is not a good nearest neighbor approximator. Hashing-based methods tackle fast distance computation rather than feature encoding, and have been typically used to speed up image retrieval in large-scale databases [Indyk & Motwani, 1998; Weiss et al., 2008; Gordo & Perronnin, 2011; Hervé Jégou, 2011]. The binarizations used by these methods range from thresholding linear combinations of input features to solving optimization problems. These have a negligible cost in image retrieval, but not when applied to feature encoding. For example, in Locality Sensing Hashing the binary embedding is $h_n(x) = T(\mathbf{r}^T\mathbf{x})$, where $\mathbf{r}$ is a projection vector, and $T(\,\cdot\,)$ is a threshold function. Hashing uses L different N-dimensional embedding functions $h(x)$ of an image signature $\mathbf{x}$, and thus, the cost of computing $h(\mathbf{x})$ is $\mathcal{O}(LNq)$. Hashing is typically used in applications where the cost $\mathcal{O}(LNq)$ is negligible compared to the cost of searching the nearest-neighbors, because this search might be among hundreds of thousands of elements. However, in feature encoding, the time of computing $h(\mathbf{x})$ is not negligible anymore, because the nearest neighbor search is only among thousands of elements (the size of the codebook), and in addition, $h(\mathbf{x})$ is computed for each patch. In contrast, the inner quantization of NSQ obtains the binary $\hat{\mathbf{x}}$ at a cheap cost $\mathcal{O}(q)$. Furthermore, the binary embeddings generate each bit independently of the others [Gordo & Perronnin, 2011], which is different from NSQ, in which the output bits are dependent since only k of them are 1.

Sparse Quantization for Patch Descriptors

In [Boix et al., 2013], we extended the NSQ for feature encoding for patch descriptors. We introduce a new formulation to patch description based on Sparse Quantization. We take advantage of the capabilities of our formulation to design novel, more discriminative and computationally efficient (binary) descriptors. This allows for efficient encodings, leading to powerful, novel binary descriptors, yet also to the generalization of existing descriptors like SIFT or BRIEF. We also show the advantage of having a general formulation that can be adapted to the task at hand. We demonstrate the capabilities of our formulation for both keypoint matching and image classification. Our binary descriptors achieve state-of-the-art results for two keypoint matching benchmarks. In image classification we show that, the parameters of our formulation that recover the SIFT descriptor work best for this task. Since SIFT descriptors and Hard-Assignment can be formulated as a SQ, we can see the full network for image classification as a hierarchy of SQ from the pixel level. In [Boix et al., 2013] we report that the hierarchy of SQ achieves comparable results to the hierarchy of Sparse Coding [Yu et al., 2011]. We refer to [Boix et al., 2013] for more details about the learning of patch descriptors with Sparse Quantization.

4.5 Experiments in Image Classification

In this Section, we report experiments of the NSQ for image classification on different datasets, namely Caltech101 [Fei-Fei et al., 2006], PASCAL VOC 2007 [Everingham et al., 2007] and ImageNet [Deng et al., 2009]. After describing the datasets and implementation details, we report the results, and analyze the influence of the different parameters of the NSQ. In the next Section, we report results for semantic image segmentation.

4.5.1 Datasets

In none of the experiments we use flipped or blurred images to extend the training set, for fair comparison between the different methods.

Caltech 101

This dataset contains 102 different classes with about 50 images per class. We use 3 random splits of 30 images per class for training and the rest for testing, and evaluate it with the average classification accuracy across all classes. We resize the image to have a maximum of 300 pixels per dimension.

PASCAL VOC 2007

It consists of a total of 9,963 images with 20 different object classes. Half of the dataset is used for training and the other half for testing. The evaluation is based on the mean average precision (mAP) across all classes.

ImageNet

We create a new dataset taking a subset of $1,065,687$ images of ImageNet. This subset contains images of 909 different classes that do not overlap in the synset. We randomly split this subset into two halfs, one for training and one for testing, maintaining the proportion of images per class. For evaluation, we report the average classification accuracy across all classes. Note that we do not use the protocol of taking the 5 highest scores per image and keeping the best. We resize the image to have a maximum of 300 pixels per dimension.

4.5.2 Implementation Details

Patch Descriptors

We use SIFT [Lowe, 2004] extracted from patches on a regular grid, at different scales. In Caltech 101 and ImageNet they are extracted at every 8 pixels and at the scales of 16, 32 and 48 pixels diameter. In VOC07, SIFT is sampled at each 4 pixels and at the scales of 12, 24 and 36 pixels diameter.

Feature Encoding

Apart from comparing NSQ and HA, we also report results with LLC [Wang et al., 2010] and the Super Vector method [Zhou et al., 2010]. We found that for all settings $k = 5$ is the optimal, except for the combination of HA, max-pooling and linear SVM, which is $k = 10$. In NSQ, $k' = 25$ for the inner quantization (see discussion in 4.5.3 for those choices).

Codebook Generation

The codebook $\mathbf{B}$ is typically built with a learning algorithm either unsupervised [Csurka et al., 2004; Sivic & Zisserman, 2003] or supervised [Moosmann et al., 2007]. Recently, Coates & Ng [2011] showed that a similar performance can be achieved by randomly picking a set of patches as codebook entries. We compared a codebook obtained using k-means clustering versus building it taking random patches from the training set. Tallying with the observations in [Coates & Ng, 2011], compared to k-means clustering, random selection considerably reduces the training time but does not lower the performance for large codebooks. For the methods that are not AC, like LLC or Super Vector, we learned the vocabularies as specified for each of them.

Pooling

We use in all datasets spatial pyramids. In Caltech 101 and ImageNet we divide the image in 4×4, 2×2 and 1×1 regions, yielding 21 different regions, and in VOC07, 3×1, 2×2 and 1×1 regions, yielding 8 regions. We use max-pooling because it has been shown to systematically outperform other types of pooling. In the case of NSQ and HA, the max-pooling results in a binary vector, which is the descriptor of the whole image, which we do not normalize.

Classification

For Caltech 101 and VOC07, we use a linear one versus rest SVM classifier for each class with the parameter C of the SVM set to 1000. In Caltech 101, we also report results using the RB-χ^2 kernel, $\exp\{-\beta\chi^2\}$, setting β to the mean of the pairwise distances in the training data. Before computing the χ^2 distance, the features are normalized with the ℓ_1-norm. When the image descriptor is binary, the RB-χ^2 kernel can be computed efficiently using dedicated CPU instructions [Shakhnarovich, 2005; Calonder et al., 2010]. In ImageNet we use as classifier an approximated nearest neighbor that takes a maximum of 400 random examples per class.

Computational Evaluation

We use CPUs at 3.07GHz with the SSE4.1 instruction set, that enables fast popcnt instructions for bit strings.

4.5.3 Results

Performance Evaluation

In Table 4.1 we report results on Caltech 101. We show the results for NSQ in combination with different SVM kernels and pooling schemes. We do the same for HA, and we also include a comparison with LLC and Super Vector. We use a codebook of $8,192$ entries for all methods except for Super Vector for which we use $1,024$, since it achieves the compromise between accuracy and efficiency. LLC and Super Vector perform better than the others, as was reported in previous papers. However, their coding consumes much more memory and time than NSQ. The performance of NSQ is only 2.7% less than the best competing method which is Super Vector, but NSQ requires 500 times less memory than Super Vector and is about 10 times faster, and is 20 times faster than other methods that use the same codebook size.

In Table 4.2 we summarize the results in VOC07. We use a codebook size of $16,258$ entries. NSQ obtains a 30 times speed up, while only degrading 2% in performance.

In both tables we also provide the state-of-the-art methods in the literature. Even though they obtain better performance than our method, these techniques come at a far higher computational cost, as seen in the previous comparisons. We did not reproduce results of SA because it was already observed by Liu et al. [2011a] that SA only outperforms HA by less than 1%. Also, note that in VOC07 there is a gap between the performance of LCC reported by Wang et al. [2010] and the LCC using our descriptors. According to Chatfield et al. [2011], this can be explained by the fact that Wang et al. [2010] extend the training set by blurring and flipping the images, and we do not.

Influence of the Parameters

In Figure 4.2, we compare time and accuracy when changing the codebook size for NSQ and HA. Up to a certain point, the larger the codebook, the better

	Method				**Time per Image 1 CPU *(sec)***			**Memory**
	Coding	**Pool**	**Kernel**	**Acc. *(%)***	**Feature**	**Coding**	**Kernel**	***(Bytes)***
NSQ	HA$_{\{25,10\}}$	Max	Linear	$74.2 \pm .7$	$0.27 \pm .11$	$0.17 \pm .05$	–	21K
	HA$_{\{20,5\}}$	Max	Linear	$73.7 \pm .3$	$0.27 \pm .11$	$0.15 \pm .05$	–	21K
	HA$_{\{20,5\}}$	Max	RB-χ^2	$72.0 \pm .6$	$0.27 \pm .11$	$0.15 \pm .05$	.02	21K
	HA$_{\{20,5\}}$	Sum	RB-χ^2	$71.6 \pm .3$	$0.27 \pm .11$	$0.17 \pm .07$	2.10	672K
Standard	HA$_{\{10\}}$	Max	Linear	$74.6 \pm .6$	$0.27 \pm .11$	$2.78 \pm .9$	–	21K
	HA$_{\{5\}}$	Max	Linear	$74.2 \pm .1$	$0.27 \pm .11$	$2.75 \pm .9$	–	21K
	HA$_{\{5\}}$	Max	RB-χ^2	$73.4 \pm .9$	$0.27 \pm .11$	$2.75 \pm .9$	.02	21K
	HA$_{\{5\}}$	Sum	RB-χ^2	$72.1 \pm .4$	$0.27 \pm .11$	$2.78 \pm .9$	2.10	672K
	LLC	Max	Linear	$76.1 \pm .7$	$0.27 \pm .11$	$2.95 \pm .9$	–	672K
	SV	Max	Linear	$76.9 \pm .9$	$0.27 \pm .11$	$1.52 \pm .9$	–	10752K
Other features	HA	Max	Linear	$73.3 \pm .6$	–	–	–	–
	SA	Max	Linear	$74.2 \pm .8$	–	–	–	–
	LLC	Max	Linear	$73.4 \pm -$	–	–	–	–
	FK	Max	Linear	$77.8 \pm .6$	–	–	–	–

***Table 4.1**:* Quantitative Results on Caltech 101. *The time and memory measurements are for one image. We refer to HA$_{\{k',k\}}$ for the HA with NSQ, and HA$_{\{k\}}$ for the standard, in which we indicate the k and k' parameters of the outer and inner quantizations, respectively. For the last four rows, the results are from the original paper, this is, for HA are from [Liu et al., 2011a], for SA from [Liu et al., 2011a], and for LLC from [Wang et al., 2010] FK Chatfield et al. [2011]*

	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dinning Table	Dog	Horse	Motorbike	Person	Potted Plant	Sheep	Sofa	Train	TV/Monitor	**Average**	**Feature Time**	**Coding Time**	**Kernel Time**	**Memory**
NSQ	67	56	41	66	21	56	72	55	41	48	42	37	71	65	78	21	45	53	71	51	52.87	0.85	0.86	0.08	42K
HA	67	56	44	66	22	57	75	56	44	50	43	37	74	69	78	24	46	56	74	51	54.54	0.85	21.22	0.08	42K
LLC	69	58	46	70	21	60	74	57	43	52	44	42	73	68	81	24	48	57	72	52	55.56	0.85	23.81	–	1344K
LLC Wang et al. [2010]	75	65	51	71	29	69	79	62	54	49	52	44	77	67	84	31	45	53	79	54	59.30	–	–	–	–
S. V. [Zhou et al., 2010]	79	72	55	74	34	72	83	64	57	53	63	49	81	72	85	36	46	60	83	59	64.00	–	–	–	–

Table 4.2: PASCAL VOC 2007 classification results. *The average score provides the per-class average. The time and memory measurements are for one image. We reproduced all experiments, expect for the last two rows, where LLC results are extracted from [Wang et al., 2010], and for S.V. from [Zhou et al., 2010].*

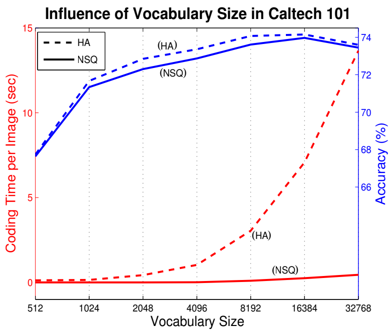


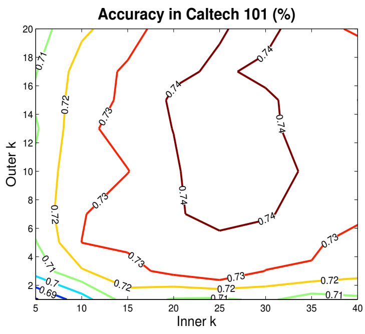
***Figure 4.2**:* Influence of the parameter k and the vocabulary size in Caltech 101. *Comparison of performance and time between NSQ and HA when varying the size of the codebook.*

the accuracy becomes. Interestingly, the computational time for HA explodes when increasing the codebook size, while in NSQ the increase is negligible compared to HA.

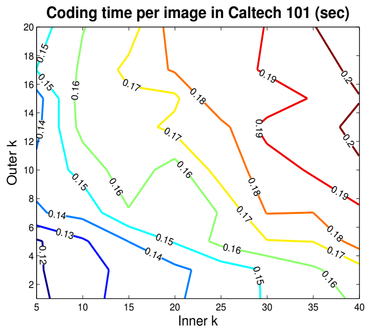
We analyze the accuracy and time performance of our method when changing the quantization parameters. We use the first split of Caltech 101 to show the results. In Figures 4.3a and 4.3b, we compare the accuracy of our method when changing the k parameter for the inner quantization and the outer quantization, and keeping the codebook size at $8,192$. We see a peak of performance: When the inner k is around 20 and 35 and the outer k is around 10 and 16. In the right plot we analyze the time when changing this parameters, also keeping the codebook at size $8,192$. The lower both k are, the less time classification takes. The best compromise between time and performance is using $k' = 25$ for the inner quantization, and $k = 10$ for the outer.

