Learning and Matching Binary Local Feature Descriptors

THÈSE N^{O} 6226 (2014)

PRÉSENTÉE LE 25 AOÛT 2014

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

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Tomasz Piotr TRZCIŃSKI

acceptée sur proposition du jury:

Prof. M. C. Gastpar, président du jury
Prof. P. Fua, Dr V. Lepetit, directeurs de thèse
Prof. M. Bronstein, rapporteur
Prof. P. Vandergheynst, rapporteur
Prof. A. Vedaldi, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2014

Abstract

Contemporary Computer Vision applications, such as visual search or 3D reconstruction, need to handle massive amounts of visual content. This poses a significant challenge, especially when these applications are expected to run in real-time on mobile platforms with limited resources. One of the computational bottlenecks in a typical real-life Computer Vision system is establishing a set of correspondences across multiple views of a scene, which is known as the local feature matching problem. In this thesis, we address the local feature matching problem from two angles: first, by learning efficient and compact local feature descriptors, and second, by providing a set of fast methods to match the proposed descriptors using approximate nearest neighbor search.

We start by presenting two methods to construct robust binary local feature descriptors. To represent salient image regions in a way that is invariant to unwanted image transformations, we employ machine learning techniques, such as linear discriminative analysis and boosting. Our interest is mainly focused on binary descriptors, because they enable faster processing while requiring significantly less memory than their floating-point competitors. This approach enables modern Computer Vision applications, even those deployed on mobile devices, to process higher amounts of visual data much faster.

We first propose an extremely compact and quick to compute binary descriptor, D-Brief, that is built by projecting an image patch to a more discriminative subspace and thresholding the resulting coordinates. However, applying complex projections to the patches is slow which negates some of the advantages of binary descriptors. Hence, our key idea is to learn the discriminative projections so that they can be decomposed into a small number of simple filters for which the responses can be computed fast. The resulting 32-bit descriptor outperforms the state-of-the-art intensity-based binary descriptors, in terms of both accuracy and efficiency, and is therefore a perfect candidate for real-time applications run on low-end mobile devices.

We then introduce a more complex and powerful binary descriptor called BinBoost that provides much higher discriminative power and robustness, characteristics necessary for large-scale applications such as visual search. To efficiently train our binary descriptor, we leverage the *boosting-trick* and optimize consecutive binary dimensions so that they complement each other, which is key to robustness and compactness of the resulting descriptor. The final feature representation is computed with a set of hash functions that are applied directly to the image patches, which frees us from any intermediate representation and lets us automatically learn the descriptor sampling pattern. As a result, our formulation encompasses that of many hand-crafted descriptors and allows BinBoost to outperform the state-of-the-art binary and floating-point descriptors at a fraction of matching time and memory footprint.

Although binary descriptors can be matched much faster than their floating-point competitors, exhaustive search over truly large-scale datasets still remains problematic, especially on mobile devices. Hence, we investigate different approximate nearest neighbor search methods in the context of binary vectors, as they improve the efficiency of matching while maintaining a good matching quality. Unfortunately, the readily available ANN search methods for floating-point vectors fail on binary ones. We explain this phenomenon, that we call *thick Voronoi boundaries*, by analyzing the peculiarities of binary spaces and propose two effective ways to overcome this limitation by appropriately randomizing either a tree-based algorithm or hashing-based one.

Keywords: Computer vision, machine learning, binary local feature descriptors, binary approximate nearest neighbor search.

Résumé

Aujourd'hui, de nombreuses applications de vision par ordinateur, telles que la recherche d'images ou de la reconstruction 3D, doivent traiter du contenu visuel en grande quantité. Cela pose un défi de taille, surtout lorsque ces applications doivent fonctionner en temps réel sur des plateformes mobiles aux ressources limitées. L'un des goulots d'étranglement pour un système classique de vision par ordinateur est d'établir un ensemble de correspondances entre plusieurs vues d'une scène, ceci est connu comme le problème d'appariement de points d'intérêts locaux. Dans cette thèse, nous abordons le problème d'appariement de points d'intérêts locaux sous deux angles: d'abord, par l'apprentissage de descripteurs de points caractéristiques locaux efficaces et compacts, et d'autre part, en fournissant un ensemble de méthodes rapides pour apparier les descripteurs proposés en utilisant la recherche approximative du plus proche voisin.

Nous commenons par présenter deux méthodes pour construire de solides descripteurs binaires de point caractéristiques locaux. Pour représenter les régions d'image saillantes d'une manière qui soit invariante aux transformations indsirables de l'image, nous utilisons des techniques d'apprentissage automatique, comme l'analyse discriminante linéaire et boosting. Notre intérêt se concentre principalement sur les descripteurs binaires, car ils permettent un traitement plus rapide tout en exigeant beaucoup moins de mémoire que leurs concurrents à virgule flottante. Cette approche permet aux applications de vision par ordinateur modernes, même celles qui sont déployées sur les appareils mobiles, de traiter de plus grandes quantités de données visuelles beaucoup plus rapidement.

Nous proposons d'abord un descripteur binaire extrêmement compact et rapide pour calculer, D-Brief, qui est construit par la projection d'image

sur un sous-espace plus discriminant et par le seuillage des coordonnées obtenues. Toutefois, l'application des projections complexes aux images est lente ce qui annule une partie des avantages des descripteurs binaires. Par conséquent, notre idée clé est d'apprendre à decomposer les projections discriminatoires en un petit nombre de filtres simples dont les réponses peuvent être calculées rapidement. Le descripteur 32 bits résultant surpasse les meilleurs descripteurs binaires basés sur l'intensité, tant en termes de précision que d'efficacité, et est donc un candidat idéal pour les applications en temps réel exécutées sur les appareils mobiles bas de gamme.

Nous introduisons ensuite un descripteur binaire plus complexe et puissant appelé BinBoost dont la robustesse et le pouvoir discriminant sont plus élevés. Ce sont là, les caractéristiques nécessaires pour les applications à grande échelle telles que la recherche d'images. Pour entraîner efficacement notre descripteur binaire, nous utilisons le *boosting-trick* et optimisons des dimensions binaires consécutives de sorte qu'ils se complémentent mutuellement, ce qui est essentiel pour la robustesse et la compacité du descripteur résultant. La représentation finale est calculée avec un ensemble de fonctions de hachage qui sont appliqués directement sur l'image, qui nous libère de toute représentation intermédiaire et nous permet d'apprendre automatiquement le descripteur de motif d'échantillonnage. En conséquence, notre formulation englobe de nombreux descripteurs construits à la main et permet à BinBoost de surpasser les meilleurs descripteurs binaire et à virgule flottante en une fraction du temps et de l'occupation.

Bien que les descripteurs binaires peuvent être mis en correspondance beaucoup plus rapidement que leurs concurrents en virgule flottante, la recherche exhaustive sur des ensembles de données de grande échelle reste problématique, en particulier sur les appareils mobiles. Par conséquent, nous étudions différentes méthodes approximatives de recherche de plus proches voisins, utilisant des vecteurs binaires, car ils améliorent l'efficacité de l'appariement à faible coût. Malheureusement, les méthodes approximatives de recherche de plus proches voisins disponibles pour vecteurs à virgule flottante ne marchent pas bien pour les vecteurs binaires. Nous expliquons ce phénomène,

que nous appelons limites de Voronoi épaisse, en analysant les particularités des espaces binaires et nous proposons deux moyens efficaces pour surmonter cette limitation par randomisation appropriée.

Mots-clés: Vision par ordinateur, apprentissage automatique, descripteurs binaires de point caractéristiques locaux, méthodes approximatives de recherche de plus proches voisins binaires.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor Prof. Pascal Fua for giving me an opportunity to pursue my research interests and human curiosity under his wings. I would also like to sincerely thank my co-supervisor Prof. Vincent Lepetit for his insightful comments, numerous advices, and for the always opened door to his office. It has been a tremendous challenge, yet also a very rewarding one, that they have significantly contributed to and I will always remain grateful for that.

I also thank the members of my committee Prof. Michael Gastpar, Prof. Michael Bronstein, Prof. Pierre Vandergheynst and Prof. Andrea Vedaldi for accepting to evaluate this thesis.

My special thanks go to my CVLab colleagues, especially to Carlos with whom I had a pleasure of sharing the office (and too many delicious Havannas or Emilie's cupcakes). To Mario for all the support and great ideas that we developed together on a whiteboard and elsewhere. To Roberto for letting me set my watch according to his departures and for reminding me that there are more important things than computers (*e.g.* mountains). To Anne, Antoine, Aurélien, Horesh, Raphael and Yunpeng for all the more and less serious talks we had over the lunch table. To alumni, Karim and Aydin, and all the lab members who were always interested in discussing the research presented in this thesis. Last but not least, I would like to thank Josiane for her never-ending kindness and all the good words.

Finally, I give my thanks to my parents for their love and support throughout all the years. I would not be where I am without them. My deepest gratitude goes to my wife and lifelong friend, Gosia, for her positive attitude and continuous support and for the great love that we share.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Dedicated to my beloved son, Leon.

CHAPTER

ONE

INTRODUCTION

1.1 Motivation

A multitude of Computer Vision applications, such as visual search, 3D reconstruction or object recognition, is based on finding correspondences between images. This problem can be solved by representing salient image patches in a way that is invariant to illumination and viewpoint changes and finding the correspondences in the resulting representation space. The representation of an image patch, referred to as a local feature descriptor, can be typically defined as a multi-dimensional vector of floating-point or binary values. In addition to robustness to various image transformations, local feature descriptors should also possess a high discriminative power, *i.e.* they should be able to discriminate between patches corresponding to different 3D points. The problem of designing optimal feature descriptors has received a significant attention from the Computer Vision community [3, 10, 17, 21, 54, 62, 65, 92, 103, 114]. To model the non-linear nature of the above mentioned transformations, well-known local feature descriptors typically apply a set of expert-designed filters and aggregate or *pool* their responses within pre-defined regions of the image patch. Although significant progress has been made, most of the proposed solutions are mainly designed for standalone machines with high processing power and few memory limitations.

Today, in the era of ubiquitous mobile computing, there is an ever growing need for Computer Vision technologies which can be deployed on portable devices with limited resources. The increasing content of visual data along with the need to access this data from any point in the world and as quickly as possible require extremely fast processing. It also requires lowering the data throughput as mobile networks are still underdeveloped to handle the constant flow of megabytes of visual information.

To tackle these problems, several binary local feature descriptors have been proposed in the recent years [3, 21, 54, 92]. Their binary nature reduces their memory footprint and allows for significantly faster processing due to the fast computation of the Hamming distance. Furthermore, they are typically built as a concatenation of simple intensity comparisons which results in an extremely short computation time. Therefore, binary descriptors have quickly become an attractive alternative to floating-point descriptors, especially for real-time applications run on low-end handheld devices. Nevertheless, their performance quality still cannot reach the level of their floating-point competitors which blocks their proliferation into more quality-oriented applications such as visual search or image-based localization.

Another approach to mobile computing is to use machine learning to find an optimal representation of an image patch [17, 98, 103]. By optimizing over the selection of pooling regions and filter responses, these solutions aim at reducing the dimensionality of the resulting representation and, hence, decreasing the computational burden of matching and storing them. These approaches, however, are either built on top of hand-crafted representations [98] or still require significant parameter tuning, as in [17] that relies on a non-analytical objective that is difficult to optimize. Finally, they typically result in floating-point descriptors which requires matching with Euclidean distance instead of Hamming distance and further increases the computational cost.