Influence of the Quantization Error

We analyze the impact of having more quantization error due to the inner quantization. We use Caltech 101 to show results using $8,192$ codebook entries. In Figure 4.4a, we show that the quantization error of the inner quantization does not have a clear relation with the classification accuracy. The accuracy is maximal for k' at 20 and 35, and the quantization error has its minimum around

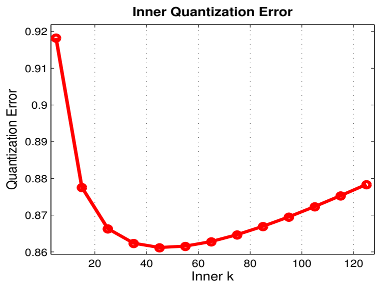


(a)

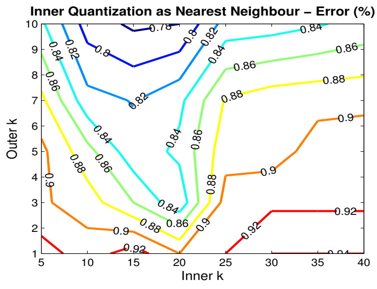


(b)

***Figure 4.3**: Influence of the parameter k and the vocabulary size in Caltech 101. (a) Accuracy and (b) Time when varying the inner and outer quantization parameters.*



(a)



(b)

***Figure 4.4**: (a) Quantization error of the inner quantization vs k', in Caltech 101. (b) Evaluation of the inner quantization as a nearest neighbor on Caltech 101 changing k and k'.*

45. Moreover, we show the percentage of equal activated elements of $\boldsymbol{\alpha}$ in NSQ and in HA using the same $\mathbf{B}$, in Figure 4.4b. Around 90% of the nearest neighbors are different. This shows that NSQ is a new encoding and not an approximate nearest-neighbor approach.

	NSQ	**HA**
Time Dataset		
Features	17h 21m	17h 21m
Coding	6h 54m	7h 9m
Total	24h 18m	24h 30m
SizeDataset	5.4GB	700MB
Acc.Top-1	20.01%	16.6%

Table 4.3: Quantitative results on ImageNet. *To fulfill the time constraints, NSQ uses a codebook of* $2,048$ *and HA of* 256 .

ImageNet in a Laptop in one Day

We introduce a new benchmark to test the efficiency of the encodings. We used ImageNet with restrictions on time and memory, and only let the image encoding algorithms run about 24 hours on a laptop of 4 CPUs. We compared our method with HA with max pooling in which we adjusted the parameters to fulfill the requirements. With NSQ we were able to use a codebook of $2,048$ entries, and in HA only of 256. In Table 4.3 the results are summarized. In this benchmark, we outperform HA because in the same amount of time NSQ can use a larger codebook, which is crucial to have good classification performance. The reported results are of the same order of magnitude as reported in [Deng et al., 2010], though a direct comparison is not possible because we use a different subset of ImageNet. The state-of-the-art for this dataset is reported by Lin et al. [2011], but the computational effort of this method is huge compared to ours.

4.6 Experiments in Image Semantic Segmentation

Once we have tested the efficiency and efficacy of NSQ in standard object recognition benckmarks, we now report results when using NSQ to encode local regions for semantic segmentation. We use MSRC-21 [Shotton et al., 2009], which is a standard semantic segmentation dataset. We first describe the implementation details, and then report the results.

4.6.1 Implementation Details

The components for semantic segmentation used in this experiment are described below.

Superpixel Segmentation. The images are first over-segmented using the SEEDS superpixel algorithm [Van den Bergh et al., 2014]. The MSRC-21 images are segmented with about 300 superpixels, as reported in [Roig et al., 2013].

Unary Potentials. We use the features and classifiers reported in [Lucchi et al., 2011], which are based on color histograms and SIFT descriptors, but only using SIFT descriptor. We compare the encoding of the SIFT descriptor using Bag-of-Words (used in [Lucchi et al., 2011]), and NSQ.

Inference. We take the maximum score for each superpixel, obtained by the one-versus-rest classifiers, which was reported by Lucchi et al. [2011] to give comparable results than using more complex probabilistic models.

4.6.2 Results

In Figure 4.5 we show the results when changing the amount of quantization in the feature encoding with NSQ, and in Table 4.4 the results of NSQ compared to Bag-of-Word encoding. We can see that we obtain comparable results with both encodings, but as reported in the previous section, NSQ is much more efficient to compute.

Encoding Method	accuracy
Bag-of-Words	75.200%
NSQ ($k' = 15\%, k = 5$)	75.045%

***Table 4.4**: Semantic segmentation accuracy on MSRC-21. Comparision between NSQ and Bag-of-Words feature encoding.*

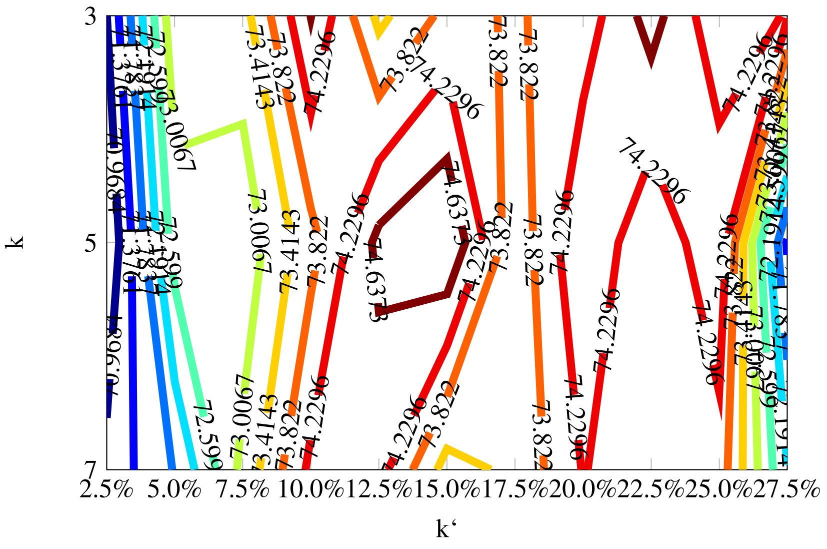


Figure 4.5: *Semantic Segmentation accuracy on MSRC-21 with NSQ feature coding, changing the inner and outer quantization parameter.*

4.7 Conclusion

We presented a novel encoding formulation based on Sparse Quantization that allows to fuse Sparse Coding and Assignment-based Coding schemes. We investigated efficient feature encoding schemes for object recognition using binary quantizations with the new formulation. We proposed a very efficient encoding algorithm called Nested Sparse Quantization (NSQ), which is based on two nested quantizations. The implementation of NSQ is very simple, and can virtually be incorporated everywhere where Assignment-based Coding schemes are employed, but higher efficiency and more compact representations are demanded, *e.g.* real-time and large scale recognition problems. This potential was corroborated by our experiments on the Caltech 101, PASCAL VOC 07 and ImageNet benchmarks. Finally, we tested NSQ in image segmentation framework, and we obtained similar accuracy performance with high gains in efficiency.

5

Active MAP Inference for Efficient Semantic Segmentation

5.1 Introduction

In many state-of-the-art methods for semantic segmentation, contextual information plays a central role. A successful trend has been to encode the contextual constraints with a Conditional Random Field (CRF) [Lafferty et al., 2001], by modeling the interactions between different regions and scales of the image. Most methods use sophisticated potentials between different neighboring regions [Gould et al., 2008; Verbeek & Triggs, 2007], and the state-of-the-art has been boosted with the use of high-order potentials in hierarchical CRFs [Boix et al., 2012; Kohli et al., 2009; Plath et al., 2009].

Another common way to include contextual information has been to extend image descriptors with contextual cues [Fulkerson et al., 2009; Jiang & Tu, 2009; Pantofaru et al., 2008], or also, combining semantic classifiers fed from different contextual features [Csurka & Perronnin, 2010; Maire et al., 2011; Munoz et al., 2010]. It is a remarkable feat the balance achieved between accuracy and efficiency by the semantic texton forests of Shotton et al. [2009]. The good performance exhibited by many methods that do not benefit from introducing context to a CRF, lead Lucchi et al. [2011] to ask the provocative question: 'Are spatial and global constraints really necessary for segmentation?' From the experimental results, they conclude that the CRF structures boost performance when the features only encode local information, whereas the further gain is very little when the features already encode contextual information. This begs the question whether we can really benefit from CRFs in

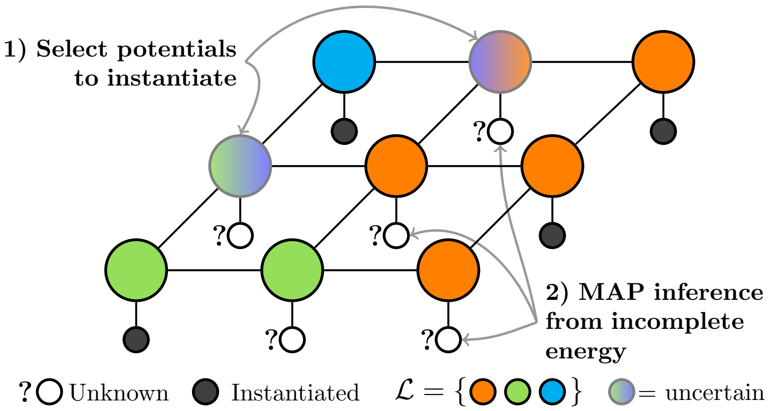

1) Select potentials to instantiate
2) MAP inference from incomplete energy
? Unknown
Instantiated
$\mathcal{L} = \{$ ● ● ● $\}$
= uncertain

***Figure 5.1**:* Active MAP inference (best seen in color). *Example of CRF with unknown unary potentials. Active MAP selects the potentials to instantiate that maximize the expected reward. Also, it estimates the MAP labeling from the incomplete energy function.*

semantic segmentation when using such powerful features that already encode context.

We present a novel use of CRFs for semantic segmentation. We exploit CRFs to estimate the semantic labeling without computing the descriptors and classifiers everywhere in the image. Given a budget of time, our algorithm decides which potentials to compute. In doing so, it dramatically reduces the computational complexity of the whole pipeline. This is because of the computational burden of instantiating the potentials that extract descriptors and apply classifiers, which can be much higher than MAP inference for most of the energy functions in the literature [Boix et al., 2012; Fulkerson et al., 2009; Lucchi et al., 2011].

We introduce a relation between CRFs with some unknown unary potentials, which correspond to the features and classifiers that we do not compute, and the Perturb-and-MAP (PM) random field model [Papandreou & Yuille, 2011]. We build our MAP inference algorithm - coined Active MAP inference - based on this finding. We use the term 'active' because during inference it selects which potentials to instantiate *on-the-fly*. This stands in contrast to previous MAP

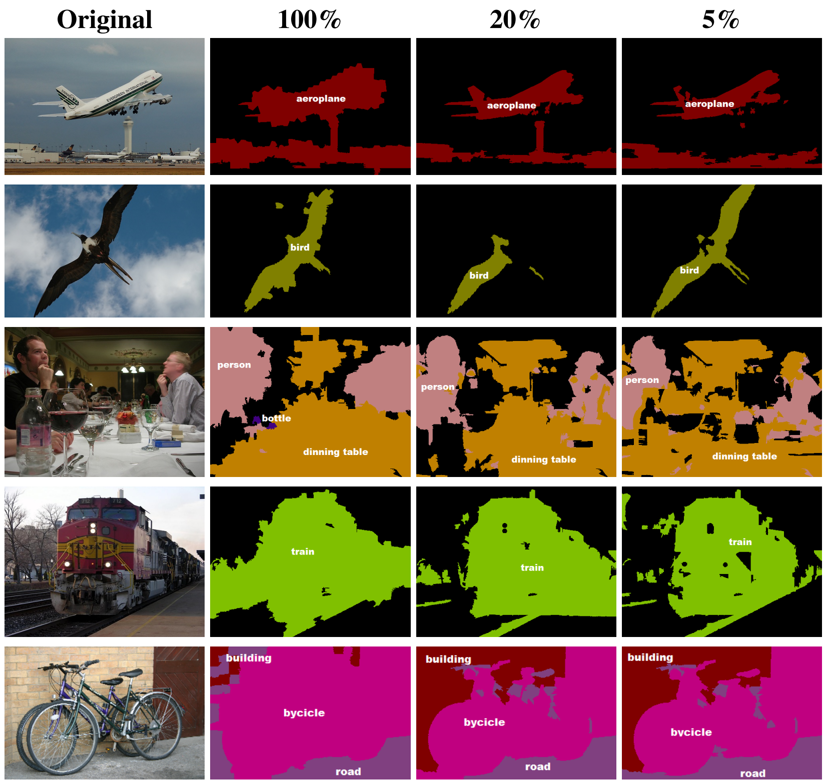


Figure 5.2: Examples of segmented images on VOC10 and MSRC-21. *Active MAP inference using different percentages of instantiated unary potentials. Results are obtained by selecting the unary potentials with the expected labeling change.*

inference methods, which first execute the features/classifiers that instantiate the CRF, and then run the MAP-CRF inference. In Figure 5.1 we illustrate the principles of Active MAP inference. Surprisingly, seeing the instantiation of the CRF energy function and MAP-CRF inference as two joint steps received little attention in the community.

In a serie of experiments, we show that active MAP inference successfully exploits spatial consistency to avoid evaluating the classifiers and features everywhere. It obtains comparable results to instantiating all the potentials in the

CRF for the PASCAL VOC 2010 segmentation challenge [Everingham et al., 2010] and for the MSRC-21 dataset [Shotton et al., 2009], but with major efficiency gains. In Figure 5.2 we illustrate some results on semantic segmentation obtained with active MAP inference.

5.2 Active MAP Inference in CRFs

This Section describes the approach for active MAP inference. Its formulation uses a CRF to model the probability density distribution expressing the likeliness of a certain labeling. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph that represents such distribution, and $\mathbf{X}$ the set of random variables or nodes of the graph. The elements of $\mathcal{V}$ are indices of the nodes, *i.e.* $\mathbf{X} = \{X_i\}$ in which $i \in \mathcal{V}$, and the elements of $\mathcal{E}$ are the indices of the undirected edges of the graph. We denote an instance of the random variables as $\mathbf{x} = \{x_i\}$, where x_i takes a value from a set of discrete labels $\mathcal{L}$. Thus, $\mathbf{x} \in \mathcal{L}^N$, with N the cardinality of $\mathcal{V}$.

We denote $P(\mathbf{x}|\boldsymbol{\theta})$ as the probability density distribution of a labeling modeled with the graph $\mathcal{G}$. According to the Hammersley-Clifford theorem (*cf.* [Kolmogorov & Wainwright, 2005]), the probability density that satisfies the Markov properties with respect to the graph $\mathcal{G}$ is a Gibbs distribution. Thus, $P(\mathbf{x}|\boldsymbol{\theta})$ can be written as the normalized negative exponential of an energy function

$$E_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}), \tag{5.1}$$

in which $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(x), \ldots, \phi_M(x))^T$ is the vector of potentials, or the so-called sufficient statistics, and $\boldsymbol{\theta} \in \mathbb{R}^M$ are the parameters of the potentials. We use the *canonical over-complete representation*, in which $\{\phi_i(\mathbf{x})\}$ are built using indicator functions that allow us to express the energy function as such linear combination of the potentials (*cf.* [Wainwright et al., 2005]). The most probable state $\mathbf{x}^\star$ is obtained by inferring the Maximum a Posteriori (MAP) of $P(\mathbf{x}|\boldsymbol{\theta})$, or equivalently by minimizing the energy, *i.e.*

$$\mathbf{x}^\star = \arg \min_{\mathbf{x} \in \mathcal{L}^N} \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}). \tag{5.2}$$

As usual, we categorize the potentials of the energy function depending on the number of random variables that they involve: unary and pairwise.

In the case of semantic segmentation, there is a node defined for each pixel or superpixel in the image. The parameters of $\boldsymbol{\theta}$ related to the unary potentials are typically the result of evaluating classifiers fed with features extracted from the image. The pairwise and high-order potentials use some *a priori* assumptions like the smoothness of the labeling. It is important to note that the instantiation of $\boldsymbol{\theta}$ might be orders of magnitude more computationally expensive than MAP inference. Usually, state-of-the-art methods for semantic segmentation use features and classifiers that take minutes to compute for a single image [Fulkerson et al., 2009; Lucchi et al., 2011; Boix et al., 2012].

At testing phase, the common way to proceed is to instantiate $\boldsymbol{\theta}$, and then to run an off-the-shelf MAP inference algorithm to obtain the most probable labeling. Active MAP inference aims at estimating $\mathbf{x}^\star$ with only a subset of the elements of $\boldsymbol{\theta}$, $\{\theta^j\}$, which is selected by the algorithm. The computational gain comes from not computing all classifiers and features needed to fully instantiate $\boldsymbol{\theta}$. Even though we do not have the complete energy function anymore because part of $\boldsymbol{\theta}$ is unknown, we will show in the sequel that we can still estimate $\mathbf{x}^\star$. We define $\boldsymbol{\delta} \in \{0, 1\}^M$, with the purpose of introducing the concept of selected parameters in our notation, *i.e.* it works as an indicator function. When the element j of the vector $\boldsymbol{\theta}$, *i.e.* θ^j, is not computed, then, δ^j is zero, and if the parameter is computed, then δ^j equals 1. This is

$$\theta^j_{\delta^j} = \begin{cases} \theta^j & \text{if } \delta^j = 1 \\ \text{unknown} & \text{otherwise} \end{cases} . \tag{5.3}$$

Note that with this notation we can still easily express the initial formulation that instantiates all parameters, using $\boldsymbol{\delta} = \mathbf{1}$ and $\boldsymbol{\theta}_{\mathbf{1}}$, where $\mathbf{1}$ is a vector of ones.

With missing parameters, the energy function does not represent the initial labeling problem anymore. It would be wrong to replace the unknown parameters by 0, or any value indicating that 'the potential is missing'. There is no guarantee that, in doing so, the new energy function would assign energy values similar to the ones given by the complete energy.

General Overview

Given a time budget, Active MAP inference instantiates a subset of the potentials ($\boldsymbol{\delta}$), and only with them, it computes the complete MAP labeling, $\mathbf{x}^\star$. In the following Section, we introduce Perturb-and-MAP, as we use this mathematical tool in the rest of this Chapter. In Section 5.4, we introduce the estimation of $\mathbf{x}^\star$ when $\boldsymbol{\delta}$ is given, and in Section 5.5, we introduce the algorithm to determine $\boldsymbol{\delta}$. Finally, we show results for the application of semantic image segmentation, where we save the cost of instantiating all the unary potentials. Active MAP inference is more general and can also be applied in many other applications that use a CRF.

5.3 Perturb-and-MAP

Generating samples from CRFs is unusual in computer vision. For most problems, sampling over the discrete space of the CRF is prohibitive due to the complexity of these spaces. Recently, Papandreou & Yuille [2011] introduced the Perturb-and-MAP random field (PM), which is a model that allows for generating samples, built around the effective MAP inference algorithms in CRF. In a follow-up paper, Tarlow et al. [2012] extended this idea to a more broader set of models.

PM is based on injecting noise in the energy function to perturb it, and then, it calculates the frequency that labellings are the MAP of the perturbed energy. Let $\boldsymbol{\epsilon} \in \mathbb{R}^M$ be the random variable that it is used to perturb the parameters of the energy function, and let $f_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ be the probability density of $\boldsymbol{\epsilon}$. We denote the perturbed parameters of the energy as $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + \boldsymbol{\epsilon}$. For each perturbed $\tilde{\boldsymbol{\theta}}$, we can infer a MAP labeling. The different $\tilde{\boldsymbol{\theta}}$s that yield the same MAP labeling $\mathbf{x}$, can be grouped together. We use $\mathcal{P}_{\mathbf{x}}$ to denote such set of $\tilde{\boldsymbol{\theta}}$s,

$$\mathcal{P}_{\mathbf{x}} = \left\{ \tilde{\boldsymbol{\theta}} \in \mathbb{R}^M | \mathbf{x} = \arg\min_{\mathbf{x}' \in \mathcal{L}^N} \tilde{\boldsymbol{\theta}}^T \boldsymbol{\phi}(\mathbf{x}') \right\}. \tag{5.4}$$

Analogously, we can define the set of perturbations $\boldsymbol{\epsilon}$, that yields the labeling $\mathbf{x}$ when doing MAP inference. We denote this set as $\mathcal{P}_{\mathbf{x}} - \boldsymbol{\theta}$, and it is

$$\left\{ \boldsymbol{\epsilon} \in \mathbb{R}^M | \mathbf{x} = \arg\min_{\mathbf{x}' \in \mathcal{L}^N} (\boldsymbol{\theta} + \boldsymbol{\epsilon})^T \boldsymbol{\phi}(\mathbf{x}') \right\}. \tag{5.5}$$

PM assigns a probability to $\mathbf{x}$ equal to the probability of drawing a perturbation $\boldsymbol{\epsilon}$ that belongs to the set $\mathcal{P}_{\mathbf{x}} - \boldsymbol{\theta}$. Thus, the PM distribution is

$$f_{PM}(\mathbf{x}|\boldsymbol{\theta}) = \int_{\mathcal{P}_{\mathbf{x}} - \boldsymbol{\theta}} f_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}. \tag{5.6}$$

Intuitively, the PM calculates how frequent is that a labeling $\mathbf{x}$ is the MAP labeling, when injecting noise to the energy function. Even though calculating the exact value of $f_{PM}(\mathbf{x};\boldsymbol{\theta})$ might be not feasible for most practical cases, note that we can easily draw samples from a PM distribution by simply doing MAP inference on a perturbed energy. For a complete explanation of the PM random field we refer to the paper [Papandreou & Yuille, 2011].

5.4 MAP Inference for Incomplete Energies

This Section aims at estimating the labeling from the incomplete energy function. We assume that $\boldsymbol{\delta}$ is given, and the potentials indicated by $\boldsymbol{\delta}$ have been instantiated.

5.4.1 Relation to Perturb-and-MAP

Rather than filling in the energy function by inventing the unknown parameters or setting them to a learned constant value, we use $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ to model them. $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ is the probability that the parameters of the potentials take the values $\boldsymbol{\theta}$ given $\boldsymbol{\theta}_{\boldsymbol{\delta}}$. The CRF models the probability of the labeling, but it does not directly model $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$. In order to alleviate the lack of an exact expression for $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$, we use a model to approximate it, referred to as $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta},\boldsymbol{\pi})$, where $\boldsymbol{\pi}$ are the parameters of the model. The definition of this model is open and adaptable to each problem. We specify $f_{\boldsymbol{\theta}}$ and $\boldsymbol{\pi}$ in the subsequent Section.

Changing $\boldsymbol{\theta}$ in the energy function produces different MAP labelings, $\mathbf{x}^\star$. Therefore, $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ induces a probability on $\mathbf{x}^\star$. We use $P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ to define such probability on $\mathbf{x}^\star$, *i.e.* the probability that $\mathbf{x}$ is the MAP labeling. It can be computed as

$$\int_{\mathbb{R}^M} \mathbf{I}\left[\mathbf{x} = \arg\min_{\mathbf{x}'\in\mathcal{L}^N} E_{\boldsymbol{\theta}}(\mathbf{x}')\right] P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})d\boldsymbol{\theta}, \tag{5.7}$$

where $\mathbf{I}[\,\cdot\,]$ is the indicator function. Equation (5.7) can be seen as a natural way to calculate $P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$, since it accumulates the probability density of $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ with $\boldsymbol{\theta}$ yielding the minimum energy labeling equal to $\mathbf{x}$. The integral explores all complete energy functions, $E_{\boldsymbol{\theta}}(\mathbf{x})$, and for each of them, it checks whether the MAP labeling is $\mathbf{x}$ or not. In case it is equal to $\mathbf{x}$, the corresponding probability density of $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ is accumulated into the final probability.

Deriving the exact $P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ is computationally intractable, because of the number and complexity of the constraints needed to define $E_{\boldsymbol{\theta}}$. Fortunately, it can be shown that $P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ is indeed a PM random field, from which we can easily draw samples. We state it formally in the following Proposition.

Proposition 5. *Let $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}}) = f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta},\boldsymbol{\pi})$, and $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta},\boldsymbol{\pi})$ has mean equal to $\boldsymbol{\mu} \in \mathbb{R}^M$. Let $f_{PM}(\mathbf{x}|\boldsymbol{\delta},\boldsymbol{\mu})$ be the density distribution of a PM model with energy $E_{\boldsymbol{\mu}}(\mathbf{x})$,* i.e. *the energy with parameter $\boldsymbol{\mu} \in \mathbb{R}^M$, and the perturbations are drawn from $\boldsymbol{\epsilon} \sim f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}+\boldsymbol{\mu}|\boldsymbol{\delta},\boldsymbol{\pi})$. Then,*

$$P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}}) = f_{PM}(\mathbf{x}|\boldsymbol{\delta},\boldsymbol{\mu}). \tag{5.8}$$

Proof. With few calculus we obtain:

$$P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}}) =$$

$$= \int_{\mathbb{R}^M} \mathbf{I}\left[\mathbf{x} = \arg\min_{\mathbf{x}'\in\mathcal{L}^N} E_{\boldsymbol{\theta}}(\mathbf{x}')\right] P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}})d\boldsymbol{\theta} \tag{5.9}$$

$$= \int_{\mathbb{R}^M} \mathbf{I}\left[\mathbf{x} = \arg\min_{\mathbf{x}'\in\mathcal{L}^N} E_{\boldsymbol{\theta}}(\mathbf{x}')\right] f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta},\boldsymbol{\pi})d\boldsymbol{\theta} \tag{5.10}$$

$$= \int_{\mathbb{R}^M} \mathbf{I}\left[\mathbf{x} = \arg\min_{\mathbf{x}'\in\mathcal{L}^N} E_{(\boldsymbol{\mu}+\boldsymbol{\epsilon})}(\mathbf{x}')\right] f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}+\boldsymbol{\mu}|\boldsymbol{\delta},\boldsymbol{\pi})d\boldsymbol{\epsilon} \tag{5.11}$$

$$= \int_{\mathcal{P}_{\mathbf{x}}-\boldsymbol{\mu}} f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}+\boldsymbol{\mu}|\boldsymbol{\delta},\boldsymbol{\pi})d\boldsymbol{\epsilon} \tag{5.12}$$

$$= f_{PM}(\mathbf{x}|\boldsymbol{\delta},\boldsymbol{\mu}), \text{with energy: } E_{\boldsymbol{\mu}}(\mathbf{x}). \tag{5.13}$$

$\mathbf{I}[\cdot]$ is the indicator function. In Equation (5.9), we use the definition of $P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ in Equation (5.7). In Equation (5.10), we introduce the assumption we made in the Proposition that $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}}) = f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta},\boldsymbol{\pi})$. In Equation (5.11), we do the change of variable $\boldsymbol{\theta} = \boldsymbol{\mu}+\boldsymbol{\epsilon}$. Finally, in Equations (5.12) and (5.13), we use the definitions of the PM model, in which

$$\mathcal{P}_{\mathbf{x}} - \boldsymbol{\mu} = \left\{\boldsymbol{\epsilon} \in \mathbb{R}^M | \mathbf{x} = \arg\min_{\mathbf{x}'\in\mathcal{L}^N} (\boldsymbol{\mu}+\boldsymbol{\epsilon})^T \boldsymbol{\phi}(\mathbf{x}')\right\}. \tag{5.14}$$

□

Observe that the density distribution of the PM model in Proposition 5 is

$$f_{PM}(\mathbf{x}|\boldsymbol{\delta},\boldsymbol{\mu}) = \int_{\mathcal{P}_{\mathbf{x}}-\boldsymbol{\mu}} f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}+\boldsymbol{\mu}|\boldsymbol{\delta},\boldsymbol{\pi})d\boldsymbol{\epsilon}, \tag{5.15}$$

where $\mathcal{P}_{\mathbf{x}} - \boldsymbol{\mu}$ is the set of $\boldsymbol{\epsilon} \in \mathbb{R}^M$ such that $\mathbf{x}$ minimizes the energy function $E_{(\boldsymbol{\mu}+\boldsymbol{\epsilon})}$ (see Equation (5.4)). Note also that we draw $\boldsymbol{\epsilon}$ from $f_{\boldsymbol{\theta}}(\boldsymbol{\epsilon} + \boldsymbol{\mu}|\boldsymbol{\delta}, \boldsymbol{\pi})$, which is $f_{\boldsymbol{\theta}}$ centered at $\mathbf{0}$. Proposition 5 shows that this PM distribution reproduces the definition of $P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ in Equation (5.7). To obtain samples of $\mathbf{x}^\star$ in practice, we simply perturb $\boldsymbol{\mu}$ using $\boldsymbol{\epsilon}$, and then, we apply MAP inference to $E_{(\boldsymbol{\mu}+\boldsymbol{\epsilon})}(\mathbf{x})$.