Although the recent techniques to compute binary local feature descriptors allow for using fast Hamming distance computations instead of the Euclidean ones to compute similarities between the descriptors, it still remains prohibitively expensive when using linear search to find the nearest match in a database of millions or billions of datapoints. Approximate Nearest Neighbor search is therefore a valid alternative that enables truly large-scale processing of visual data. A lot of research has been focused on providing working solutions for approximate nearest neighbor search in the multi-dimensional space of floating-point vectors [62, 73, 78]. Surprisingly few works, however, focus on this problem in the case of binary descriptors [74]. One explanation of this situation might be that under favorable conditions binary vectors can be used as individual indices to access the memory and hence nearest neighbor matching amounts to simple look-ups. Unfortunately, this approach does not provide a feasible solution when the

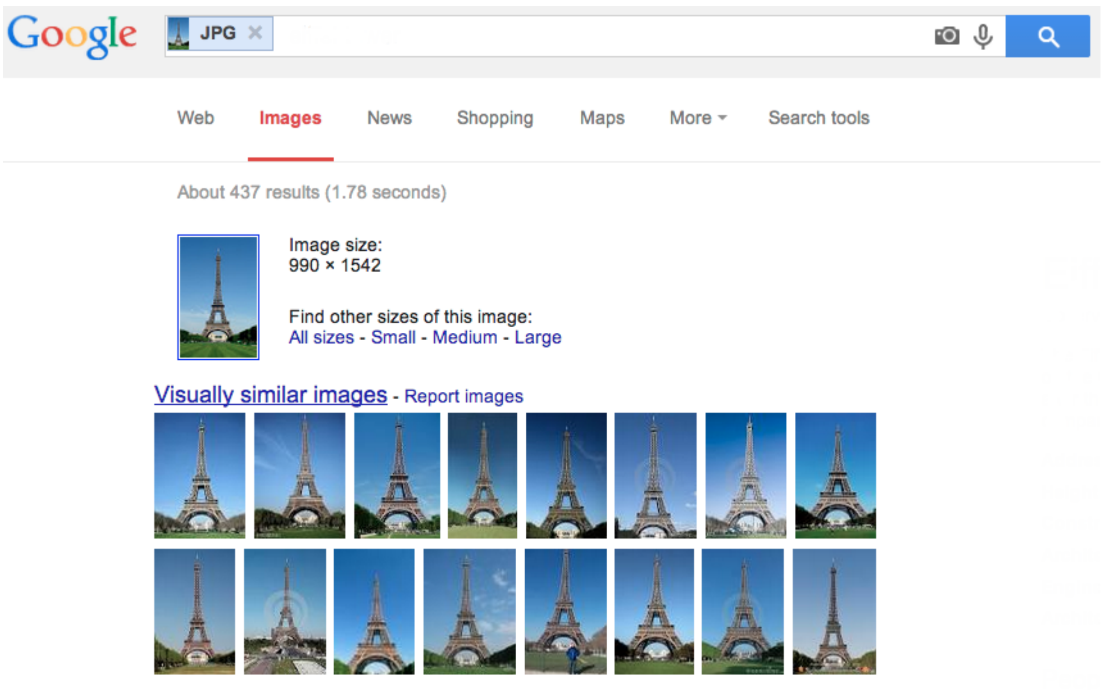

Google
JPG
Web
Images
News
Shopping
Maps
More
Search tools
About 437 results (1.78 seconds)
Image size:
990 × 1542
Find other sizes of this image:
All sizes - Small - Medium - Large
Visually similar images - Report images

Figure 1.1: `images.google.com` relies on Computer Vision techniques for querying a database of images by finding similarities between local feature descriptors.

size of the binary vectors becomes larger than a few bits. It is therefore necessary to investigate if the solutions available for the floating-point vectors can be used to find approximate nearest neighbors also in the case of binary descriptors.

1.2 Applications

The image patch representation that is robust to various illumination and viewpoint changes, plays a crucial role in a plethora of Computer Vision applications. Among the most important ones are:

Large-scale Visual Search

Thanks to Google and its text-based search engine, the idea of searching the web with a set of words is a natural way of discovering the world and answering various questions. However, in the world of ubiquitous mobile cameras, the users become more and more interested in querying the Internet not only with textual hints, but also with visual data. As a matter of fact, images can convey much more contextual data than even the most descriptive phrase. Thus, there is a significant demand to provide solutions for visual search which can be defined as retrieving information, both textual and visual, in response to a visual query, *e.g.* in the form of an image. The most popular framework developed to answer this need [105] relies on a set of quantized local feature descriptors, so called *visual words*. After the quantization, each image can be represented with a histogram of visual words and a set of corresponding weights. This process is repeated for descriptors of a query image and the database images are ranked according to the dot product between the query weight vector and the corresponding vectors of database images. After the first ranking step, the list of images is typically re-ranked using spatial verification [84] which uses the projective geometry constraints to detect the correct transformation between the query and reference image. Fig. 1.1 presents a screenshot of a representative example of a large-scale visual search application, namely `images.google.com`, which is already available to the public. Other use cases of visual search engine include image-based localization [88] and structured medical image retrieval [102]. Finding more discriminative and compact binary representations can potentially lead to increased quality of visual search as well as faster processing, *e.g.*, in the spatial verification step.

3D Reconstruction

One of the recent Computer Vision applications that was enabled by the abundance of visual information is automatic image-based 3D reconstruction which can be described as a process of reconstructing 3D models of objects seen in a collection of images from different vantage points. Fig. 1.2 shows an example of a 3D model reconstructed from a set of images. A crucial component of 3D reconstruction systems is matching local feature descriptors to provide the correspondences between the images representing the same object. Since this operation is usually performed on a significant number of images, extraction as well as matching of the descriptors should be very fast to enable efficient processing. Another requirement of 3D reconstruction system is the reliability of the extracted feature descriptors in terms of description of the same 3D points. To provide a solid set of correspondences between images, local feature descriptors have to be robust to various illumination and viewpoint changes and the distance between descriptors representing the same 3D point should be significantly smaller than the distance between distinct 3D points. Thus, using machine learning methods

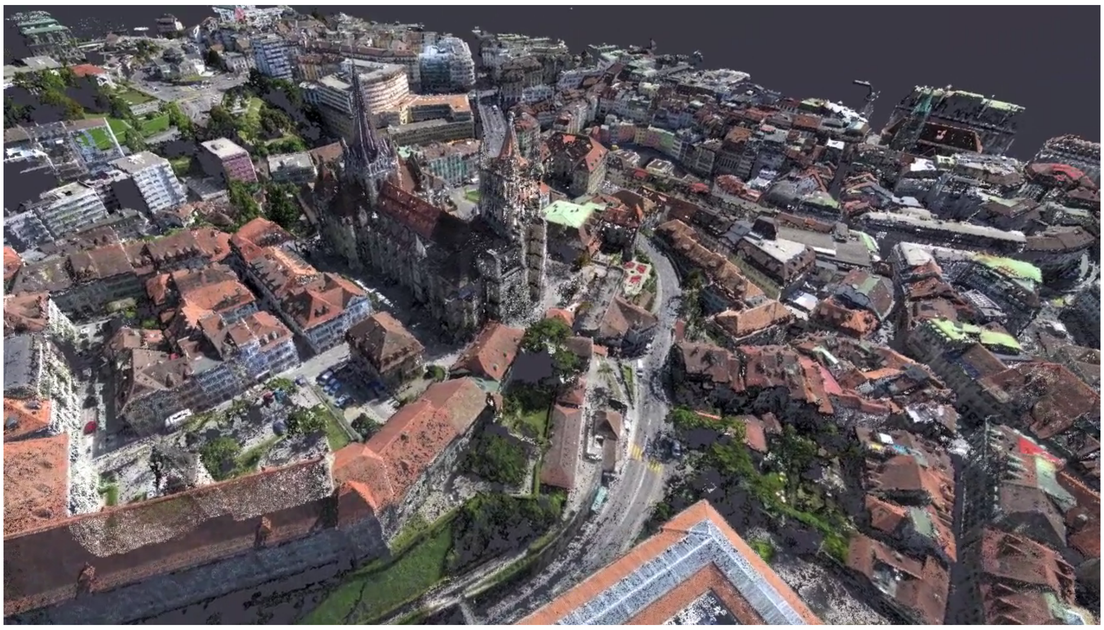
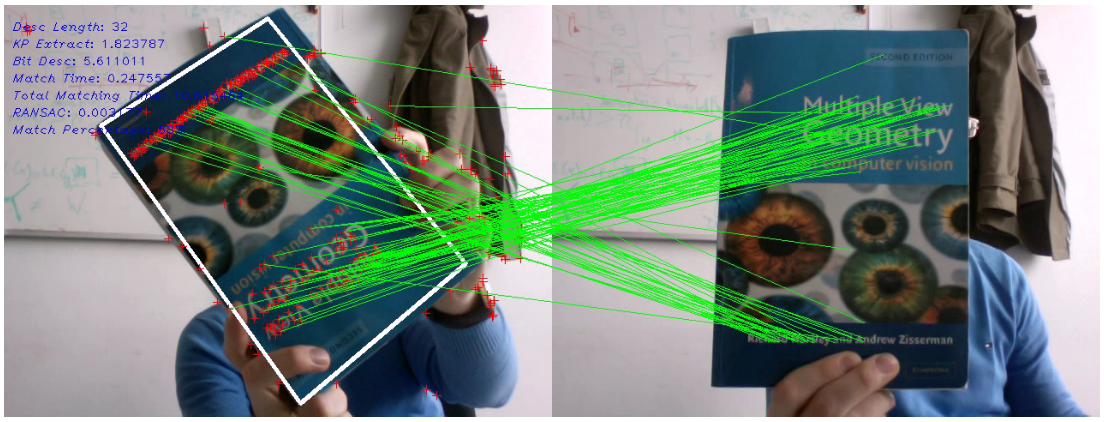


Figure 1.2: A textured 3D model reconstructed from the images of Lausanne, Switzerland. Courtesy of Pix4d (`pix4d.com`).

to obtain an image patch representation that can reliably discriminate between similar and different 3D points can immensely contribute to a better performance of many 3D reconstruction systems.

Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping refers to a task performed by a robot whose goal is to build a map of the surrounding, unknown environment while keeping track of its own location. This problem when it involves cameras as sensors, also called V-SLAM for Visual SLAM, poses a significant challenge due to the reduced computational budget of a typical CPU used by robots. The problem is further exacerbated by the noise of measurements performed by robot's sensors. Furthermore, SLAM can be considered a chicken-egg problem as an accurate position is required to build a precise map, while a correct map is needed to determine one's position. Due to the above mentioned difficulties SLAM has gained a significant attention from the computer vision community [7, 28, 29]. One of the fundamental steps to solve the SLAM problem requires assigning subsets of image features to common 3D world points. Since this assignment assumes certain level of robustness of the features with respect to typical transfor-



```
Desc Length: 32
KP Extract: 1.823787
Bit Desc: 5.611011
Match Time: 0.247557
Total Matching Time: 10.615464
RANSAC: 0.003177
Match Percentage: [illegible]
```

Figure 1.3: Screenshot of a running object detection application.

mations and robot's viewpoint changes, the resulting partitioning depends highly on the quality of selected feature descriptors. Moreover, real-time image processing on a mobile platform poses yet another challenge to the underlying feature extraction and matching algorithm, as the computational power of the mobile CPU is highly limited. Hence, it becomes evident that SLAM, or more precisely V-SLAM, would immensely benefit from using compact and discriminative binary local feature descriptors which are fast to match and can reduce memory footprint.