Note that Proposition 5 is valid for any $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta}, \boldsymbol{\pi})$. Yet, the key assumption in Proposition 5 is $P(\boldsymbol{\theta}|\boldsymbol{\theta}_{\boldsymbol{\delta}}) = f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta}, \boldsymbol{\pi})$, which presupposes an underlying model for the known and unknown θ. This is addressed in the following.

5.4.2 Model of the Missing Parameters

We use a simple collection of independent Gaussian variables to define $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta}, \boldsymbol{\pi})$. The parameters for this model are the mean and the standard deviation, referred to as $\boldsymbol{\mu} \in \mathbb{R}^M$ and $\boldsymbol{\sigma} \in \mathbb{R}^M$ respectively, where for notation simplicity $\boldsymbol{\pi}$ indicates both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. We use the standard Gaussian distribution due to its simplicity and its well-known properties. Specifically, we define $f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{\delta}, \boldsymbol{\pi})$ such that, if the parameter of the potential is unknown ($\delta^i = 0$), it is a univariate Gaussian distribution, centered at μ^i and deviation σ^i. Otherwise it is consistent with the instantiated potential, $f_\theta(\theta^i|\delta^i = 1, \pi^i) = \mathbf{I}[\theta^i = \theta^i_{\delta^i}]$, where $\mathbf{I}[\,\cdot\,]$ is the indicator function. In this latter case, there is no uncertainty, and π^i and σ^i are not used.

We set $\boldsymbol{\pi}$ to a fixed value that we learn by cross-validation. Thus, all $f_\theta(\theta^i|\delta^i = 0, \pi^i)$ are a Gaussian distribution with the same parameters. We could find the more likely $\boldsymbol{\pi}$ given the observations, but it is out of the scope of this work. From a practical perspective, it suffices to assume that $\boldsymbol{\pi}$ takes a fixed value to achieve good performance in practice.

5.5 Selection of Potentials

In this Section we describe the selection of the potentials, indicated in $\boldsymbol{\delta}$. The algorithm starts from $\boldsymbol{\delta} = \mathbf{0}$, and it sequentially determines which potential to compute next, until the time budget, t_{total}, expires. We denote the known potentials at time t as $\boldsymbol{\theta}_{\boldsymbol{\delta}_t}$. The algorithm ranks the unknown potentials with a score, and thus prioritizes the potentials in the time budget. This is done by

Algorithm 7: Active MAP

```
$\boldsymbol{\delta}_0 = \mathbf{0}$;
while $t<t_{total}$ do
    ▷ Compute the score for the Unkown Unary Potentials:
    forall the $\delta_t^i = 0$ do
        $S^i_{\boldsymbol{\delta}_t} = \mathbb{E}_\theta \left[R\left(f_{PM}\left(\mathbf{x}|\boldsymbol{\delta}_t : \delta^i = 1, \boldsymbol{\pi}_t : \mu^i = \theta\right)\right)\right]$
    end
    ▷ Instantiate the Unary Potential with higher $S^i_{\boldsymbol{\delta}_t}$:
    $i^\star = \arg\max_i S^i_{\boldsymbol{\delta}_t}$
    $\delta^{i^\star} = 1$, Compute $\theta^{i^\star}$
end
$\mathbf{x}^\star = \arg\max_{\mathbf{x}} f_{PM}(\mathbf{x}|\boldsymbol{\delta}, \boldsymbol{\mu})$
```

selecting the potentials with higher score. We summarize all steps in Algorithm 7.

Let $S^i_{\boldsymbol{\delta}_t}$ be the score that ranks the potentials. We define $S^i_{\boldsymbol{\delta}_t}$ as the expected reward of instantiating the potential i. This is

$$S^i_{\boldsymbol{\delta}_t} = \mathbb{E}_\theta \left[R\left(P\left(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}_t} : \theta^i = \theta\right)\right)\right], \tag{5.16}$$

where the expected value is over $\theta \sim f_\theta\left(\theta^i|\delta^i = 0, \pi^i\right)$, which is the Gaussian model of the posterior $P(\theta^i|\boldsymbol{\theta}_{\boldsymbol{\delta}_t})$. We use $\boldsymbol{\theta}_{\boldsymbol{\delta}_t} : \theta^i = \theta$ to indicate that θ^i in $\boldsymbol{\theta}_{\boldsymbol{\delta}_t}$ has been set to θ. $R(\,\cdot\,)$ is the reward of instantiating $\theta^i = \theta$, and it evaluates the probability distribution of $\mathbf{X}^\star$. The reward prioritizes probability distributions using a pre-defined criterion, such as having low uncertainty in the labeling of $\mathbf{X}^\star$. There are different possible criterion to define it, and we analyze two of them in the sequel. Observe that Equation (5.16) evaluates the expected value of the reward by sampling θs from $f_\theta(\theta^i|\delta^i = 0, \pi^i)$, and evaluating the reward we would get if θ^i is clamped to the sampled θ.

We can further develop $S^i_{\boldsymbol{\delta}_t}$ in Equation (5.16). According to Proposition 5,

$$P\left(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}_t} : \theta^i = \theta\right) \tag{5.17}$$

is a PM, which is $f_{PM}\left(\mathbf{x}|\boldsymbol{\delta}_t : \delta^i = 1, \boldsymbol{\pi}_t : \mu^i = \theta\right)$. Thus, $S^i_{\boldsymbol{\delta}_t}$ becomes

$$S^i_{\boldsymbol{\delta}_t} = \mathbb{E}_\theta \left[R\left(f_{PM}\left(\mathbf{x}|\boldsymbol{\delta}_t : \delta^i = 1, \boldsymbol{\pi}_t : \mu^i = \theta\right)\right)\right], \tag{5.18}$$

Below, we introduce two possible criteria for the expected reward, and analyze the computational cost of calculating the reward.

5.5.1 Expected Reward

To compute the reward, we adapt two standard techniques from the active learning literature [Settles, 2009], namely the residual entropy and the labeling change. In the following we discuss them in the context of active MAP inference, and we show that both criteria can be effectively computed from a set of samples derived from the PM.

Expected Residual Entropy (ERE)

We can compute the reward using the residual entropy in order to reduce the uncertainty of the MAP labeling. Then, the reward $R(\,\cdot\,)$ becomes

$$-H\left(f_{PM}\left(\mathbf{x}|\boldsymbol{\delta}_t : \delta^i = 1, \boldsymbol{\pi}_t : \mu^i = \theta\right)\right), \tag{5.19}$$

where $H(\,\cdot\,)$ is the entropy, and can be computed by drawing samples from the PM. Note that reducing the uncertainty of the MAP labeling does not necessarily mean that the labeling is closer to the true MAP labeling.

Expected Labeling Change (ELC)

Vezhnevets et al. [2012] proposes to evaluate the expected change in the labeling. In the case of our problem, it is the change in the labeling induced from instantiating a potential. Thus, the reward $R(\,\cdot\,)$ is

$$\Delta\left(\mathbf{x}_t^\star, \arg\max_{\mathbf{x}} f_{PM}(\mathbf{x}|\boldsymbol{\delta}_t : \delta^i = 1, \boldsymbol{\pi}_t : \mu^i = \theta)\right), \tag{5.20}$$

where $\mathbf{x}_t^\star$ is the MAP labeling at iteration t, and $\Delta(\,\cdot\,,\,\cdot\,)$ is a function that counts how many labels of $\mathbf{x}_t^\star$ differ from the labeling that we obtain with the PM when instantiating $\theta^i = \theta$.

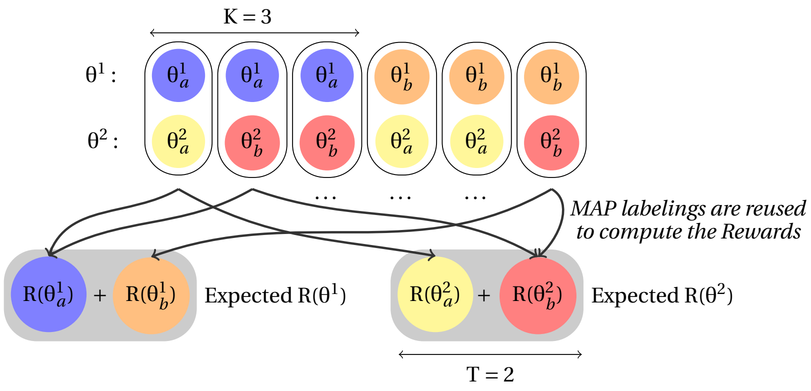


***Figure 5.3**:* Illustrative example of the TK samples of $\boldsymbol{\theta}$ to compute the expected Reward for all unknown potentials (best seen in color). *Example with* 2 *unknown potentials,* $T = 2$ *and* $K = 3$.

5.5.2 Efficient Computation of the Reward

We can see by analyzing Equation (5.18), that in the calculation of the expected reward, there are TK computations of MAP inference, where T is the number of samples of θ^i, and K the number of samples of the PM. This is because for the T samples of θ^i, we evaluate a PM that computes MAP K times. Thus, the cost of computing the scores for a number U of unknown potentials is $O(TKUm)$, where m is the cost of inferring the MAP labeling.

According to Algorithm 7, the scores are evaluated every time we instantiate a potential. Thus, if doing TKU times MAP has a comparable cost to instantiate one potential, rather than speeding up the whole pipeline, Active MAP may become the computational bottleneck. In the following, we introduce two complementary strategies that render the evaluation of the scores efficient in practice.

Efficient computation of the expected reward

We first introduce a strategy to reduce $O(TKUm)$ to $O(TKm)$. It is based on the observation that the PM draws the unknown parameters of the energy from $f_{\boldsymbol{\theta}}\left(\theta^i|\delta^i = 0, \pi^i\right)$, which is the same probability distribution that we use to generate the θs for the expected value. Thus, we could reuse the same samples of $\boldsymbol{\theta}$ to calculate both the expected value, and the energy function of the different perturbations of the PM.

Recall that in the expected reward, for each θ^i, PM computes MAP inference K times with μ^i fixed to θ^i. We can generate a set of TK samples that can be used for all PMs of any unknown potential. This is feasible by drawing T values of θ^i from $f_{\boldsymbol{\theta}}$, and extending them, by repeating those T initial values of θ^i by K times, in a random order. We do this for all unknown potentials and obtain a set of TK different vectors of $\boldsymbol{\theta}$. Figure 5.3 is an example of $\boldsymbol{\theta}$s generated in that way. Note that for each value of θ^i, we always have K different samples of $\boldsymbol{\theta}$, having the unknown potentials perturbed and θ^i fixed. This coincides with the form of the energy function of the K perturbations of the PM.

The limitation of this method is that θ^i takes only T different values in the TK samples, and they might not be diverse enough to correctly estimate the reward. Yet, we observed in the experiments that this approach achieves the same performance as using TKU different samples, even with small K and T.

Area of Influence

We propose a simple strategy to avoid re-computing the scores every time we instantiate a potential. It is summarized in Algorithm 8. It is assumed that instantiating a potential reduces the score of the potentials that are in its "area of influence", while the rest remain unchanged. Under this assumption, only the scores in the area of influence are unreliable if they are not recomputed. We discard such scores as candidates until we re-compute the scores. This is done at the point that all potentials have been discarded.

We define an heuristic way, yet effective, to compute the area of influence. We define the area of influence as set of nodes that in all samples drawn from $P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$ take the same labeling value, and form a connected blob in the image. In Figure 5.4 we illustrate several examples of the estimated area of influence for some potentials, and in the experiments section we show that using the area of influence is as effective as not using it, but it yields dramatic speed ups. Note that computing the area of influence does not incorporate any

Algorithm 8: Active MAP with Area of Influence

```
$\boldsymbol{\delta}_0 = \mathbf{0}$;
while $t<t_{total}$ do
    ▷ Compute the score for the Unknown Unary Potentials:
    forall the $\delta_t^i = 0$ do
        $S^i_{\boldsymbol{\delta}_t} = \mathbb{E}_\theta \left[R\left(f_{PM}\left(\mathbf{x}|\boldsymbol{\delta}_t : \delta^i = 1, \boldsymbol{\pi}_t : \mu^i = \theta\right)\right)\right]$
    end
    while $\exists S^i_{\boldsymbol{\delta}_t} \neq -\infty$ do
        ▷ Instantiate the Unary Potential with higher $S^i_{\boldsymbol{\delta}_t}$
        $i^\star = \arg\max_i S^i_{\boldsymbol{\delta}_t}$
        $\delta^{i^\star} = 1$, Compute $\theta^{i^\star}$
        ▷ Delete Candidates from the Area of Influence:
        forall the $x^j \in$ A. Infl. $(\delta^i = 1, \{\mathbf{x}\}_{K^2})$ do
            $S^i_{\boldsymbol{\delta}_t} = -\infty$
        end
    end
end
$\mathbf{x}^\star = \arg\max_{\mathbf{x}} f_{PM}(\mathbf{x}|\boldsymbol{\delta}, \boldsymbol{\mu})$
```

extra major cost, since we can use the samples of the PM used for calculating the rewards.

5.6 Experiments

We report results of our method on two popular datasets for semantic segmentation, namely the PASCAL VOC 2010 [Everingham et al., 2010] and MSRC-21 dataset [Shotton et al., 2009]. We use the standard evaluation set up. We first describe the implementation details and discuss the computational times (with a CPU 2.8GHz i7 with 8 cores). Then, we analyze the impact of the parameters and the heuristics we use, and we report results on the two datasets. In the next section, we slightly modify the experimental setup and we show active MAP for human-in-the-loop semantic segmentation.

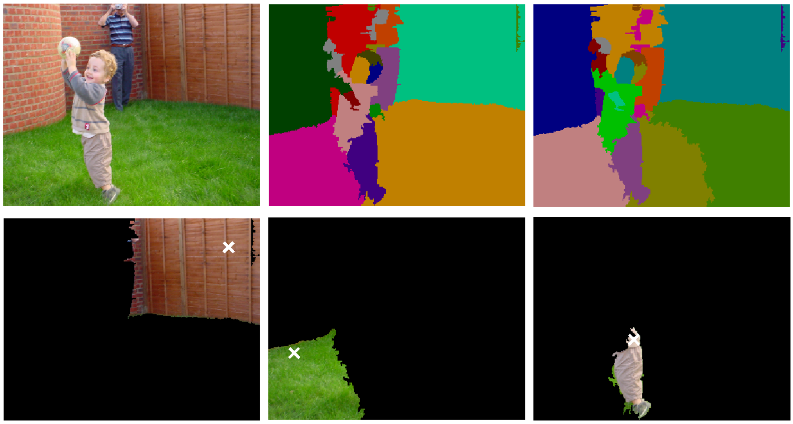


Figure 5.4: Area of Influence and Expected Reward. *The top row shows the original image and two samples of PM with* 0% *of observed unary potentials. The bottom row shows the area of influence of the first selected superpixels that ranked higher with ELC.*

5.6.1 Implementation and Computational Time

We use a typical CRF with Potts pairwise potential modulated by the difference in color [Fulkerson et al., 2009]. We use Active MAP to select which unary potentials to compute, since they have the higher computational load. The smoothness potentials are always computed, and thus, δ is initialized to include the smoothness potentials. Below we describe each of the pipeline components for semantic image segmentation.

Unsupervised Segmentation

We first over-segment the images using SLIC superpixels [Achanta et al., 2012], which allows us to work at superpixel level. The VOC10 images are over-segmented with about 800 superpixels, and for MSRC-21 we use about 300. SLIC takes on average 0.2 seconds per image.

Unary Potentials

In order to show that state-of-the-art methods can benefit from Active MAP, we use the publicly available features and classifiers in [Lucchi et al., 2011]. It extracts features taking into account the context of the image at different scales. Overlapping patches are described with SIFT and RGB histograms, and are encoded using Bag-of-Words (BoW) at 6 different contextual scales around the superpixels. The classifiers for each unary potential are SVMs with intersection kernel. In [Lucchi et al., 2011] they showed that with these features, they achieve comparable performance with or without using a CRF.

The computational cost for the different parts in VOC10 are 0.11s to compute dense SIFT, 0.05s to compute dense RGB histogram over the patches, and 0.6s to build all the BoW of an image with a fast nearest neighbor extraction. For MSRC-21 these costs are 0.03s to compute dense SIFT, 0.01s to compute RGB histograms, and 0.06s to build the BoWs. The cost of computing these features cannot be saved by the Active MAP inference, because we use a global classifier that uses features over the entire image.

In the case of VOC10, computing the classification score with an SVM for a superpixel takes 0.02 seconds, and, hence, for an image with 800 superpixels this takes 16 seconds. In MSRC-21, computing the classification score for each superpixel takes 0.01 seconds, and for 300 superpixels 3 seconds. Most of the classifier costs - *i.e.* the main bottleneck of the pipeline - can be saved by Active MAP inference.

Smoothness Potentials

The smoothness potentials are a Potts model modulated by the difference of the mean of the RGB color of the connected superpixels, *i.e.* $||c_i - c_j||$, where i, j indices two neighboring superpixels, and c is the mean RGB color vector of a superpixel. It takes 0.1 seconds to compute for 800 superpixels, and 0.03 seconds for 300 superpixels. We denote the weight that multiplies the smoothness term in the energy function as λ, and it is one of the parameters that we learn in the following section.

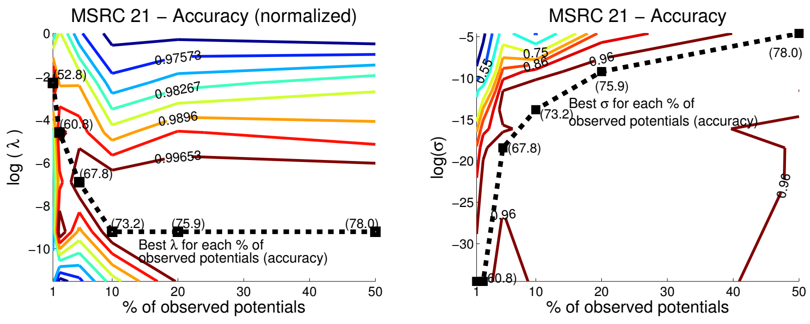


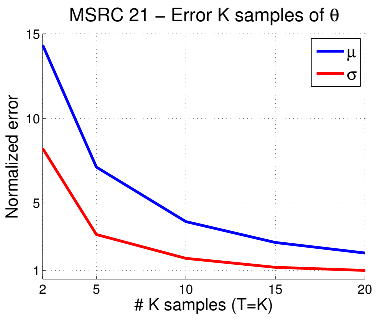
Figure 5.5: Learning the Parameters on MSRC-21. *(a) Impact of λ and (b) σ when varying the percentage of instantiated potentials. The accuracy is normalized by the maximum accuracy for each amount of observed potentials.*

Inference

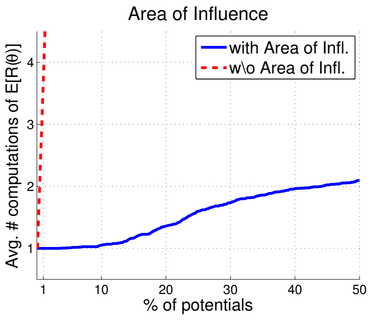
We use α-expansion graph cuts [Boykov & Kolmogorov, 2004] to compute the MAP labeling in a complete energy function. For the PM we use $K = 5$ samples, which takes 0.03 seconds for VOC10 and 0.02 seconds for MSRC-21. For the expected reward we use $T = 5$, and it takes 0.15 and 0.12 seconds in total, respectively. The final labeling $\mathbf{x}^\star$ is computed with $T = 1$, *i.e.* a single time MAP inference.

5.6.2 Learning the Parameters

The parameters that we learn are the weight of the smoothness potentials (λ), and the model of the missing unary potentials ($\boldsymbol{\mu}$, $\boldsymbol{\sigma}$). We learn them by cross-validation in the validation set, depending on the amount of instantiated potentials and the specific reward we use. In the following, we show the results in MSRC-21 when using the ELC reward. For VOC10 and the other rewards, we follow the same procedure.



(a)



(b)

***Figure 5.6**:* Impact of the Heuristics on MSRC-21. *(a) Average squared deviation (error) from the mean and variance of the set of K samples used for a PM in the calculation of the reward for one unknown unary potential, when $T = K$. The error is normalized to the error of not using the heuristic. (b) Average amount of times the scores are calculated when varying the percentage of observed unary potentials.*

Weight of the Smoothness Potential (λ)

In Figure 5.5a we report the impact of λ on the accuracy. We can see that depending on the percentage of instantiated potentials, the optimal value for λ may vary (indicated with the black line). Note that when few potentials are instantiated, the value of λ increases. This is because higher λ encourages label propagation, which is more important when we have less observations. When all potentials are observed, setting λ to 0 or very little gives the best performance, which is in accordance to Lucchi et al. [2011]. We use the best λ for each fraction of instantiated potentials.

Model of the Missing Parameters ($\boldsymbol{\mu}$, $\boldsymbol{\sigma}$)

We may use $\boldsymbol{\mu}$ to enforce a prior distribution over the classes. Yet, since the datasets we evaluate have only about 20 object classes, using this prior distribution can artificially boost the performance. Thus, we set all entries of $\boldsymbol{\mu}$ to the same constant value, which only adds an offset to the energy function that has no effect on the MAP labeling.

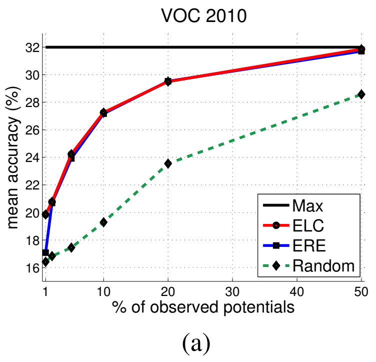
In Figure 5.5b we show the impact of $\boldsymbol{\sigma}$ on the accuracy, when varying the percentage of observed unary potentials. $\boldsymbol{\sigma}$ is the level of injected noise in $P(\mathbf{X}^\star = \mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{\delta}})$, which is necessary to effectively evaluate the expected rewards. Note that when there are more potentials instantiated, the optimal $\boldsymbol{\sigma}$ increases. This might be used to calibrate the amount of injected noise in the energy when less potentials can be perturbed.

5.6.3 Results

Efficient Computation of the Expected Rewards

We analyze the impact of the heuristics we introduced in Section 5.5.2. In Figure 5.6a we show the error of the mean and variance of the K samples of θ^i when reusing the samples generated for the expected value. Ideally, the samples of θ^i follow a Gaussian distribution, but due to the heuristic we use to generate them, the samples could have deviated from the original Gaussian distribution. We evaluate the average squared deviation (error) from the mean and variance of the set of K samples used in the calculation of the reward, for one unknown unary potential. The average is over all unknown potentials of all images, when there are no instantiated potentials. We set $T = K$ to proportionally increase the amount of samples. The error is normalized to the error of not using the heuristic. Note that as expected, the normalized error tends to 1 (same error as not using the heuristic) when we increase the number of samples. In the experiments, we use $T = K = 5$ samples because it is a good tradeoff between computational cost and accuracy.

In Figure 5.6b we analyze the impact of using the area of influence (Algorithm 7 compared to Algorithm 8). Recall that we discard the potentials that are in the area of influence of an instantiated potential, and we recompute the scores when all potentials have been discarded. We report the average number of times the scores are computed when varying the number of observed unary potentials. We can see that with 50% of the nodes instantiated, the scores are only computed 2 times (in average for all images). Note that this is a dramatic reduction of the computational cost, since without area of influence, the number of times it needs to compute the scores increases linearly to the number of observed unary potentials. Additionally, we observed that both methods obtain the same accuracy (we could only compare up to 10% of instantiated potentials due to the high computational cost of not using the area of influence).



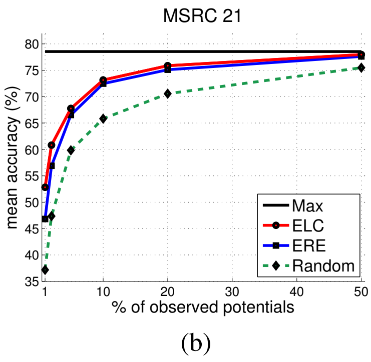


Figure 5.7: Results on (a) VOC10 and (b) MSRC-21. *Accuracy when varying the percentage of instantiated potentials.*

Method	Global	Average	Features	Inference	Total
All CRF	78	**78**	3s	0.02s	3.02s
All max	78	**78**	3s	–	3s
ELC 20%	76	**76**	0.6s	0.34s	0.94s
ERE 20%	76	75	0.6s	0.34s	0.94s
Random 20%	72	70	0.6s	0.1s	0.7s
ELC 5%	70	**68**	0.15s	0.34s	0.49s
ERE 5%	69	67	0.15s	0.34s	0.49s
Random 5%	65	60	0.15s	0.1s	0.25s

Table 5.1: Results on MSCR-21. *The average score provides the per-class average. The time measurements are for one image.*

Active MAP for semantic segmentation

We report results on MSRC-21 and VOC10, of the active MAP inference, with the ERE and ELC, and randomly selecting the unary potentials to compute the classifiers (referred as *Random*). We also report the results of using all the unary potentials and taking the maximum value of each, referred as *Max*, and when having the complete CRF, referred as *All CRF* in the tables.

In Figure 5.7 we show the evolution of the performance when increasing the amount of instantiated potentials on MSRC-21 and VOC10 (on the validation set), and in Table 5.1 we report more detailed results on the MSRC-21 dataset,

Method	Average	Features	Inference	Total
All CRF	**32.9**	16s	0.03	16.03s
All max	32.0	16s	–	16s
ELC 20%	**29.5**	3.2s	0.37s	3.57s
ERE 20%	29.5	3.2s	0.37s	3.57s
Random 20%	23.5	3.2s	0.12s	3.32s
ELC 5%	**24.2**	0.8s	0.12s	0.92s
ERE 5%	23.9	0.8s	0.12s	0.92s
Random 5%	17.4	0.8s	0.03s	0.83s

***Table 5.2**:* Results on VOC10 validation set. *The average score provides the per-class average. The time measurements are for one image.*

Method	Average	Features	Inference	Total
All	33.5	16s	–	16s
ELC 20%	30.4	3.2s	0.37s	3.57s
ELC 5%	24.8	0.8s	0.12s	1.17s
ELC 1%	–	–	–	–

***Table 5.3**:* Results on VOC10 testing set. *The average score provides the per-class average. The time measurements are for one image.*

and in Tables 5.2 and 5.3 on VOC10, for validation and test set respectively. We also report the times for computing the features and classifiers related to the potentials, and the inference time which includes the overhead of computing the active MAP inference. We can see that on VOC10, the Active MAP with ELC reward, yields a speed-up of around 20x when only instantiating 5% of the unary potentials, achieving very competitive results. When instantiating only 20% of the unary potentials, there is a speed-up of 5x, and the performance only decreases about 3% with respect to computing all the unary potentials. Note that the overhead of extra computation of Active MAP is very small for all cases. The ELC achieves slightly better accuracy than ERE reward, specially with fewer observed unary potentials, and both methods outperform the Random strategy.