Object Detection

Given an object presented in one or more images, we define object detection as the process of finding the object in each image. If the object can be found, the system should return a set of image coordinates defining the location of the object, the so-called region of interest, otherwise it should inform that the object is not present in the scene. Furthermore, we can define *instance-level* object detection when we are interested only in a given object and *category-level* object detection when we look for all the objects of the same category as the selected object, *e.g.* when we look for all cars or all chairs. For the purpose of this thesis, we focus on the instance-level object detection problem, as successful approaches to solve it rely heavily on local feature descriptors due to their robustness to occlusions and various transformations. An example of instance-level object detection application can be seen in Fig. 1.3 where the object selected by the user is detected and tracked across the video frames. Performing this kind of tasks in real time, *i.e.* with more than 25 frames per second, requires fast feature description and descriptor processing. These are exactly the constraints that binary descriptors can successfully address and, thus, they give us yet another motivation for the work on compact descriptors that can be efficiently computed.

1.3 Contributions

In this thesis, we first propose two novel methods to develop robust binary local feature descriptors. We essentially try to bridge the performance gap between the state-of-the-art binary and floating-point descriptors without increasing the computational cost of computing and matching our binary descriptors. To that end, we employ machine learning techniques, such as linear discriminant analysis and boosting, that allow us to learn the invariance desired for local feature descriptors from the data. Since local feature descriptors are typically employed in more complicated systems, the shift from floating-point to binary descriptor type has to be accompanied with a corresponding change in many other elements, one of them being nearest neighbor search. Thus, this thesis also provides an analysis of the multi-dimensional space generated by binary descriptors along with an approach that addresses the peculiarities of this space.

More precisely the contributions of this thesis are the following:

- An efficient method for approximating discriminative projections which enables us to quickly compute an extremely compact binary descriptor we call D-Brief [118] (**Chapter 3**).
- A boosting-based framework to learn a highly discriminative non-linear binary descriptor called BinBoost that encompasses various hand-crafted descriptors, while significantly improving their performance [119, 121] (**Chapter 4**).
- An extensive study of the performances of proposed binary descriptors that show a clear advantage of using the binary descriptors over the state-of-the-art floating point descriptors, both in terms of quality and computational cost (**Chapter 5**).
- An analysis of the partitioning of the space of binary descriptors, which we call the *thick Voronoi boundary* problem [120] (**Chapter 6**).
- A set of solutions for the thick Voronoi boundary problem that enable effective approximate nearest neighbor search for binary vectors [120] (**Chapter 6**).

To summarize, the work presented in this thesis provides a set of algorithms to learn compact binary descriptors which address both the complexity and quality requirements of mobile Computer Vision applications. Since the shift from floating-point to binary descriptors has a significant impact on many elements of those applications, including the matching step, we also analyze the characteristics of approximate nearest neighbor search in binary spaces and provide a set of methods that successfully adapt the existing techniques for matching local feature descriptors.[1]

1.4 Outline

This thesis is organized as follows. The next chapter provides an extensive overview of the state of the art in the field of local feature detection and description. In Chapter 3 we propose a method for building compact binary representations that relies on efficient approximations of discriminative projections. Chapter 4 introduces a more complex learning algorithm based on boosting which allows us to build a highly non-linear binary representation of image patches. We then compare the above mentioned methods against the state of the art in Chapter 5. In Chapter 6 we analyze high-dimensional spaces generated by binary descriptors in terms of nearest neighbor search and we explain the crucial differences between floating-point and binary approximate nearest neighbor search. Finally, Chapter 7 provides final remarks and discusses future research.

[1]The source codes for the binary descriptors, approximate nearest neighbor methods and experiments reported in this thesis can be found at the address: `http://cvlab.epfl.ch/research`.

CHAPTER
TWO

RELATED WORK

In this chapter we discuss works related to local feature descriptors. Our goal is to provide an extensive overview of the currently available feature descriptors. Nevertheless, the number of proposed feature descriptors is still growing and, hence, providing a description of all the methods remains a very challenging task. We therefore focus here on the works that remain the most relevant for this thesis.

Let us first define the concept of an *image feature* as an image structure that provides a distinctive piece of information relevant for a given task. Typical image features, such as points or regions, can be characterized by a given property, *e.g.* a dominant color or orientation of a particular set of pixels. The representation of this characteristic, which we refer to as a *image feature descriptor*, is a corresponding scalar or vector that comprises the information about a given image. Depending on the final application, feature descriptors should provide a certain level of robustness towards image transformations such as image scaling, rotation, illumination and viewpoint changes. Due to the complexity of the image registration process as well as some intrinsic difficulties of mapping 3D objects into 2D image planes, obtaining this kind of invariance remains a challenging task, not only for machines but also for humans.

We can divide image feature descriptors based on the image structure that they aim to describe. Image feature descriptors that represent the entire image are typically referred to as *global feature descriptors*, as their goal is to summarize all the information provided in the image [75, 112, 122]. One well-known example of this type of descriptor is GIST [82] which relies on a set of orientation histograms computed over a grid of image regions. Since global feature descriptors aim to provide a very compact representation of multiple objects shown in the image, their robustness to occlusions and diversified scene compositions is fairly low. To solve this problem, an alternative approach that relies on multiple *local feature descriptors* has been proposed [62, 95, 97, 124]. Local feature descriptors, contrary to the global ones, provide a representation of an image patch, instead of the entire image. Although this approach is more robust against occlusions, it requires a stable method to find salient image patches in the image. This task is typically referred to as a *feature* or *interest point detection* and it should be clearly distinguished from the process of encoding a given salient image patch into a feature descriptor, which is defined as *feature description.* In this thesis, we focus on the latter part, *i.e.* on local feature description, however here we provide an overview of the feature detection algorithms as well for completeness.

A notable group of feature descriptors called *dense descriptors* does not rely on the detection algorithm and computes the image region representation for each element of a regularly spaced grid [58, 114, 117]. The motivation is the fact that in some settings uniform regular sampling-based descriptors were reported to outperform detection based descriptors [27, 81]. This approach is truly interesting when the number of key points to be detected is so high that avoiding altogether the detection step and describing each consecutive image patch leads to computational savings. Although in this thesis we focus on the detection-based feature descriptors, the methods presented here are detector-invariant and, hence, one can image their potential extensions to the dense descriptor case.

Perhaps the simplest way of building a local feature descriptor of a given image patch is by using its pixel values, *i.e.* by encoding the image grayscale values into a multi-dimensional vector. This method is typically combined with a matching algorithm based on the Euclidean distance and is known under the name of *sum of squared differences* (SSD) [44]. Along with its extension, the *normalized sum of squared differences* (NSSD) it is still used in applications where the differences in image viewpoint or illumination between the analyzed images are not significant. It is, however, fairly sensitive to more complex transformation.

As a result, more robust algorithms have been proposed for describing image feature points using other types of image information, *e.g.* gradients, image moments, higher order image derivatives or various filter responses [11, 50, 62, 95, 129]. Among those methods, the most well-known is the Scale Invariant Feature Transform (SIFT) descriptor [62]. Although highly discriminative and robust to various image transformations, SIFT relies on computationally expensive gradient histogram pooling and results in a high dimensional feature vector of floating-point values. To reduce the computational complexity, the Speeded Up Robust Features (SURF) [11] descriptor substitutes gradient pooling by aggregating Haar wavelet responses and reduces the dimensionality of the output descriptor from 128 to 64.

To decrease the dimensionality of local feature descriptors, few recent methods propose to construct efficient descriptors using compression or quantization techniques [14, 24, 48, 49, 133]. These approaches, however, rely on an underlying high-dimensional feature representation, such as concatenated responses of gradient filters, and have to be therefore coupled with a particular compression technique, such as Huffman coding to present a valid alternative to SIFT or SURF.

In order to provide even more compact descriptors, recent works propose several approaches to design binary, not real-valued, local feature descriptors [3, 21, 54, 92, 93, 136], since they require much less storage. They also enable faster matching as there are efficient indexing schemes for binary vectors [36, 131] and Hamming distances can be computed fast on many architectures, including mobile devices. Thanks to all these properties, binary descriptors are perfect candidates for real-time applications. The most recently proposed binary feature descriptors [3, 21, 54, 92, 93] are computed by applying simple tests directly to the image patch . These tests compare the image intensities at two pixel locations and, hence, they are extremely fast to compute. For noise removal, the image is pre-smoothed with a box filter or a Gaussian filter. BRIEF [21] uses random pre-determined locations, whereas BRISK [54] uses an exhaustive set of comparisons of close locations. ORB [92] relies on optimization like we do and aims at improving the recognition rates by choosing the locations that decorrelate the tests. Similarly, FREAK [3] selects the intensity tests that provide the highest bit variance. Thanks to the simplicity of the above mentioned binary descriptors, they quickly attracted a lot of attention in the Computer Vision community and have become a very popular element of many real-time applications [19, 109]. Their performances, however, suffer from a relatively moderate discriminative power of intensity comparisons and leave some place for improvement.

To improve the performances of hand-crafted binary descriptors, much research has been done on learning the optimal image patch representation. Both unsupervised and supervised approaches have been proposed. Unsupervised approaches seek for a transformation that preserves the similarity as evaluated in the original space, typically using the Euclidean distance [6, 37, 131]. The supervised approaches aim to outperform unsupervised ones by exploiting labeled data to produce similar binary representations for data with the same class labels [98, 111, 116, 128]. For example LDAHash [111], one of the most successful supervised approaches, computes a binary descriptor using discriminative projections applied to SIFT descriptors. Nevertheless, most of the data-driven methods mentioned above, including LDAHash, are applied to a sophisticated descriptor, such as SIFT or GIST [115], and thus can lead to unnecessary computational overhead and in many cases are limited to the accuracy of the original input space. Recent works have emphasized the importance of learning the entire descriptor design starting from raw image patches [17, 101, 103, 104] and in this thesis we follow this approach to learn our binary local feature descriptors.

In the remainder of this chapter we first discuss the algorithms for detecting salient image patches, *i.e. feature detection* methods. Then, in Section 2.2 we outline the second step of generating local feature descriptors that is *feature description.* We start by presenting the details of the most prominent floating-point local feature descriptors, namely SIFT and SURF. Following the evolution of the research on local feature descriptors we introduce the intensity-based binary feature descriptors such as BRIEF, BRISK and ORB. We conclude this chapter with an overview of the machine learning algorithms to build compact local feature descriptors.