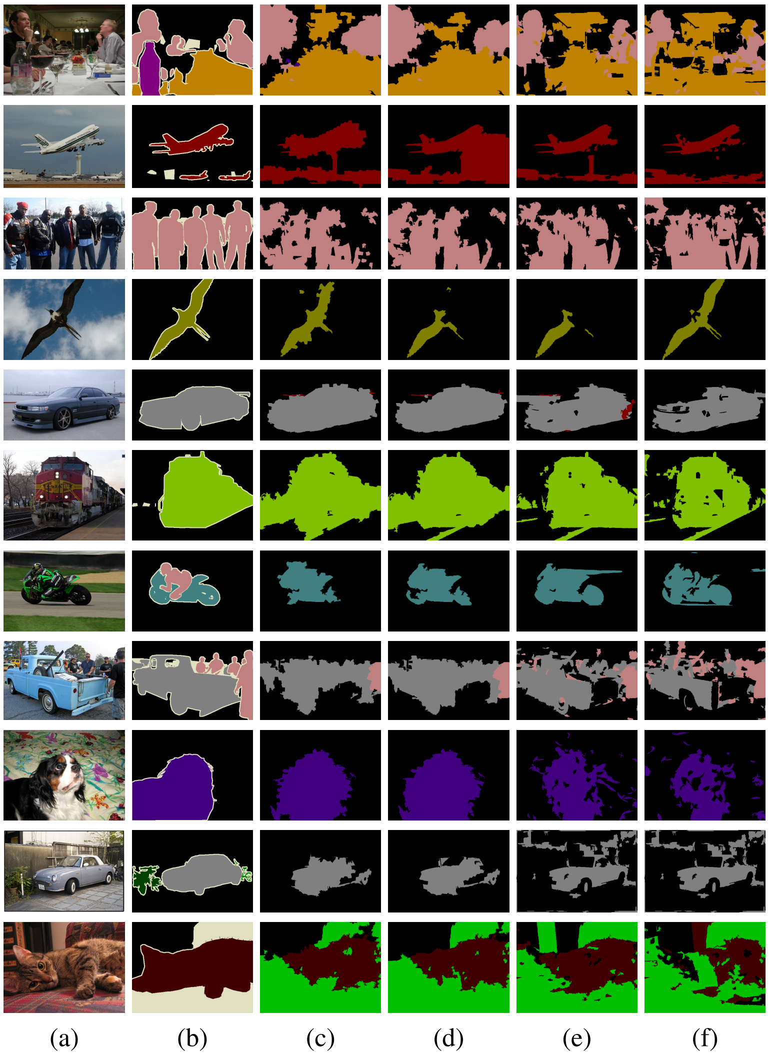


Figure 5.8: Examples of segmented images on VOC 2010. *(a) Original image, (b) groundtruth, (c) results with all unary potentials observed taking the maximum value, (d) all unary potentials observed using a pair-wise CRF model, (e) 20% of observed unary potentials, and (f) 5% observed unary potentials.*

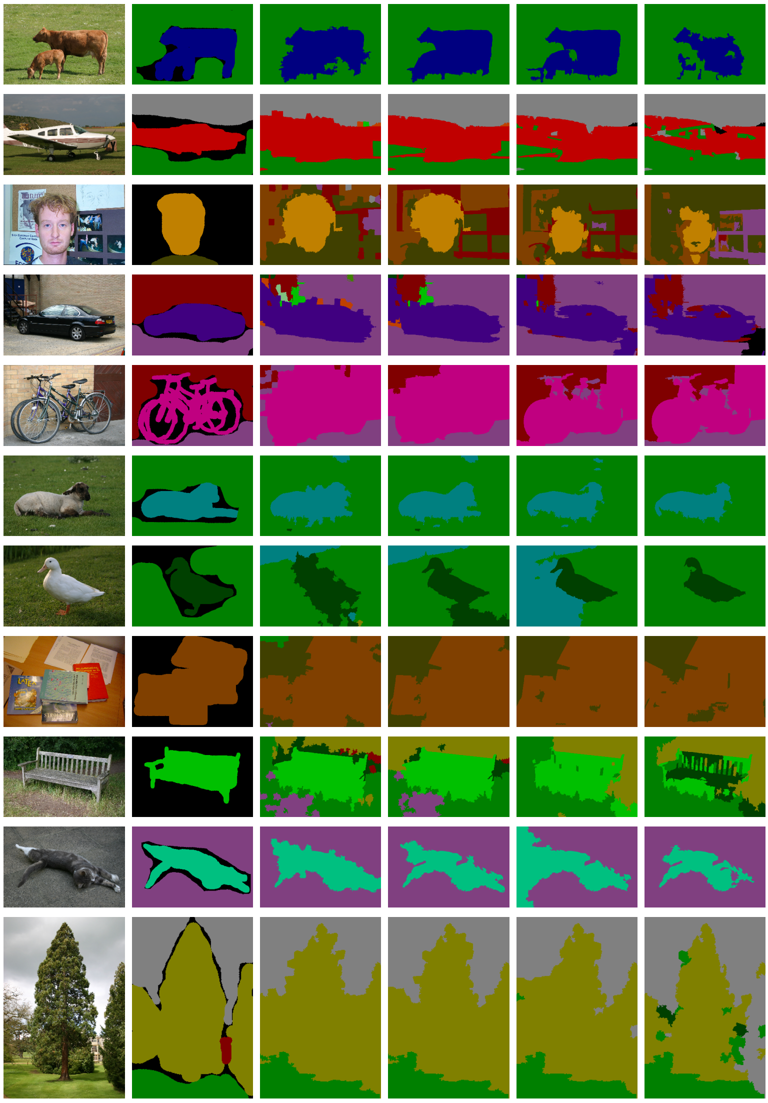


Figure 5.9: Examples of segmented images on MSRC-21 (see Figure 5.8).

Method	Global	Average	Features	Inference	Total
All	98	**97**	–	–	300 clicks
ELC 20%	94	**92**	–	0.34s	60 clicks
ELC 5%	86	84	–	0.34s	15 clicks
ELC 1%	67	67	–	0.34s	3 clicks

***Table 5.4**:* Results on MSCR-21 with a human in the loop. *The average score provides the per-class average. The time measurements are for one image.*

Images of Results

In Figure 5.8 and 5.9 we show some more examples of segmentation results achieved with Active MAP inference for different time budgets, on VOC 2010 dataset and MSRC-21, respectively. The first column corresponds to the original image and the second one to its grountruth. The third column is the obtained result of taking the maximum value of each unary potential independently. The fourth column are results with all potentials observed and a pair-wise CRF model. The fifth column are results obtained with a speed up of 5 (20% of the unary potentials computed), and the sixth column with a speed up of 20 (5% of the unary potentials).

5.7 Active MAP for human-in-the-loop segmentation

We evaluate the case of having the true labeling for some superpixels. This could be the case of having some unary potentials that may be prohibitive to compute, or also, when Active MAP interacts with a human that is asked the ground-truth for some superpixels. We slightly modify the set up used in previous experiments, by setting the instantiated unary potentials to add high penalties for the labels different from the ground-truth, or 0 otherwise.

We include the case that all the unary potentials are known with the true label (referred as *Max*), which gives an upper-bound of the performance limited by the errors introduced by the superpixels. We can see that with 2% of the superpixels, which is about 6 superpixels in the image, we obtain the same performance as the state-of-the-art method [Boix et al., 2012]. We show some examples of semantic segmentation results with a human-in-the-loop in Figure 5.11.

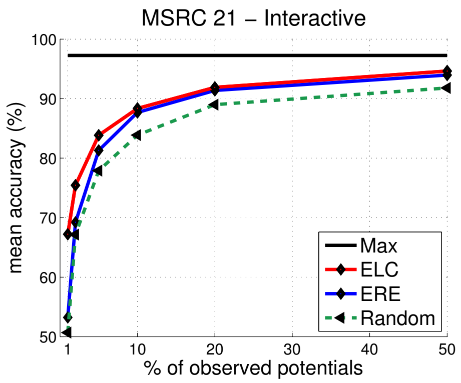


***Figure 5.10**:* Results on MSRC-21 with a human-in-the-loop. *Average accuracy*

5.8 Conclusion

We presented a method for active MAP inference on a CRF with unknown parameters. We showed its relation to the Perturb-and-MAP random field. The method incrementally adds the most promising parameters to the energy function using ranking criteria borrowed from active learning. Experiments on various datasets show that active MAP inference leads to significant computational savings, that clearly compensate for the overhead of computing the complete set of parameters of the energy function. The proposed method is useful when the computation of the energy function is more demanding than the MAP inference, as is often the case in semantic image segmentation.

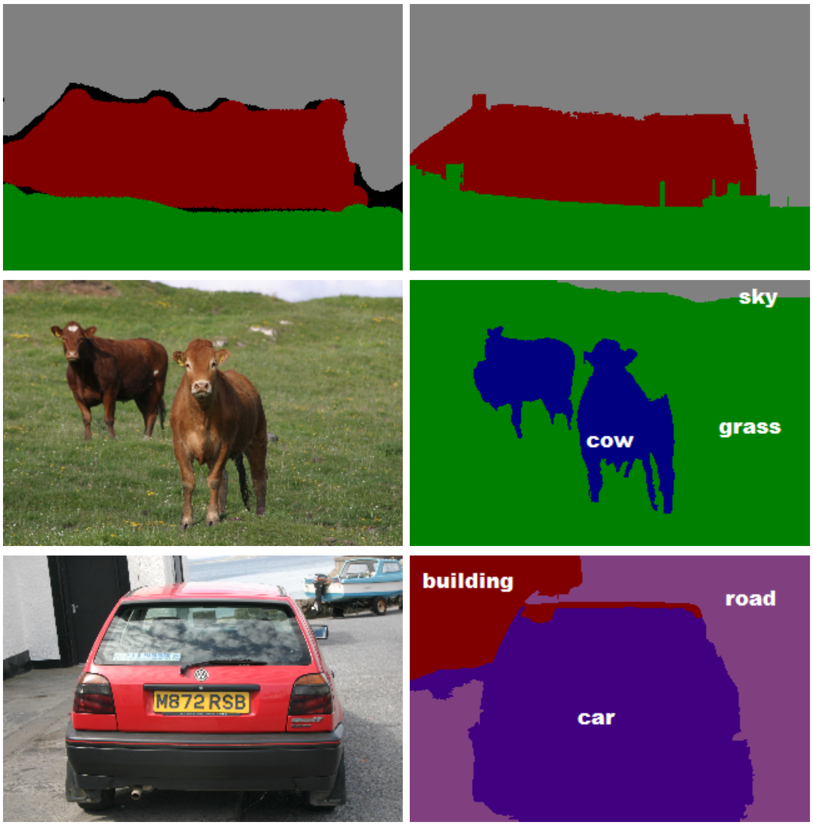


***Figure 5.11**:* Example of resulting images on MSRC-21 with a human-in-the-loop.

6

Conclusions

In this work we have proposed an efficient semantic image segmentation framework that delivers an output on a time budget that can be decided *on-the-fly*. This framework is suited for robotic applications, and autonomous navigating systems, which have limited computational resources and may need an output in a limited amount of time. We divided the pipeline of semantic segmentation in three steps, namely over-segmentation, feature extraction and classification, and labeling inference. We made a contribution in each of them putting special emphasis on efficiency.

We have proposed a superpixel algorithm for an efficient over-segmentation of the image, which can be stopped at any time, and is able to deliver an over-segmentation of the image in real time, obtaining state-of-the-art results. It is based on filters that are learned using Spectral Clustering techniques, and then the final optimization is done with a very efficient hill-climbing algorithm that starts from a grid partition of the image, and then, refines the partitioning moving the boundaries of the superpixels. We also introduced a binary quantization for the filters and descriptors to be able to speed-up the distance computations that are involved in the optimization, achieving major speed-ups without a significant decrease in the accuracy performance on standard superpixel benchmarks.

For the feature extraction we proposed an efficient feature encoding that is based on the same quantization used in the superpixels to binarize the feature descriptors an speed up the computations. It is based on a k-sparse quantization, which a quantization to the set of sparse vectors that have k ones and the rest elements of the vector to 0. Sparse quantization is computed with a simple sorting of the input feature vector, and a selection of the highest values of the vector. The highest entries of the feature vector are set to 1, and the rest to 0. We proposed a nested sparse quantization to binarize the input descriptors and compute the distances involved in the feature encoding with binary computations, which are very efficient to compute. We tested our feature encoding on image classification benchmarks, as well as in image segmentation, and showed that the accuracy performance is competitive to state-of-the-art methods while being very efficient to compute.

For the labeling inference, we introduced Active MAP inference for CRFs. Active MAP inference is able to infer where to put more computational effort in computing features and classifiers in regions of the image, and then propagate the information to the rest of the image with features that are cheap to compute, such as color information. With unobserved regions of the image is able to infer a labeling for all the image. It also delivers an uncertainty measure that can be used in robotic applications to determine the state of the robot, or to integrate the semantic segmentation in a probabilistic model that integrates multiple sensors of the robot.

Overall, we proposed a semantic segmentation pipeline that actively selects where to compute features and classifiers, and that can deliver a labeling of all image given a time budget.

7

Future Directions

The work presented in this thesis can be further extended to estimate the semantic segmentation in videos, exploiting the time consistency. This work could also be applied in other scenarios than efficiency and computing the output on a time budget. For instance, we could use it in applications where partial observability is needed or could help in the task. In the following we will further explain how this work could be applied in videos. Also how it could be used for eye movement prediction, where semantic uncertainty and partial observability might be beneficial to get a good approximation of the scanpath eye prediction.

Semantic Video Segmentation on a Time Budget

Extending our semantic segmentation framework to videos could be done using a video over-segmentation as a pre-processing step. For instance we could use Video SEEDS, which is the extension of SEEDS superpixels to video that we proposed in [Van den Bergh et al., 2013]. It is based in the same algorithm as SEEDS, including also the time consistency information. It generates tubes in time of superpixels, and runs on-line and in real-time. The tubes could be use to propagate the information, and compute features and classifiers only in some regions of the tube. In this scenario, we could build a graph, in which each node corresponds to a superpixel tube. Also, with the Active MAP inference, we could select in which tubes of superpixels we compute the features and classifiers, and propagate the labeling information to the adjacent tubes.

We presented a preliminary work of the video idea in [de Nijs et al., 2012], where we used the uncertainty computed with the entropy of the multiple samples to obtain an online semantic segmentation for an autonomous robot. The selection was done with the entropy, and it was based on superpixels for still images, and not tubes of superpixels. We expect that using a more sophisticated selection of the regions in which to compute features and classifiers, as the ones we proposed in the Active MAP inference, and the tubes of superpixels that leverage the time consistency, will improve the performance and the efficiency of the method in [de Nijs et al., 2012].

Another extension of the Active MAP in semantic segmentation is to explore different reward functions for selecting which superpixels to compute (or tubes of superpixels in the scenario with videos). A reasonable alternative is to use Reinforcement Learning to learn from examples which are the best superpixels to select. The objective function could be maximazing the accuracy of the semantic segmentation labeling of the whole image at each time step.

Partial Observability with Active MAP for Saliency and Scanpath Prediction

Visual attention models aim to predict human eye fixations in natural scenes, [Itti et al., 1998]. Many state-of-the-art models incorporate semantic-level features, which often are a stronger cue than the low-level features to predict eye fixations. The semantic-level features are implemented by evaluating object classifiers everywhere in the image. As a result, saliency models assume the full observability of the image, which does not correspond to the function of the eye gaze shifts that gradually capture the semantic-level information. We could drop the assumption of full observability of the semantic labels in all the image using our active MAP inference for semantic segmentation to model the partial observability of the semantic-level, and compute semantic features for saliency prediction based on the uncertainty and rarity of the semantics during the gradual exploration of the image. Some of the semantic-level features that could be computed with the active semantic segmentation for scanpath prediction could be:

- *Label Probability:* It has been shown that certain object categories attract attention more strongly and rapidly than others [Cerf et al., 2009]. The label probability could be computed as the normalized distribution of class labels from the samples of the active semantic segmentation.
- *Semantic Uncertainty:* since with the active MAP inference we have a probability for the labeling, we could compute the Shannon entropy of

the label probability, which can be used as a semantic-level feature for scanpath prediction.

Also, the use of SEEDS superpixels, instead of pixels, as the basic unit for saliency computation may allow enhancement on signal-to-noise-ratio at a reduced computational cost, and features defined on superpixels may also provide richer representations in the descriptions of a coherent group of pixels.

A

Appendix SEEDS Superpixels

Evaluating Pixel-level and Block-level Movements

In this appendix we give more detail of the observation used to speed up the evaluation of the pixel-level and block-level movements in SEEDS.

Recall that $\mathcal{A}_k^l$ is the set of pixels that are candidates to be moved from the superpixel $\mathcal{A}_k$ to $\mathcal{A}_n$.

Observation 1 *Let the sizes of $\mathcal{A}_k$ and $\mathcal{A}_n$ be similar, and $\mathcal{A}_k^l$ much smaller,* i.e. $|\mathcal{A}_k| \approx |\mathcal{A}_n| \gg |\mathcal{A}_k^l|$. *If the histogram of $\mathcal{A}_k^l$ is concentrated in a single bin, then*

$$\mathbf{int}(c_{\mathcal{A}_n}, c_{\mathcal{A}_k^l}) \geq \mathbf{int}(c_{\mathcal{A}_k \setminus \mathcal{A}_k^l}, c_{\mathcal{A}_k^l}) \iff H(s) \geq H(s_t). \tag{A.1}$$

Recall that the color term of the energy function is:

$$H(s) = \sum_k \sum_{\{\mathcal{H}_j\}} \left(\frac{1}{|\mathcal{A}_k|} \sum_{i \in \mathcal{A}_k} \delta(I(i) \in \mathcal{H}_j) \right)^2, \tag{A.2}$$

in which we simply merged Equation (3.4) and (3.5). We write $H(s) \geq H(s_t)$ taking into account that s and s_t only differ in $\mathcal{A}_k^l$, and the assumption of the Observation on the size of the superpixels, *i.e.* $|\mathcal{A}_k| \approx |\mathcal{A}_n| \gg |\mathcal{A}_k^l|$. Thus, the expression does not take into account the color at superpixels different from

k and n, and we can get rid of the normalization of the histograms due to the assumption. Then, the evaluation becomes,

$$
\begin{aligned}
& H(s) \geq H(s_t) \iff \\
& \sum_{\{\mathcal{H}_j\}} \left(\sum_{i \in \mathcal{A}_n} \delta(I(i) \in \mathcal{H}_j) + \sum_{i \in \mathcal{A}_k^l} \delta(I(i) \in \mathcal{H}_j) \right)^2 + \\
& + \sum_{\{\mathcal{H}_j\}} \left(\sum_{i \in \mathcal{A}_k \backslash \mathcal{A}_k^l} \delta(I(i) \in \mathcal{H}_j) \right)^2 \geq \\
& \geq \sum_{\{\mathcal{H}_j\}} \left(\sum_{i \in \mathcal{A}_k \backslash \mathcal{A}_k^l} \delta(I(i) \in \mathcal{H}_j) + \sum_{i \in \mathcal{A}_k^l} \delta(I(i) \in \mathcal{H}_j) \right)^2 + \\
& + \sum_{\{\mathcal{H}_j\}} \left(\sum_{i \in \mathcal{A}_n} \delta(I(i) \in \mathcal{H}_j) \right)^2 .
\end{aligned} \tag{A.3}
$$

The second assumption of the Observation is that $\mathcal{A}_k^l$ is concentrated in a single bin. Let $\mathcal{H}^*$ be the color in which A_k^l is concentrated. Then, the evaluation in Equation (A.3) becomes

$$\left(\sum_{i\in\mathcal{A}_n}\delta(I(i)\in\mathcal{H}^\star)+\sum_{i\in\mathcal{A}_k^l}\delta(I(i)\in\mathcal{H}^\star)\right)^2+$$
$$+\sum_{\{\mathcal{H}_j\}\backslash\mathcal{H}^\star}\left(\sum_{i\in\mathcal{A}_n}\delta(I(i)\in\mathcal{H}_j)\right)^2+$$
$$+\sum_{\{\mathcal{H}_j\}}\left(\sum_{i\in\mathcal{A}_k\backslash\mathcal{A}_k^l}\delta(I(i)\in\mathcal{H}_j)\right)^2\geq$$
$$\geq\left(\sum_{i\in\mathcal{A}_k\backslash\mathcal{A}_k^l}\delta(I(i)\in\mathcal{H}^\star)+\sum_{i\in\mathcal{A}_k^l}\delta(I(i)\in\mathcal{H}^\star)\right)^2+$$
$$+\sum_{\{\mathcal{H}_j\}\backslash\mathcal{H}^\star}\left(\sum_{i\in\mathcal{A}_k\backslash\mathcal{A}_k^l}\delta(I(i)\in\mathcal{H}_j)\right)^2+$$
$$+\sum_{\{\mathcal{H}_j\}}\left(\sum_{i\in\mathcal{A}_n}\delta(I(i)\in\mathcal{H}_j)\right)^2. \tag{A.4}$$

Then, note the following simple equality:

$$\left(\sum_{i\in\mathcal{A}_n}\delta(I(i)\in\mathcal{H}^\star)+\sum_{i\in\mathcal{A}_k^l}\delta(I(i)\in\mathcal{H}^\star)\right)^2= \tag{A.5}$$
$$\left(\sum_{i\in\mathcal{A}_n}\delta(I(i)\in\mathcal{H}^\star)\right)^2+\left(\sum_{i\in\mathcal{A}_k^l}\delta(I(i)\in\mathcal{H}^\star)\right)^2+$$
$$+2\left(\sum_{i\in\mathcal{A}_n}\delta(I(i)\in\mathcal{H}^\star)\right)\left(\sum_{i\in\mathcal{A}_k^l}\delta(I(i)\in\mathcal{H}^\star)\right), \tag{A.6}$$

and we introduce it to the evaluation in Equation (A.4). Reordering the terms, and canceling the same terms in both sides of the inequality, Equation (A.4) becomes:

$$H(s) \geq H(s_t) \Longleftrightarrow \tag{A.7}$$

$$\sum_{i \in \mathcal{A}_n} \delta(I(i) \in \mathcal{H}^\star) \geq \sum_{i \in \mathcal{A}_k \backslash \mathcal{A}_k^l} \delta(I(i) \in \mathcal{H}^\star). \tag{A.8}$$

Now, we develop the intersection distances in the Observation to arrive to Equation (A.8). We use the following expression:

$$\mathbf{int}(c_{\mathcal{A}_n}, c_{\mathcal{A}_k^l}) = \tag{A.9}$$

$$\sum_{\{\mathcal{H}_j\}} \min \left(\frac{1}{|\mathcal{A}_n|} \sum_{i \in \mathcal{A}_n} \delta(I(i) \in \mathcal{H}_j), \frac{1}{|\mathcal{A}_k^l|} \sum_{i \in \mathcal{A}_k^l} \delta(I(i) \in \mathcal{H}_j) \right),$$

and since we assumed that the histogram of $\mathcal{A}_k^l$ is concentrated in one bin, the expression becomes

$$\mathbf{int}(c_{\mathcal{A}_n}, c_{\mathcal{A}_k^l}) = \frac{1}{|\mathcal{A}_n|} \sum_{i \in \mathcal{A}_n} \delta(I(i) \in \mathcal{H}^\star). \tag{A.10}$$

Finally, we use this expression and the assumption of $|\mathcal{A}_k| \approx |\mathcal{A}_n|$,and we obtain Equation (A.8):

$$\mathbf{int}(c_{\mathcal{A}_n}, c_{\mathcal{A}_k^l}) \geq \mathbf{int}(c_{\mathcal{A}_k \backslash \mathcal{A}_k^l}, c_{\mathcal{A}_k^l}) \Longleftrightarrow \tag{A.11}$$

$$\sum_{i \in \mathcal{A}_n} \delta(I(i) \in \mathcal{H}^\star) \geq \sum_{i \in \mathcal{A}_k \backslash \mathcal{A}_k^l} \delta(I(i) \in \mathcal{H}^\star) \Longleftrightarrow \tag{A.12}$$

$$H(s) \geq H(s_t) \tag{A.13}$$

Bibliography

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Susstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. (document), 3.2.2, 3.15, 3.5.4, 5.6.1

Alexe, B., Deselaers, T., & Ferrari, V. (2012a). Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *34*(11), 2189–2202. 2.2, 3.1

Alexe, B., Heess, N., Teh, Y., & Ferrari, V. (2012b). Searching for objects driven by context. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 2.2

Arbelaez, P., Hariharan, B., Gu, C., Gupta, S., Bourdev, L., & Malik, J. (2012). Semantic segmentation using regions and parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.1

Bach, F. R., & Jordan, M. I. (2006). Learning spectral clustering, with application to speech recognition. *JMLR*. 3.4.1, 3.4.2

Benoit, L., Mairal, J., Bach, F., & Ponce, J. (2011). Sparse image representation with epitomes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.1

Boiman, O., Shechtman, E., & Irani, M. (2008). In defense of nearest-neighbor based image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.1

Boix, X., Gonfaus, J. M., van de Weijer, J., Bagdanov, A. D., Serrat, J., & Gonzàlez, J. (2012). Harmony potentials - fusing global and local scale for semantic image segmentation. *International Journal of Computer Vision (IJCV)*. 2.1, 3.1, 5.1, 5.1, 5.2, 5.7

Boix, X., Gygli, M., Roig, G., & Van Gool, L. (2013). Sparse quantization for patch description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.4.2

Boureau, Y., Bach, F., LeCun, Y., & Ponce, J. (2010a). Learning mid-level features for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.1

Boureau, Y., Ponce, J., & LeCun, Y. (2010b). A theoretical analysis of feature pooling in vision algorithms. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 4.1

Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 5.6.1

Brubaker, S. C., Wu, J., Sun, J., Mullin, M. D., & Rehg, J. M. (2008). On the design of cascades of boosted ensembles for face detection. *Int. J. Comput. Vision*. 2.2

Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010). Brief: Binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 4.4.1, 4.5.2

Carreira, J., Caseiro, R., Batista, J., & Sminchisescu, C. (2012a). Semantic segmentation with second-order pooling. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2.1

Carreira, J., Li, F., & Sminchisescu, C. (2012b). Object recognition by sequential figure-ground ranking. *International Journal of Computer Vision (IJCV)*. 2.1

Cerf, M., Frady, E., & Koch, C. (2009). Faces and text attract gaze independent of the task: Experimental data and computer model. *Journal of Vision (JoV)*. 7

Chang, K. Y., Liu, T. L., Chen, H. T., & Lai, S. H. (2011). Fusing generic objectness and visual saliency for salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2.2

Chatfield, K., Lempitsky, V., Vedaldi, A., & Zisserman, A. (2011). The devil is in the details: An evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference (BMVC).* (document), 4.1, 4.3.1, 4.5.3, 4.1

Chen, X., & Yuille, A. L. (2005). A time-efficient cascade for real-time object detection: with applications for the visually impaired. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2.2

Cheng, M.-M., Zhang, Z., Lin, W.-Y., & Torr, P. H. S. (2014). BING: Binarized normed gradients for objectness estimation at 300fps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2.2

Coates, A., & Ng, A. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the International Conference on Machine Learning (ICML).* 4.1, 4.5.2

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *24*(5), 603–619. 3.2.2

Commission, E. (2010–2013a). FP7-ICT-248314, IURO: Interactive Urban Robot. http://www.iuro-project.eu/. 1

Commission, E. (2010–2013b). FP7-ICT-248873, RADHAR: Robotic ADaptation to Humans Adapting to Robots. http://www.radhar.eu/. 1

Cour, T., Benezit, F., & Shi, J. (2005). Spectral segmentation with multiscale graph decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 3.2.1

Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. (2004). Visual categorization with bags of keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV).* 4.1, 4.5.2

Csurka, G., & Perronnin, F. (2010). An efficient approach to semantic segmentation. *International Journal of Computer Vision (IJCV).* 2.1, 5.1

de Nijs, R., Ramos, J., Roig, G., Boix, X., Van Gool, L., & Kuehnlenz, K. (2012). On-line semantic perception using uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 7

Demeester, E., Vander Poorten, E., Huentemann, A., De Schutter, J., Hofmann, M., Rooker, M., Kronreif, G., Lau, B., Kuderer, M., Burgard, W., Gelin, A., Vanopdenbosch, K., Van der Beeten, P., Vereecken, M., Ilsbroukx, S., Fossati, A., Roig, G., Boix, X., Ristin, M., Van Gool, L., Fraeyman, H., Broucke, L., Goessaert, H., & Josten, J. (2012). Robotic ADaptation to Humans Adapting to Robots: Overview of the FP7 project RADHAR. In *International Conference on Systems and Computer Science (ICSCS)*. 1

Deng, J., Berg, A. C., Li, K., & Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us? In *Proceedings of the European Conference on Computer Vision (ECCV)*. 4.5.3

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.5

DeVore, R. (1998). Nonlinear approximation. *Acta Numerica*. 4.2.2

Donath, W., & Hoffman, A. J. (1973). Lower bounds for the partitioning of graphs. *IBM J. Res. Develop.*. 3.4.1

Eriksson, A., Olsson, C., & Kahl, F. (2007). Normalized cuts revisited: a reformulation for segmentation with linear grouping constraints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 3.2.1

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 4.5

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The PASCAL Visual Object Classes Challenge 2010. 5.1, 5.6

Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 4.5

Felzenszwalb, P., & Huttenlocher, D. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*. (document), 3.1, 3.2.1, 3.15, 3.5.4

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Math. J.*. 3.4.1

Floros, G., Rematas, K., & Leibe, B. (2011). Multi-class image labeling with top-down segmentation and generalized robust p^n potentials. In *Proceedings of the British Machine Vision Conference (BMVC)*. 2.1

Fulkerson, B., Vedaldi, A., & Soatto, S. (2009). Class segmentation and object localization with superpixel neighborhoods. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2.1, 3.1, 5.1, 5.1, 5.2, 5.6.1

Gall, J., Yao, A., Razavi, N., Van Gool, L., & Lempitsky, V. (2011). Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 2.2