2.1 Feature Detection

An ideal local feature detector would be able to identify features that correspond to semantically meaningful objects or their parts. Due to the complexity of this task, however, it is infeasible as it requires a deeper understanding of the scene presented in the image. Instead, currently available feature detectors modestly aim at detecting image patterns that differ significantly from their imminent neighborhoods. Consequently, good detectors should be able to provide features that are [123]:

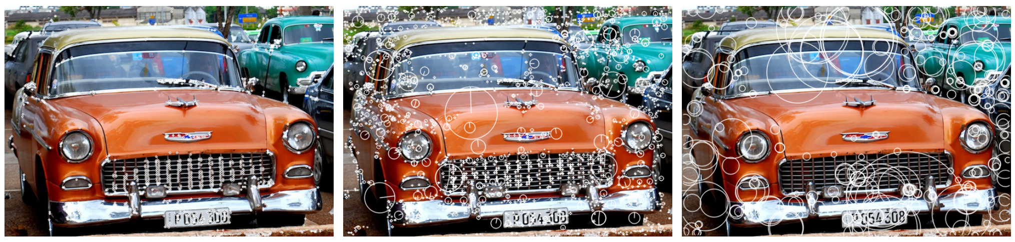

(a) Corner features (b) Blob features (c) Region features

Figure 2.1: Features detected with different detection algorithms. (a) Harris corners. (b) SIFT features (Difference-of-Gaussian points). (c) MSER features [68].

- *repeatable*: the overlap of features detected in two images of the same object should be high, regardless of the change in viewing conditions.
- *distinctive*: the underlying intensity pattern of the features should allow to distinguish and match features.
- *local*: the detections should be local to enable simple model approximations and avoid occlusions.
- *numerous*: there should be a multitude of features detected in a typical image.
- *accurate*: the features should provide the exact location, also with respect to scale, and preferably more, *e.g.* feature orientation.
- *efficient*: the detection should be fast as this plays a significant role, especially in the case of real-time applications.

Depending on the feature type detected in the image, feature detectors can be divided into three categories: corner, blob and region detectors. An example of the features detected using different detectors are shown in Fig. 2.1. Discussing all the feature detection algorithms proposed in the literature is beyond the scope of this thesis and, thus, we focus here on the most prominent and widely used methods. More precisely, we start by presenting Harris corner detector [41] and its efficient alternative designed for real-time applications - FAST [90]. We then discuss more complex, yet more robust blob detectors: Difference of Gaussian [61] and Determinant of Hessian [57].

2.1.1 Harris Detector

The Harris detector [41], also known as the *Harris operator* is one of the first successful attempts to localize image features. It proposes to use a measure of *cornerness*, defined as the response strength of the following operation:

$$\text{cornerness} = \det(M) - \lambda \operatorname{trace}(M) \tag{2.1}$$

where M is an auto-correlation matrix describing the gradient distribution in a local neighborhood of a point, *i.e.*:

$$M = \sum_{u,v} w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.2}$$

with $w(u, v)$ representing averaging with a Gaussian smoothing function, and I_x and I_y being the image derivatives along x and y axis, respectively. A typical value for λ is 0.04. The matrix M can be useful to detect corners, as its eigenvalues correspond to the principal signal changes in two orthogonal directions and so large eigenvalues represent significant variation of the signal which can be interpreted as a corner. To simplify computations, instead of performing eigenvalue decomposition, the Harris detector calculates the difference between the determinant of matrix M and its scaled trace. Despite the efficiency of such computations, the elementary Harris detector provides only moderate performance as it is not robust enough for various viewpoint and scale changes [69].

Several extensions of Harris detector have been proposed, one of the most successful ones being Harris-Laplace and Harris-Affine detectors [70]. The Harris-Laplace operator introduces scale invariance by means of a scale space [55, 57] and, hence, allows to detect stable feature points across different scales and image resolutions. The Harris-Affine detector builds up on [56] to obtain affine invariant corners. Instead of detecting circular neighborhoods, the Harris-Affine detector iteratively estimates elliptical affine regions. This approach provides a better estimate of the object shape, regardless of transformations caused by the viewpoint changes.

2.1.2 FAST

Although the extensions discussed above significantly improve robustness of the Harris corner detection method, they also lead to a massive increase of computational

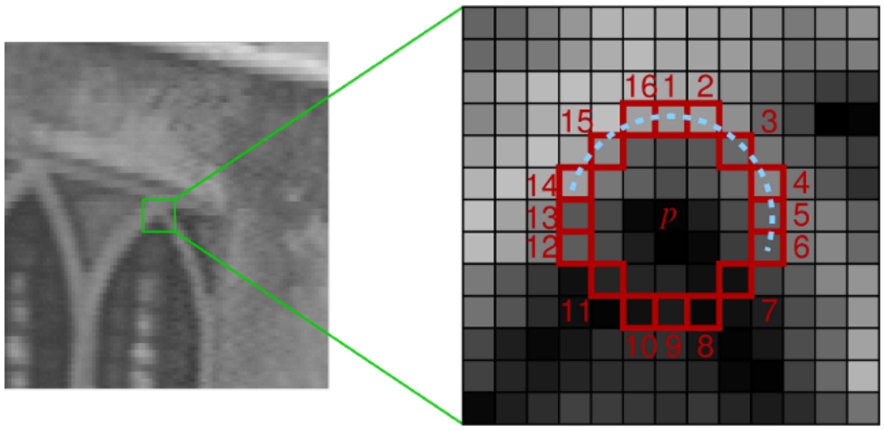


Figure 2.2: Selection of the circle pixels used to detect corners in FAST. Source: [90].

complexity. To overcome this limitation, which becomes especially important for the real-time applications, Smith and Brady proposed the SUSAN detector [106]. Instead of using local gradient information, SUSAN relies on a simple morphological analysis of the neighboring intensity values and, based on the proportion of similar to different pixel values, it decides whether to classify the central point as a corner. Building up on this idea, FAST [90] suggests using only the intensities of the pixels lying on a circle of radius 3. To classify the central point as a corner, FAST uses a set of comparisons between the intensity values of the pixels on the circle and the central pixel. The selection of the comparisons is entropy-driven and if n consecutive comparisons are consistent, $i.e.$ if all n pixels on the circle have higher (or lower) intensity than the central one, the central pixel is considered to be a corner candidate. The process is concluded with a non-maximum suppression stage [76]. Fig. 2.2 shows angraphical interpretation of the FAST corner detection. The scale invariance of the FAST detector is obtained by modifying the size of the area that is examined. Since intensity lookups are fairly inexpensive in terms of computational cost, the FAST detector can run up to 30 times faster than the competing state-of-the-art feature detection methods.

2.1.3 Difference of Gaussian

One of the most well-known blob detectors is the Difference of Gaussian (DoG) function. It can be interpreted as an approximation of a more complex, but also effective blob detector, namely Laplacian of Gaussian (LoG). In the case of images, we define the

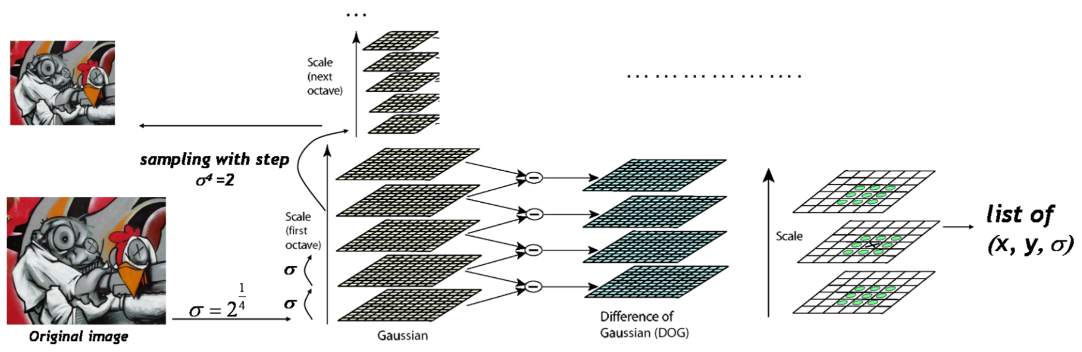


Figure 2.3: Overview of the DoG blob detection scheme. Source: [123].

LoG operator as a trace of the Hessian matrix, *i.e.* $LoG = \text{tr}(H)$ where

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \tag{2.3}$$

with I_{xx} a second-order Gaussian smoothed image derivative along the x axis. Although intrinsically LoG is not affine invariant, an extension similar in spirit to this for Harris corner detector was proposed [70].

Nonetheless, the computational complexity of applying the LoG operator makes this approach quite cumbersome and therefore a more efficient Difference of Gaussian is preferred. Introduced by Lowe and coupled with the SIFT descriptor [61], DoG operator avoids computing second-order derivatives inherent in the LoG operator by calculating differences between Gaussian-blurred images representing different scales. This approach is consistent with the diffusion equation in a scale-space theory, since it accounts for approximating derivatives in the scale direction.

The scale-space pyramid is computed from the bottom to the top where each consecutive image is a smoothed version of the preceding and the transition from one octave to the next corresponds to downsampling the image by a factor of 2

Fig. 2.3 shows the process of applying the DoG operator in practice. Generation of the image scale space is very efficient, since the pyramid is build from the bottom to the top and each consecutive image is a Gaussian smoothed version of the preceding image. Additionally, images at higher octaves are subsampled versions of the original and therefore Gaussian blurring can be applied even faster.

Once the scale-space pyramid is built, a non-maximum suppression is computed over the DoG blob response maps. The candidate locations defined as extrema in

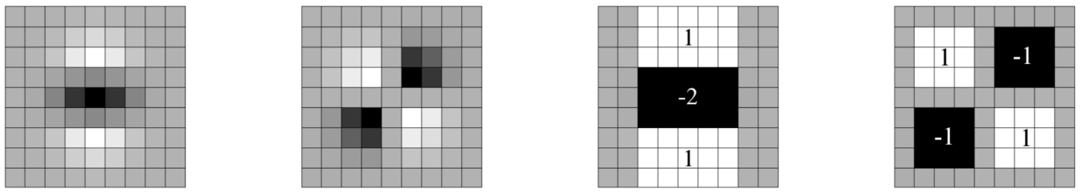


Figure 2.4: Example approximations of Gaussian second order derivatives proposed for SURF feature detector. From left to right: partial derivative in x and xy direction, their approximations with box filters. Source: [10].

the response maps, are further refined with quadratic interpolation. Since Laplacian (approximated by the DoG operator) gives fairly strong responses on edges, additional filtering step based on Harris cornerness score is necessary.

2.1.4 Determinant of Hessian

The determinant of Hessian (DoH) is another blob detection method that can be derived from the Hessian matrix. Instead of approximating the trace of Hessian as it is done in DoG, the DoH operator looks at its determinant. Although the performance of DoH in terms of scale selection was proven superior to this of DoG [57], its popularity can be mainly traced to the success of the SURF [10] feature detector which ingeniously approximates DoH using efficient integral images. Hence, we will discuss those two detectors together.

To provide scale-space invariance, the DoH operator, similarly to the Harris corner detector, should be coupled with the appropriate Laplace operator across the scale direction. Nonetheless, the SURF detector suggests using the determinant of Hessian not only in the spatial domain, but also across the scales. More precisely, Bay *et al.* propose to use box filters to roughly approximate Gaussian kernels as shown in Fig. 2.4 and use the resulting approximations to generate the scale-space pyramid. This step, when combined with integral images [126], leads to over five-fold speedup with respect to the DoG operator with virtually no performance loss, since there are many more important sources of noise present in the detection pipeline.