Galleguillos, C., Rabinovich, A., & Belongie, S. (2008). Categorization using co-occurrence, location and appearance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.1

Gao, T., & Koller, D. (2011). Active classification based on value of classifier. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 2.2

Gordo, A., & Perronnin, F. (2011). Asymmetric distances for binary embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.4.2

Gould, S., Rodgers, J., Cohen, D., Elidan, G., & Koller, D. (2008). Multi-class segmentation with relative location prior. *International Journal of Computer Vision (IJCV)*. 2.1, 5.1

Hariharan, B., Arbelaez, P., Girshick, R., & Malik, J. (2014). Simultaneous detection and segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2.1

Hervé Jégou, C. S., Matthijs Douze (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 4.4.2

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*. 4.4.2

Ion, A., Carreira, J., & Sminchisescu, C. (2011). Image segmentation by figure-ground composition into maximal cliques. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2.1

Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 7

Jiang, J., & Tu, Z. (2009). Efficient scale space auto-context for image segmentation and labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.1, 5.1

Karayev, S., Fritz, M., & Darrell, T. (2012). Timely object recognition. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 2.2

Karayev, S., Fritz, M., & Darrell, T. (2014). Anytime recognition of objects and scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.2

Kohli, P., Ladicky, L., & Torr, P. (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision (IJCV)*. 2.1, 5.1

Kohli, P., & Torr, P. H. S. (2006). Measuring uncertainty in graph cut solutions - efficiently computing min-marginal energies using dynamic graph cuts. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2.1

Kolmogorov, V., & Wainwright, M. J. (2005). On optimality properties of tree-reweighted message-passing. In *UAI*. 5.2

Kumar, M., Torr, P., & Zisserman, A. (2005). Obj cut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.1

Ladicky, L., Russell, C., Kohli, P., & Torr, P. H. S. (2009). Associative hierarchical crfs for object class image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2.1

Ladicky, L., Russell, C., Kohli, P., & Torr, P. H. S. (2010a). Graph cut based inference with co-occurence statistics. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2.1

Ladicky, L., Sturgess, P., Alahari, K., Russell, C., & Torr, P. H. S. (2010b). What, where and how many? combining object detectors and crfs. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2.1

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML).* 5.1

Lehmann, A., Gehler, P., & Van Gool, L. (2014). Branch&rank for efficient object detection. *International Journal of Computer Vision.* 2.2

Lehmann, A., Leibe, B., & Van Gool, L. (2011). Fast PRISM: Branch and bound hough transform for object class detection. *International Journal of Computer Vision.* 2.2

Leibe, B., Leonardis, A., & Schiele, B. (2007). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision.* 2.1, 2.2

Lempitsky, V., Kohli, P., Rother, C., & Sharp, T. (2009). Image segmentation with a bounding box prior. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 2.1

Levinshtein, A., Stere, A., Kutulakos, K., Fleet, D., Dickinson, S., & Siddiqi, K. (2009). Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *31*(12), 2290–2297. 3.1, 3.2.1, 3.2.2

Li, L.-J., Socher, R., & Fei-Fei, L. (2009). Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2.1

Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., & Yu, K. (2011). Large-scale image classification: fast feature extraction and svm training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 4.5.3

Liu, L., Wang, L., & Liu, X. (2011a). In defence of soft-assignment coding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* (document), 4.1, 4.3.1, 4.3.2, 4.5.3, 4.1

Liu, M.-Y., Tuzel, O., Ramalingam, S., & Chellappa, R. (2011b). Entropy rate superpixel segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (document), 3.2.1, 3.5.1, 3.15, 3.5.4

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI).* 4.5.2

Lucchi, A., Li, Y., Boix, X., Smith, K., & Fua, P. (2011). Are spatial and global constraints really necessary for segmentation? In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 4.6.1, 4.6.1, 5.1, 5.1, 5.2, 5.6.1, 5.6.2

Luo, H. (2005). Optimization design of cascaded classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2.2

Maire, M., Yu, S. X., & Perona, P. (2011). Object detection and segmentation from joint embedding of parts and pixels. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 2.1, 3.4.1, 5.1

Maji, S., & Malik, J. (2009). Object detection using a max-margin hough transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2.2

Malik, J., Belongie, S., Leung, T., & Shi, J. (2001). Contour and texture analysis for image segmentation. *International Journal of Computer Vision (IJCV).* 3.4.1, 3.4.2, 3.4.3

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 3.1, 3.5

Meyer, F., & Maragos, P. (1999). Multiscale morphological segmentations based on watershed, flooding, and eikonal PDE. In *Proceedings int. conf. on Scale-Space Theories in Computer Vision.* 3.2.2

Moore, A., Prince, S., & Warrell, J. (2010). Lattice cut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 3.2.1

Moore, A., Prince, S., Warrell, J., Mohammed, U., & Jones, G. (2008). Superpixel lattices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 3.2.1

Moosmann, F., Jurie, F., & Triggs, B. (2007). Fast discriminative visual codebooks using randomized clustering forests. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 4.1, 4.5.2

Munoz, D., Bagnell, J. A., & Hebert, M. (2010). Stacked hierarchical labeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2.1, 5.1

Nguyen, K., Boix, X., Roig, G., Gool, L. V., & Lygeros, J. (2011). Robocup ETH Zuerich team. http://www.robocup.ethz.ch/. 1

Nister, D., & Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.1

Nowak, E., Jurie, F., & Triggs, B. (2006). Sampling strategies for bag-of-features image categorization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 4.1

Oliva, A., & Torralba, A. (2007). The role of context in object recognition. *Trends in Cognitive Sciences, 11(12)*, (pp. 520–527). 2.1

Olshausen, B., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*. 4.3.3

Pantofaru, C., Schmid, C., & Hebert, M. (2008). Object recognition by integrating multiple image segmentations. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2.1, 5.1

Papandreou, G., & Yuille, A. (2011). Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 5.1, 5.3, 5.3

Pedersoli, M., Vedaldi, A., & Gonzàlez, J. (2011). A coarse-to-fine approach for fast deformable object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.2

Perronin, F., Sanchez, J., & Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 4.1

Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.1, 4.3.1

Plath, N., Toussaint, M., & Nakajima, S. (2009). Multi-class image segmentation using conditional random fields and global classification. In *Proceedings of the International Conference on Machine Learning (ICML)*. 2.1, 5.1

Rahimi, A., & Recht, B. (2004). A feature space view of spectral clustering. In *Workshop on Statistical Learning in Computer Vision, Proceedings of the European Conference on Computer Vision (ECCV)*. 3.1, 3.4.1, 3.4.3

Ren, X., & Malik, J. (2003). Learning a classication model for segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 3.2

RoboCup, T. C. (2014). Robocup standard platform league (NAO). http://www.informatik.uni-bremen.de/spl/bin/view/Website/WebHome. 1

Roig, G., Boix, R., X.and de Nijs, Ramos, S., Kuehnlenz, K., & Van Gool, L. (2013). Active map inference in crfs for efficient semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 4.6.1

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648. University of WisconsinMadison. 5.5.1

Shakhnarovich, G. (2005). *Learning Task-Specic Similarity*. Ph.D. thesis, Massachusetts Institute of Technology. 4.4.1, 4.5.2

Sharp, H. (1968). Cardinality of finite topologies. *J. Combinatorial Theory*, *5*(1), 82–86. 3.3

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 3.2.1, 3.4.1

Shotton, J., & Kohli, P. (2014). Semantic image segmentation. *Computer Vision: A Reference Guide*, (pp. 713–716). 2.1

Shotton, J., Winn, J., Rother, C., & Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision (IJCV)*. 2.1, 4.6, 5.1, 5.1, 5.6

Sivic, J., & Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 4.1, 4.5.2

Tarlow, D., Adams, R., & Zemel, R. (2012). Randomized optimum models for structured prediction. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 5.3

Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., & Van Gool, L. (2012). SEEDS: Superpixels extracted via energy-driven sampling. In *Proceedings of the European Conference on Computer Vision (ECCV)*. (document), 3.1, 3.2, 3.4.2, 3.5.1, 3.15, 3.5.4

Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., & Van Gool, L. (2014). SEEDS: Superpixels extracted via energy-driven sampling. *International Journal of Computer Vision (IJCV), in press*. 3.1, 4.6.1

Van den Bergh, M., Roig, G., Boix, X., Manen, S., & Van Gool, L. (2013). Online video SEEDS for temporal window objectness. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2.2, 3.1, 3.3.3, 7

Vedaldi, A., & Soatto, S. (2008). Quick shift and kernel methods for mode seeking. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 3.2.2

Veksler, O., & Boykov, Y. (2010). Superpixels and supervoxels in an energy optimization framework. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 3.2.1

Verbeek, J., & Triggs, B. (2007). Scene segmentation with conditional random fields learned from partially labeled images. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 2.1, 5.1

Vezhnevets, A., Buhmann, J. M., & Ferrari, V. (2012). Active learning for semantic segmentation with expected change. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5.5.1

Vincent, L., & Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 3.2.2

Viola, P. A., & Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, *57*(2), 137–154. 2.2

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*. 3.4.1, 3.4.1

Wainwright, M. J., Jaakkola, T. S., & Willsky, A. S. (2005). MAP estimation via agreement on trees: Message-passing and linear programming. *IEEE Transactions on Information Theory*. 5.2

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2010). Locality-constrained linear coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (document), 4.1, 4.5.2, 4.5.3, 4.1, 4.5.3, 4.2

Wang, S., Lu, H., Yang, F., & Yang, M.-H. (2011). Superpixel tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 3.1

Weiss, Y., Torralba, A., & Fergus, R. (2008). Spectral hashing. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 4.4.2

Wertheimer, M. (1938). *Laws of organization in perceptual forms*. Harcourt, Brace & Jovanovitch. 3.2

Wu, Z., & Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *15*(11), 1101–1113. 3.2.1

Xu, L., Li, W., & Schuurmans, D. (2009). Fast normalized cut with linear constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3.2.1

Yang, J., Yu, K., Gong, Y., & Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.1, 4.3.3

Yedidia, J., Freeman, W., & Weiss, Y. (2001). Understanding belief propagation and its generalizations. *International Journal of Computer Vision (IJCV)*. 2.1

Yu, K., Lin, Y., & Lafferty, J. (2011). Learning image representations from pixel level via hierarchical sparse coding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4.1, 4.4.2

Yu, S. X., & Shi, J. (2003). Multiclass spectral clustering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 3.4.1, 3.4.1, 3.4.2

Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 4.1

Zeng, G., Wang, P., Wang, J., Gan, R., & Zha, H. (2011). Structure-sensitive superpixels via geodesic distance. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 3.2.2

Zhang, Y., Hartley, R., Mashford, J., & Burn, S. (2011). Superpixels via pseudo-boolean optimization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 3.2.1

Zhang, Z., & Jordan, M. I. (2008). Multiway spectral clustering: A margin-based perspective. *Statistical Science*. 3.1, 3.4.1, 3.4.2, 3.4.3

Zhou, X., Yu, K., Zhang, T., & Huang, T. (2010). Image classification using super vector coding of local image descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*. (document), 4.1, 4.5.2, 4.5.3, 4.2

Zitnick, C., Jojic, N., & Kang, S. (2005). Consistent segmentation for optical flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 3.2.2

List of Publications

Peer-Reviewed Journals

1. M. Van den Bergh, X. Boix, G. Roig, L. Van Gool. SEEDS: Superpixels Extracted via Energy-Driven Sampling. *International Journal of Computer Vision (IJCV)*. 2015.

Peer-Reviewed Conference Papers

1. X. Boix*, G. Roig*, S. Diether, L. Van Gool. Self-adaptable Templates for Self-Adaptable Templates for Feature Coding. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.
2. G. Roig*, X. Boix*, R. de Nijs, S. Ramos, K. Kühnlenz, L. Van Gool. Active MAP Inference in CRFs for Efficient Semantic Segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. (oral)
3. M. Van den Bergh, G. Roig, X. Boix, S. Manen, L. Van Gool. Online Video SEEDS for Temporal Window Objectness. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.
4. X. Boix, M. Gygli, G. Roig, L. Van Gool. Sparse Quantization for Patch Description. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
5. X. Boix*, G. Roig*, L. Van Gool. Nested Sparse Quantization for Efficient Feature Coding. In *Proceedings European Conference on Computer Vision (ECCV)*, 2012.

* Indicates equal contribution.

6. M. Van den Bergh, X. Boix, G. Roig, B. Capitani, L. Van Gool. SEEDS: Superpixels Extracted via Energy-Driven Sampling. In *Proceedings European Conference on Computer Vision (ECCV)*, 2012.

7. R. de Nijs*, S. Ramos*, G. Roig, X. Boix, L. Van Gool, K. Kühnlenz. On-line Semantic Perception Using Uncertainty. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

8. E. Demeester, E. Vander Poorten, A. Hüntemann, J. De Schutter, B. Lau, M. Kuderer, W. Burgard, A. Fossati, G. Roig, X. Boix, M. Ristin, L. Van Gool, *et al.*. Robotic ADaptation to Humans Adapting to Robots: Overview of the FP7 project RADHAR. In *Proceedings International Conference on Systems and Computer Science (ICSCS)*, 2012.

9. G. Roig*, X. Boix*, H. Ben Shitrit and P. Fua. Conditional Random Fields for Multi-Camera Object Detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

10. G. Roig*, X. Boix*, F. De la Torre, J. Serrat and C. Vilella. Hierarchical CRF with Product Label Spaces for Parts-based Models. In *Proceedings IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, 2011. (oral)

11. G. Roig*, X. Boix* and F. De la Torre. Optimal Feature Selection for Subspace Image Matching. In *Proceedings IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2009.

Pre-Prints

1. G. Roig*, X. Boix*, L. Van Gool. Random Binary Mappings for Kernel Learning and Efficient SVM. *arXiv:1307.5161*, 2014.

Others

1. G. Roig*, X. Boix*, C. Leistner, L. Van Gool. Ensemble Kernel Learning. *IEEE International Conference on Computer Vision Workshops (ICCV Workshops), IEEE Workshop on Kernels and Distances for Computer Vision*, 2011. (Abstract and Poster).

2. M. Van den Bergh, G. Roig, X. Boix, L. Van Gool. Online Video SEEDS for Temporal Window Objectness. *European Conference on Computer Vision Workshops (ECCV Workshops). International Workshop on Video Segmentation (IWVS)*, 2014. (Invited Talk and Poster).