2.2 Feature Description

Having detected feature points in the image, the next objective is to represent them in a way that is invariant to unwanted image transformations. We outline this task, defined as *feature description*, in this section. We start by discussing one of the most successful local feature descriptors: SIFT [62]. Despite its wide recognition in the Computer Vision community, SIFT remains computationally prohibitive for many applications and several simpler methods were proposed. We therefore give a short overview of those methods, starting with SURF [10]. We continue with the recent advances in the theory of local feature descriptors, namely the shift from floating-point descriptors to binary ones, which was sparked by BRIEF [20]. Although orders of magnitude faster than the state-of-the-art floating point descriptors, BRIEF was criticized for its lack of invariance towards scale and rotation and, hence, several extensions were proposed [3, 54, 92] and we discuss them briefly in the following sections. We conclude this part by outlining the most recent trend in the field of local feature descriptors: data-driven approaches used for learning local feature descriptors.

2.2.1 SIFT

SIFT local feature descriptor was introduced by Lowe in his seminal work on distinctive image features [62]. Although many competing algorithms have been proposed, SIFT still stands out of many Computer Vision algorithms as it provides a fairly stable performance regardless of the application and image registration conditions. Moreover, it has become a *de facto* standard descriptor for matching objects and scenes [15, 30, 39, 84, 107] underlying various applications, such as visual search [78] or panorama stitching [16]. We therefore outline the key concepts behind the SIFT descriptor in this section and we compare the performances of our binary descriptor to SIFT in Chapter 5.

Although theoretically, computation of local feature descriptors can be independent from feature detection step, Lowe proposes to build SIFT descriptors for features detected with the DoG operator, described in details in the previous section. The rational behind this decision is the goal of obtaining scale and rotational invariance of the resulting representation. Since DoG provides a fairly stable set of candidate feature points across different scales, the scale invariance is guaranteed for the candidate features. Orientation invariance, however, has to be provided separately. This requires an effective method of assigning a repeatable orientation to the feature points.

To that end, SIFT takes the feature points, also called *keypoints*, along with the detected scale and selects from the scale-space pyramid the image $L(x, y)$ that corresponds to the the closest scale to the keypoint's actual scale. For a given feature point at location (x, y) and scale s, the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are precomputed using pixel differences:

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2}, \quad (2.4) \\ \theta(x,y) &= \arctan\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right). \quad (2.5) \end{aligned}$$

Orientation values $\theta(x, y)$ around the feature location are used to form a 36-bin histogram which covers 360 degrees. When adding orientation samples to the histogram, each sample is weighted by the corresponding gradient magnitude and Gaussian-weighting centred on the key point. Eventually, the dominant orientation is determined according to the histogram bin with the highest peak. If the histogram contains other peaks within 80% of the value of the highest peak, the feature is replicated and the additional orientations are assigned to the newly created features. Although this step seems redundant at first, it contributes significantly to the increased stability of SIFT.

In the final step, local gradient information is summarized into a 128-dimensional descriptor by concatenating sub-regional histograms of gradients around a feature point, as shown in Fig. 2.5. Although one could imagine using simpler descriptors, *e.g.* intensity histograms, accumulating gradients is inspired by a model of biological vision that relies on complex neurons in primary visual cortex [32]. According to this model, neurons in visual cortex respond to gradients at particular orientations, but the exact location of the gradient response does not have to be precisely defined. SIFT descriptor is therefore computed by first computing a set of 8-bin orientation histograms in 4×4 sample regions and then concatenating them into a $4 \times 4 \times 8 = 128$ dimensional representation. Since the keypoint dominant orientation is known at that point, the histograms can be offset accordingly, thus providing rotational invariance. Finally, the resulting vector is clipped and renormalized to unit length to increase descriptor's robustness against affine illumination changes.

As a final remark, we would like to point out that although SIFT enjoys fairly wide recognition as one of the most robust local feature descriptor, its computational scheme

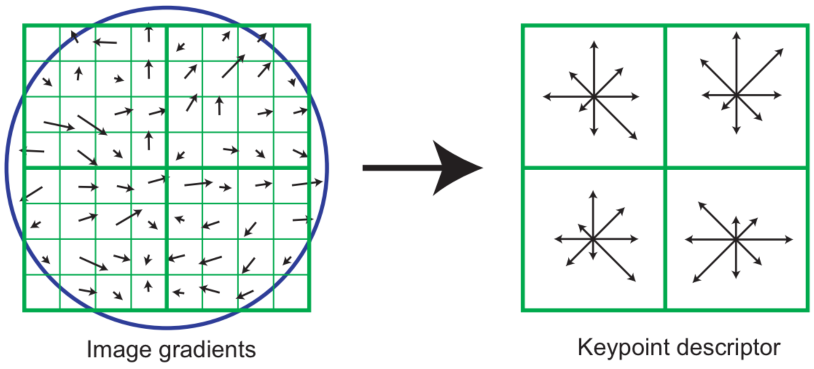


Figure 2.5: Pooling scheme of SIFT. Descriptor is built by first applying Gaussian weighting on local gradients (left) and then accumulating the resulting weighted gradients into a set of histograms (right). The histograms are then concatenated to form a final representation. Source: [62].

comprises a multitude of fine-tuned and meticulously engineered steps. In this thesis we try to understand the process of building a descriptor from a data-driven perspective and surprisingly discover that many of those steps are coherent with Lowe's findings, which proves the ingenuity of the SIFT algorithm.

2.2.2 SURF

Conceptually similar to SIFT, SURF significantly improves the computational efficiency of feature extraction process by leveraging the use of integral images and approximating gradient computations with Haar wavelets as shown in Fig. 2.6(a). Although this might seem to be a very radical step, the results show that in favorable conditions SURF can outperform SIFT. As a result, SURF has quickly become well-known as a more efficient alternative for SIFT and since we compare the performances of the descriptors proposed in this thesis against SURF, we briefly outline the process of generating SURF descriptors in this section.

After efficient feature detection presented in Section 2.1.4, SURF computes a set of orientation estimates dx and dy by means of scale-adapted Haar wavelets and weights them with an appropriate Gaussian kernel. The weighted responses dx and dy are then plotted on a corresponding 2D plane as $(dx, dy)^T$ points and a sliding window of size $\pi/3$ is finally used to estimate the dominant keypoint orientation (see Fig. 2.6(b)). Since orientation information might not be required for certain applications, SURF

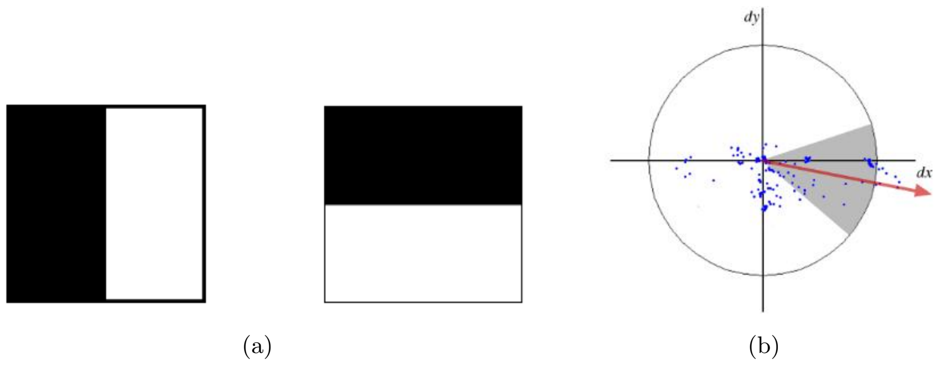


Figure 2.6: (a) Haar wavelet filters used to build SURF descriptors. Black areas are associated with a weight of +1, whereas white areas are associated with a weight of -1. (b) SURF orientation assignment mechanism. Source: [10].

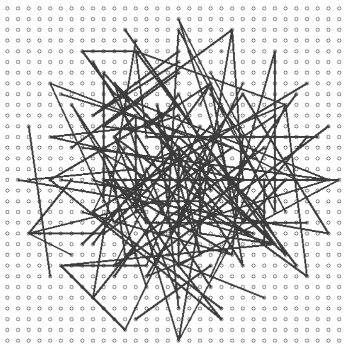
is also available in the *upright* version where the orientation computation is skipped and all keypoints are assigned with an orientation of 0. This allows for even further reduction in computational time.

The SURF descriptor, similarly to SIFT, aggregates filter responses in 4×4 oriented square sub-regions. Contrary to SIFT, however, the filters are the above mentioned Haar wavelets and the aggregation step accounts for simple summing which can be done efficiently by using integral filters. The final descriptor is then built by concatenating the sums, and results in 64-dimensional vector. Note that reducing the dimensionality by half with respect to SIFT leads to further speedup when it comes to matching SURF descriptors.

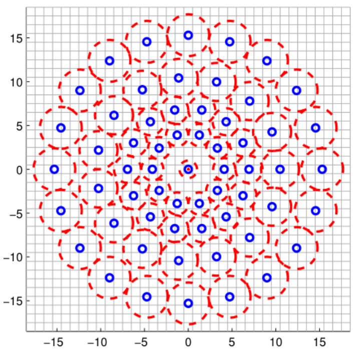
2.2.3 BRIEF and Its Extensions

Thanks to all the simplifications, SURF features can offer a significant speedup with respect to SIFT. Nevertheless, its use in real-time applications remain prohibitive, unless coupled with highly optimized GPU implementations [21]. Furthermore, with 64 dimensions of floating-point values, the memory footprint of SURF reaches 256 bytes. This becomes significant when millions of descriptors need to be processed and stored, especially on limited mobile devices.

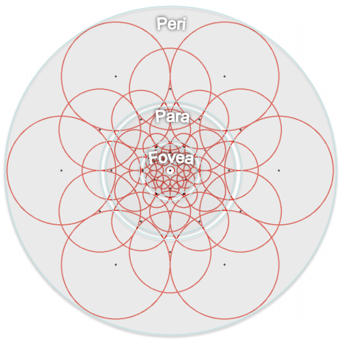
To address those limitations, Calonder *et al.* propose an intensity-based binary descriptor, called BRIEF, that is build by concatenating the results of simple comparisons



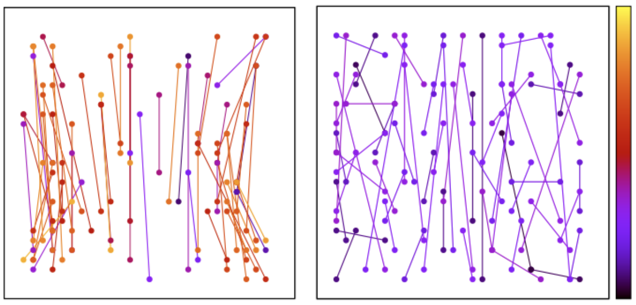
(a) BRIEF



(b) BRISK



(c) FREAK



(d) ORB

Figure 2.7: Sampling schemes of the intensity-based binary descriptors. (a) BRIEF uses random Gaussian sampling. (b) BRISK proposes a determininstic non-overlapping sampling regions pre-smoothed with various Gaussian kernels. (c) FREAK postulates using overlapping sampling regions pre-smoothed with Gaussian kernels of sizes related to log-polar retinal pattern. (d) ORB takes the initial set of intensity comparisons generated by considering high-variance under orientation (left) and decorrelates them to form the final descriptors (right). Source: [3, 20, 54, 92].

of pixel values at pre-defined sampling locations [20]. The sampling locations are drawn at random from a Gaussian distribution relative to the center of the image patch (see Fig. 2.7(a)). As a matter of fact, the intensity comparisons used to build BRIEF can be interpreted as primitive approximations of gradients along random orientations. By applying 256 of those comparisons, BRIEF represents a particular fingerprint of an image patch which can provide certain distinctiveness while remaining very robust against illumination changes. Moreover, to obtain some invariance towards spatial shifts the image is pre-smoothed with a box filter or a Gaussian filter before applying intensity comparisons. Thanks to the simplicity of the operations the BRIEF descriptor can be extracted astonishingly fast and its binary nature allows to match it efficiently with Hamming distance.

Building on the success of BRIEF, several extensions were proposed [3, 54, 92]. While BRIEF relies on random Gaussian sampling to generate a set of sampling points, BRISK [54] proposes a deterministic approach which is inspired by a wide baseline feature descriptor called DAISY [114]. The uniform sampling scheme of BRISK (see Fig. 2.7(b)) is based on a circular pattern that is rotated according to the dominant orientation detected in the feature detection step. The final descriptor is assembled by performing an exhaustive set of the short-distance intensity comparisons between the neighboring locations. Furthermore, BRISK proposes to pre-smooth sampling locations with a Gaussian kernel of standard deviation σ_i that is proportional to the distance between the points and the center of the patch. Thanks to this step, BRISK avoids aliasing effect. FREAK [3] also postulates using different Gaussian kernels to smooth sampling locations, but the size of Gaussian kernel changes in the case of FREAK with respect to the log-polar retinal pattern presented in Fig. 2.7(c).

ORB [92] is yet another extension of BRIEF which enhances the discriminative power of intensity-based binary descriptors by decorrelating the responses of intensity comparisons. By using a greedy algorithm over a set of 300k keypoints, Rublee *et al.* is able to significantly increase the variance of each bit in ORB and, hence, increase their amount of information they contain. Fig. 2.7(d) shows the initial subset of the binary tests generated by maximizing the variance for each bit and the results of the decorrelation of those tests. The rotation invariance of ORB is obtained by rotating the sampling scheme according to the dominant patch orientation, as it is done in BRISK.

Due to their efficiency, binary descriptors based on simple intensity comparisons have become widely used in many real-time applications such as Visual SLAM [109] or object detection [19]. Not only do they enable fast feature extraction, but they also are very fast to match as the Hamming distance between two binary vectors can be computed much faster than the Euclidean one, especially on modern CPU architectures where a `popcount`[1] instruction is implemented. Nevertheless, their discriminative power remains sufficient only when matching is done across reasonably sized datasets [21] and

[1]The `popcount` instruction computes the number of bits set to 1, and can be used after a bitwise XOR operation to compute the Hamming distance very efficiently.

in the case of truly large-scale applications they are still outperformed by their floating-point competitors.

2.2.4 Learning-based Methods

Although the above mentioned descriptors have been widely in Computer Vision, they still do not provide full invariance with respect to various viewpoint and illumination changes. Recent advancements in statistical methods motivated researchers to look into machine learning algorithms to improve both the efficiency and accuracy of image descriptors.

Perhaps the first attempt to use data information to form a descriptor is PCA-SIFT [50] where PCA is applied on normalized gradient patches to form feature descriptors. Other unsupervised methods include hashing-based algorithms that learn compact binary descriptors whose Hamming distance is correlated with the similarity in the original input space [38, 51, 87, 94, 131, 132]. Semantic hashing [94] trains a multi-layer neural network to learn compact representative binary codes. Spectral hashing [131] minimizes the expected Hamming distance between similar training examples, and was recently extended to optimize over example affinities [132]. Similarly, [51, 79] find codes whose Hamming distances well approximate the original Euclidean ones. In [38, 127], iterative and sequential optimization strategies that find projections with minimal quantization error are explored. While these approaches have proven highly effective for finding compact binary codes, they rely on pre-defined distance or similarity measures and in many cases are limited to the accuracy of the original input space.

Supervised learning approaches can learn feature spaces tailored to specific tasks [47, 60, 86, 111, 127]. They exploit labeled example pairs or triplets that encode the desired proximity relationships of the learned metric. In [47], a Mahalanobis distance metric is learned and optimized with respect to labeled distance constraints. Linear Discriminant Analysis is applied in [38, 111] to learn discriminative feature embeddings. Semi-supervised sequential learning algorithms are proposed in [60, 127] for finding discriminative projections. Similar to these approaches, most methods define a linear transformation of the data in either the original or a kernelized feature space and rely on a pre-specified kernel function to capture non-linearities. While they are well-suited for image categorization and indexing tasks for which task-specific kernels have been proposed, such as in [39], they are less applicable to local descriptor matching where the appropriate choice of kernel function is less well understood.

Recent descriptor learning methods have emphasized the importance of learning not only the optimal weighting, but also the optimal *shape* or pooling configuration of the underlying representation [17, 103, 104]. In [17], the authors optimize over different feature selection and pooling strategies of gradient-based features, however, the criterion considered—the area below the ROC—is not analytical making it difficult to optimize. A convex optimization strategy was developed in [103]. To make learning tractable, however, a limited set of pooling configurations was considered and restricted to circular, symmetrically arranged pooling regions centered about the patch. As shown in our experiments, our binary descriptor achieves a similar accuracy to these methods at a fraction of the matching cost. Although the method of [103] was further extended to the binary case [104] by using the quantization technique of [49], our framework still yields similar performance while requiring a much simpler optimization.

Jointly optimizing over descriptor weighting and shape poses a difficult problem due to the potentially large number of pooling configurations one might encounter. This is especially true for learning generic shapes where the number of pooling regions can easily be in the millions, even for small patch sizes. Fortunately, this is a problem for which AdaBoost [33] and other boosting methods [31, 126] are particularly well-suited. Although greedy, boosting is an effective method for constructing a highly accurate predictor from a large (potentially infinite) collection of constituent parts. The resulting *boosting-trick*, like the kernel-trick, maps the input to a high-dimensional feature space, however, the mapping it defines is explicit, with the learned embedding assumed to be sparse [25, 89]. As a result and unlike kernel methods, boosting appears to be an efficient way to find a non-linear transformation of the input that is naturally parameterized over both the descriptor shape and weighting. In Chapter 4, we will show how to use the boosting trick to learn compact and robust binary local feature descriptors.

Part I

Learning Binary Descriptors

CHAPTER
THREE

EFFICIENT DISCRIMINATIVE PROJECTIONS FOR BINARY DESCRIPTORS

In this chapter, we aim to bridge the performance gap between the recently proposed binary descriptors, such as BRIEF, BRISK and ORB and a more robust floating-point descriptors such as SIFT or SURF. In order to improve the recognition performances without increasing the computational cost, we adopt a discriminative approach as in [17, 18, 111]: We use training data to learn linear projections that map image patches to a more discriminative subspace, and to obtain a binary descriptor, we threshold the projected patches. This way we avoid the intermediate step of computing complex floating-point descriptors, which is common for many binary descriptors [111, 116] but exceeds the capabilities of many mobile platforms.

Nevertheless, projecting image patches is computationally expensive and negates the efficiency of binary descriptors, especially when they have to be computed in real time. Thus, in our approach, and this is our main contribution, we train the projections not only to be discriminative but also to be computed as a linear combination of a small number of simple filters from a given dictionary, as shown in Fig. 3.1. We design the dictionaries in such a way that the filter responses can be computed fast, with box or Gaussian filtering or using integral images. Our key idea is that this can be done by imposing sparsity constraints and using efficient optimization techniques.

To summarize, we build our binary descriptor, which we refer to as D-Brief for Discriminative BRIEF, by first projecting image patches to a more discriminant subspace and then concatenating the results of a thresholding operation applied on the projected

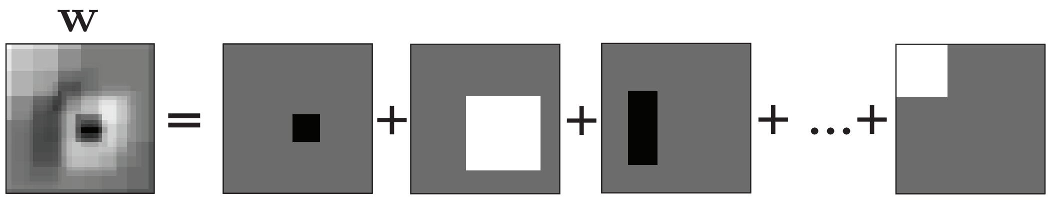


Figure 3.1: To compute our binary descriptor, we learn from a training set of corresponding image patches several discriminative linear projections that can be computed from a linear combination of a few simple filters. For the example of this figure, we used rectangular filters that can be computed efficiently with integral images, but we also consider box and Gaussian filters which are also efficient to compute. This approach enables us to build our binary descriptor fast while leveraging on training data.

coordinates. When we use box or Gaussian filters to construct the projections, we can speed up the computation of projected patches by first convolving the entire image with a given filter and then combining only a few values read from the convolved image. With rectangular filters, we use integral images to compute the responses fast. As a result, D-Brief provides better recognition performances than its direct competitors [21, 54, 92], while being significantly shorter—only 32 bits to be compared with several hundreds—and less time consuming. D-Brief can also be seen as a much more efficient binary alternative to the short floating-point descriptors of [17], which require more time to be computed, and hence target different applications than binary descriptors. Although our approach uses LDA, as done in [111], we also show that the linear projections can be optimized for very fast computation using recent optimization theory, and hence prove that our method is not limited to LDA and can encompass other projection-based techniques.

The rest of this chapter is organized as follows. In Section 3.1, we introduce our method to construct a binary descriptor from a set of discriminant projections learnt from a training dataset. We explain in details how we optimize on the projections so that they can be computed using small number of simple filter banks which enables efficient computation of projected patches. We compare the performance of our D-Brief binary descriptor with the state of the art in Chapter 5.

3.1 Method

Our binary descriptor is computed by applying a set of projections to a real-valued vector made of the intensities of an image patch and then thresholding the results:

$$\forall_{i \in 1,\ldots,N} \quad b_i = \operatorname{sign}(\mathbf{w}_i^\top \mathbf{x} + \tau_i) \ , \tag{3.1}$$

where the b_i are the N bits of our descriptor, the $\mathbf{w}_i$ the projections, the τ_i the thresholds, and $\mathbf{x}$ the image patch in vector form. We first show how to optimize over the $\{\mathbf{w}_i, \tau_i\}$ to obtain our efficient and discriminative descriptor, and then we explain how it can be computed fast. Finally, we evaluate how our descriptor is influenced by its parameters.

In principal, our approach seeks to minimize the expected Hamming distance between binary descriptors that describe similar keypoints while maximizing it for the descriptors that describe different keypoints. To that end, we learn a set of discriminative orthogonal linear projections and the corresponding thresholds from a set $\mathcal{P}$ of pairs of corresponding image patches and another set $\mathcal{N}$ of pairs of different patches. Nevertheless, applying general projections directly on the image patches is computationally expensive. Hence, our key idea is to train the projections $\mathbf{w}_i$ to be a linear combination of a few elements from a predefined set or *dictionary* D, which is designed to contain elements for which the responses can be computed fast, for example using box filters.

More formally, we express the projections as $\mathbf{w}_i = D\mathbf{s}_i$, where the dictionary D is defined as a matrix with its columns being the elements of the dictionary. We want most of the coefficients of the $\mathbf{s}_i$ vectors to be equal to zero, that is, the $\mathbf{s}_i$ should be sparse. Our goal can then be formalized as solving the following minimization problem:

$$\begin{aligned}\min_{\{(\mathbf{s}_i,\tau_i)\}} \sum_{i\in 1,\ldots,N} \quad & \sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}} \operatorname{sign}((D\mathbf{s}_i)^\top \mathbf{x} + \tau_i)\operatorname{sign}((D\mathbf{s}_i)^\top \mathbf{x}' + \tau_i) - \\ & \sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}} \operatorname{sign}((D\mathbf{s}_i)^\top \mathbf{x} + \tau_i)\operatorname{sign}((D\mathbf{s}_i)^\top \mathbf{x}' + \tau_i) + \lambda|\mathbf{s}_i|_1 \\ & \text{subject to} \quad (D\mathbf{s}_i)^\top(D\mathbf{s}_j) = \delta_{ij} \ ,\end{aligned} \tag{3.2}$$

where the last term encourages sparsity of the $\mathbf{s}_i$ with λ determining its sparsity level, and $|.|_1$ denotes the ℓ_1 norm. The products of sign functions promote the desired

Hamming distances between the patch pairs $(\mathbf{x}, \mathbf{x}')$ from the $\mathcal{P}$ and $\mathcal{N}$ sets. The constraint, with $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, makes sure that the projections are orthogonal, which reduces the redundancy across different dimensions.

Unfortunately, direct minimization of this objective function is difficult as it involves the non-differentiable sign function. In our case, typical solutions of this problem, such as smooth approximation with the hinge function [13], would lead to a quadratic non-convex problem which is challenging to solve, as it involves thousands of unknowns.

Thus, we drop the sign function and minimize the related objective function as it is done in [111]:

$$\begin{aligned} \min_{\{\mathbf{s}_i\}} \quad \sum_i \quad & \frac{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}} ((D\mathbf{s}_i)^\top(\mathbf{x}-\mathbf{x}'))^2}{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}} ((D\mathbf{s}_i)^\top(\mathbf{x}-\mathbf{x}'))^2} + \lambda|\mathbf{s}_i|_1 \\ & \text{subject to} \quad (D\mathbf{s}_i)^\top(D\mathbf{s}_j) = \delta_{ij} \end{aligned} \tag{3.3}$$

The above objective is independent of the thresholds τ_i. Hence, after finding the projections $\mathbf{w}_i = D\mathbf{s}_i$, the optimal thresholds are obtained by minimizing the original objective of Eq. (3.2) using the training sets $\mathcal{P}$ and $\mathcal{N}$. With the projections $\mathbf{w}_i$ being fixed, this requires simple one-dimensional search, as explained in [111].

A possible method to solve the minimization problem of Eq. (3.3) is by using Stochastic Gradient Descent, with soft-thresholding as the proximal operator of the ℓ_1 norm [9]. However, even after dropping the sign function, Eq. (3.3) remains non-convex and the optimization is likely to get stuck in a local minimum. Therefore, it becomes essential to initialize the optimization properly, because random initialization may not give satisfactory results, as we show below.

We propose to set the initialization point of the optimization using the following approach: We start by minimizing the first term of Eq. (3.3) and we obtain an initial set of discriminant projections $\{\mathbf{w}_i^0\}$ using Linear Discriminant Embedding (LDE) [17]:

$$\begin{aligned} \{\mathbf{w}_i^0\} \quad = \quad & \arg\min_{\{\mathbf{w}_i\}} \quad \sum_i \frac{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}} ({\mathbf{w}_i}^\top(\mathbf{x}-\mathbf{x}'))^2}{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}} (\mathbf{w}_i^\top(\mathbf{x}-\mathbf{x}'))^2} \\ & \text{subject to} \quad (\mathbf{w}_i)^\top(\mathbf{w}_j) = \delta_{ij} \ . \end{aligned} \tag{3.4}$$

As one can see, the LDE includes the orthogonality constraint from Eq. (3.3), as the resulting projections are based on the orthogonal eigenvectors. We then address the sparsity constraint of Eq. (3.3) and approximate each $\mathbf{w}_i^0$ projection with a sparse linear combination of elements from dictionary D by minimizing the following objective:

$$\{\mathbf{s}_i^0\} = \underset{\mathbf{s}_i}{\arg\min} \quad \|\mathbf{w}_i^0 - D\mathbf{s}_i\|_2^2 + \lambda |\mathbf{s}_i|_1 \,, \tag{3.5}$$

where the first term corresponds to the quality of approximation, and the second one to the sparsity of the filter representation. To simplify our optimization, we do not constrain our approximated projections to be orthogonal, although this could certainly lead to an interesting extension of our approach.

The advantage of the stepwise approach discussed above is that Eq. (3.5) can be solved in closed-form as shown in [17], while Eq. (3.5) is convex and can be solved with efficient recent techniques [9]. In practice, we use the MATLAB `lasso` function which implements [113] and lets the user define $|\mathbf{s}|_0^{\max}$, the maximal number of non-zero coefficients in the representation, which is a more convenient way of controlling the sparsity of the approximation compared to tuning λ.

To evaluate the quality of our approach, we performed the following experiments. We took two sets of projections: Random ones $\{\mathbf{w}_i^{\mathbf{R}}\}$ and those obtained using our stepwise approach $\{\mathbf{w}_i^{\mathbf{S}} = D\mathbf{s}_i^0\}$. We then minimized Eq. (3.3) with Stochastic Gradient Descent on a set of datasets presented in Section 5.1 using first $\{\mathbf{w}_i^{\mathbf{R}}\}$ and then $\{\mathbf{w}_i^{\mathbf{S}}\}$ as initializers. As a result we obtained two optimized sets of projections, $\{\mathbf{w}_i^{\mathbf{R-Opt}}\}$ and $\{\mathbf{w}_i^{\mathbf{S-Opt}}\}$ respectively. To make the comparison fair, we set the number of projections to 32 and tuned the parameters so that each projection was a linear combination of 64 columns of the BOX dictionary D, which we discuss in details in the next section. We then found the corresponding optimal thresholds and evaluated the resulting descriptors on all the test sets, using the setup of Section 3.3. In Fig. 3.2 we present two representative ROC curves and in Table 3.1 summary of the other results.

Optimizing over the random projections significantly improves the results. Nevertheless, the projections obtained using the stepwise initialization scheme we propose perform better even without optimization. Moreover, our evaluation shows that applying a global optimization on the $\{\mathbf{w}_i^{\mathbf{S}}\}$ does not lead to any significant improvement and, hence, in the remainder of this chapter we use our stepwise approach without further optimizing it.

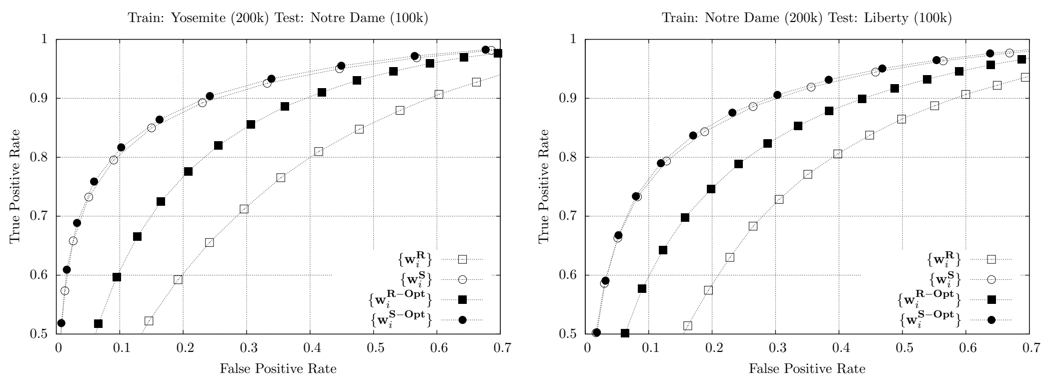


Figure 3.2: Results obtained using Stochastic Gradient Descent applied to Eq. (3.3) and initialized with random projections or projections found with our stepwise approach. We used 32 projections and tuned the parameters so that each projection is a linear combination of 64 columns of BOX dictionary. We then found the corresponding optimal thresholds. Initializing gradient descent with the stepwise approach instead of random projections boosts the results significantly. Interestingly, optimization improves the quality of the projections only slightly while requiring additional processing time. Thus, to build D-Brief we use projections computed with the stepwise approach without further optimization.

Train	Test	$\{\mathbf{w}_i^{\mathbf{R}}\}$	$\{\mathbf{w}_i^{\mathbf{R-Opt}}\}$	$\{\mathbf{w}_i^{\mathbf{S}}\}$	$\{\mathbf{w}_i^{\mathbf{S-Opt}}\}$
Yosemite	Notre Dame	69.28	54.97	44.52	41.96
Yosemite	Liberty	71.54	63.08	52.01	48.81
Notre Dame	Yosemite	76.31	62.22	46.72	47.06
Notre Dame	Liberty	70.48	60.33	48.86	46.72
Liberty	Notre Dame	75.07	52.63	43.35	38.81
Liberty	Yosemite	77.59	66.14	49.08	50.26

Table 3.1: 95% error rates for different training and testing configurations obtained with different sets of projections: random projections $\{\mathbf{w}_i^{\mathbf{R}}\}$, projections computed using our stepwise approach $\{\mathbf{w}_i^{\mathbf{S}}\}$, random projections optimized using Stochastic Gradient Descent $\{\mathbf{w}_i^{\mathbf{R-Opt}}\}$ and projections obtained with our stepwise approach optimized using Stochastic Gradient Descent. After learning the projections, we computed the optimal thresholds. To make the comparison fair, all the sets contain 32 projections and each projection is a linear combination of 64 columns of the BOX dictionary.

3.2 Dictionaries

After finding a set of projections $\{\mathbf{w}_i = D\mathbf{s}_i\}$, they can be applied on an image patch $\mathbf{x}$ as:

$$\mathbf{w}_i^\top \mathbf{x} = (D\mathbf{s}_i)^\top \mathbf{x} = \sum_{j \text{ such that } \mathbf{s}_{ij} \neq 0} \mathbf{s}_{ij} D_j^\top \mathbf{x} \,, \tag{3.6}$$

where the D_j are the columns of matrix D and contain the dictionary elements. The dictionary in D is designed so that the dot product $D_j^\top \mathbf{x}$ can be computed efficiently. In our experiments we use three different dictionaries that contain:

a) **Box filters (BOX)**: To create this dictionary we generate a set of box filters of size 5×5 that are centered at each coordinate of the image patch. Since our subsampled patches are of size 32×32, there are 1,024 elements in this dictionary.

b) **Gaussian filters (GAUSS)**: Similarly, we generate a set of Gaussian filters with $\sigma = 3$ centered at each coordinate of the image patch. The size of this dictionary is also 1,024.

c) **Rectangular filters (RECT)**: We create this dictionary by generating a set of rectangular filters of different sizes centered at each coordinate. We subsample the space of all possible rectangular filters by considering those whose horizontal or vertical edge is equal to $1, 4, 7, 10, \ldots$. The resulting dictionary size is 34,596.

At run-time, to compute the $D_j^\top \mathbf{x}$ values, we first convolve the image patch with a box filter or a Gaussian filter, or compute the integral image for the patch. All these operations can be done very efficiently. Then, the $D_j^\top \mathbf{x}$ can be obtained by reading a single value in the result of the convolution, or four values in the integral image in the case of the RECT dictionary. An extensive comparison of different types of dictionaries and their influence on the quality of the final descriptor is presented in the next section.

Although scale- and rotation-invariance is not embedded in the above formulation of our descriptor, it can be achieved by rectifying the patch according to the detection results. To still benefit from the pre-computation of the filter responses, a Gaussian scale pyramid can be generated and the dictionaries can be applied to the images at each level of the pyramid.

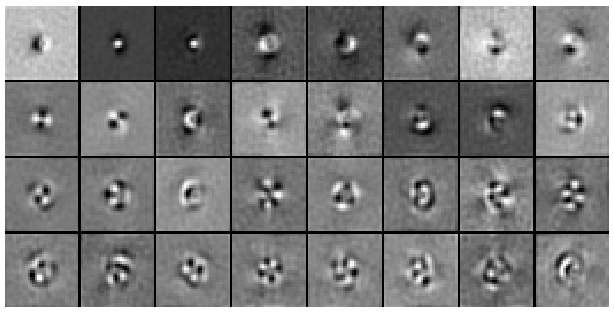
3.3 Experiments

To train our projections, we use three publicly available datasets: Liberty, Notre Dame and Yosemite [17] presented in Section 5.1. To avoid overfitting when training the LDE projections we apply a regularization method proposed in [17] with clipping parameter $\alpha = 0.01$. We tried different combinations of training and testing datasets, but as in [17], we found that choosing a specific combination does not have a strong influence on the final results. A set of sample projections learned using Liberty,Notre Dame and Yosemite datasets datasets can be seen in Fig. 3.3.

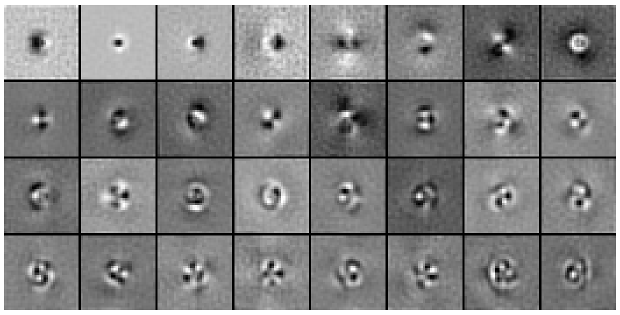
As shown in Fig. 3.4 the best performances are obtained when using the first 32 projections. When a larger number of projections is used, the performances start deteriorating. This performance peak can be explained by the fact that the binarization step gives equal importance to all the projections, while the projections computed with LDE are ranked by decreasing discriminative power. Furthermore, the results show that the binarization step improves the overall quality of the final descriptor. This can be explained by the importance of highly non-linear sign function applied on the projected coordinates and a strong classification decision it entails. Moreover, using the least ranked projections, which correspond to the dimensions of lower energy, introduces classification noise and deteriorates performance. We therefore keep the top 32 projections in the rest of the chapter, and our descriptor is made of 32 bits.

We performed a set of experiments to qualitatively and quantitatively assess the influence of different parameters on Eq. (3.5). Fig. 3.5 shows how the number of elements used to reconstruct the Gaussian and Difference-of-Gaussian projections influences the final approximation results. Fig. 3.6 shows how the dictionary type and the number of elements used influences the results on two sample LDE projections.

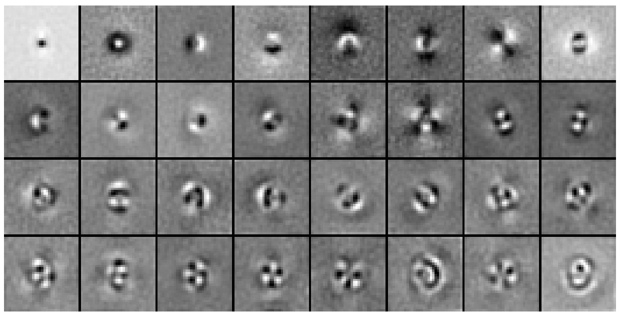
Fig. 3.7 shows the recognition rates for projections obtained by optimizing Eq. (3.5), from the top 32 LDE projections obtained with Eq. (3.5) with different dictionaries and various numbers of dictionary elements used. Before binarization, the RECT dictionary provides the best results, followed by the GAUSS and BOX dictionaries. While optimizing the projections with Eq. (3.5) hurts the recognition performances before binarization, it is not true anymore after binarization. There is actually a minor improvement, which may come from the fact that the LDE projections are typically noisy



(a) Liberty



(b) Notre Dame



(c) Yosemite

Figure 3.3: The top 32 projections learnt from 200k pairs of non-normalized patches from the Liberty, Notre Dame and Yosemite datasets using Local Discriminant Embedding.

and the optimization introduces some regularization which makes the thresholds generalize better. Surprisingly, the GAUSS dictionary performs better than the others after binarization for small numbers of elements, but then gets outperformed again by the RECT dictionary for large numbers of elements.

As explained in Section 3.1, to build the descriptors efficiently at run-time we first

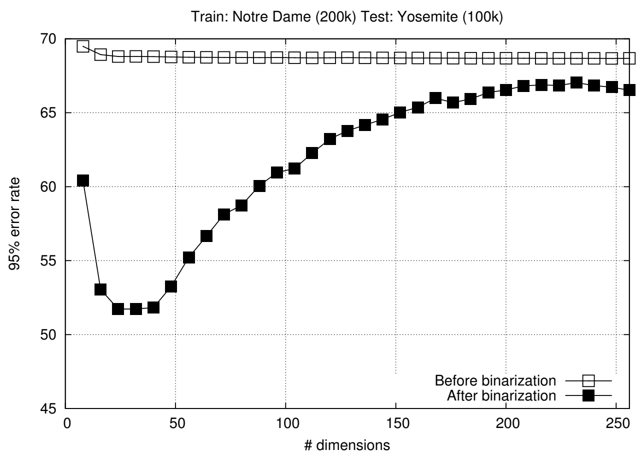


Figure 3.4: 95% error rates for different numbers of LDE projections obtained from Eq. (3.5) and used for dimensionality reduction before and after applying the thresholds (the 95% error rate is the percent of incorrect matches obtained when 95% of the true matches are found). Without binarization, the error rates do not change significantly for different dimensionalities. After binarization, however, the dimensionality has much greater influence, likely because the projections are ranked by decreasing discriminative power, while the binarization gives them equal importance. The minimum of the plotted curve corresponding to the best performance is obtained for 32 dimensions.

preprocess the image patch either by convolving it with a box or Gaussian filter or by computing the integral image. Then, the $D_j^\top \mathbf{x}$ values of Eq. (3.6) are obtained by reading either one or four values from the output of this preprocessing stage. Convolution with a box filter is faster than convolution with a Gaussian filter, and when using the RECT dictionary, we need to read four values for each $D_j^\top \mathbf{x}$ instead of only one as with the BOX and GAUSS dictionaries. The computation times are therefore different for each dictionary and Table 5.4 presents the times required for computing our descriptors using different dictionaries. With our approach, we can speed up the description time with respect to regular projections by over an order of magnitude.

Moreover, there is a trade-off between the accuracy and efficiency of the dictionaries: The slower one yields the best recognition performance, and *vice versa*. In practice, this means that we can adapt our method to the need of the final application.

As reported in [17], pre-normalizing the patches by subtracting the intensity mean value and dividing by the standard deviation and post-normalizing the descriptor to unit length improves the performance, in case of a real-valued descriptor. Inspired by

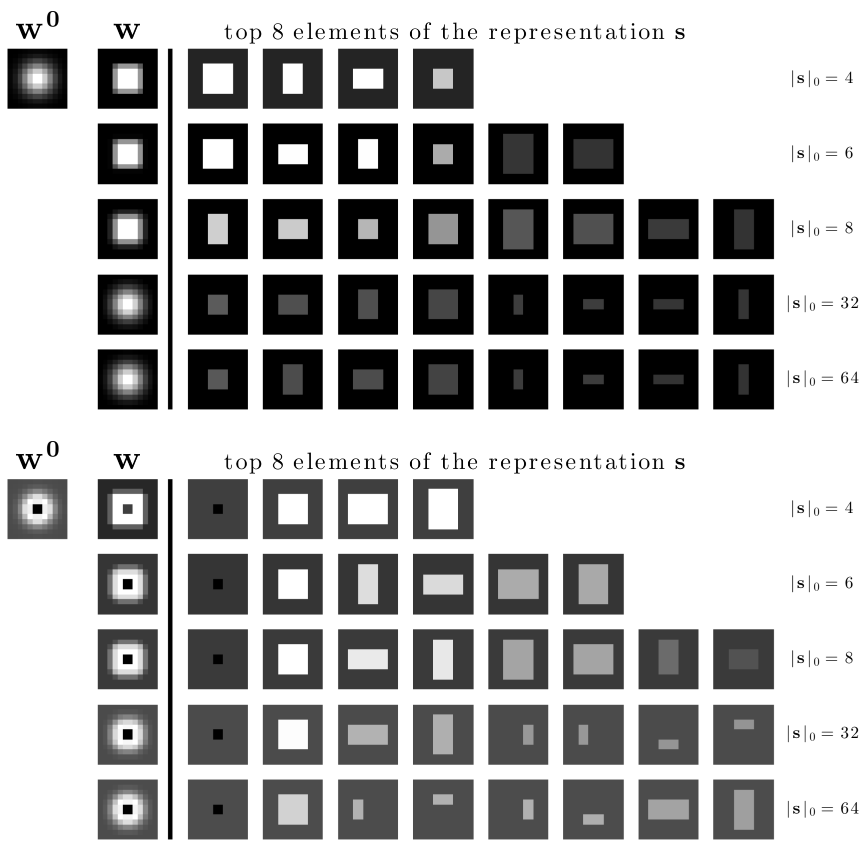


Figure 3.5: Approximations of Gaussian and Difference-of-Gaussian kernels. We used RECT dictionary and varied the maximal numbers of dictionary elements per projection. The columns on the right show the first 8 elements of the sparse representation $\mathbf{s}$

these findings, we tried normalizing our patches and descriptors as in [17]. Fig. 3.8 confirms the observed performance improvement for real-valued coordinates. However, these steps do not have much influence on the results after binarization. This is because we train the thresholds to be discriminative after applying the projections and, hence, they adapt well to the light variations. Hence, we skip the normalization, as it entails additional computational overhead.

3.4 Conclusion

In this chapter, we presented a new method to learn discriminative projections which can be computed efficiently as a linear combination of a few simple filters from a given

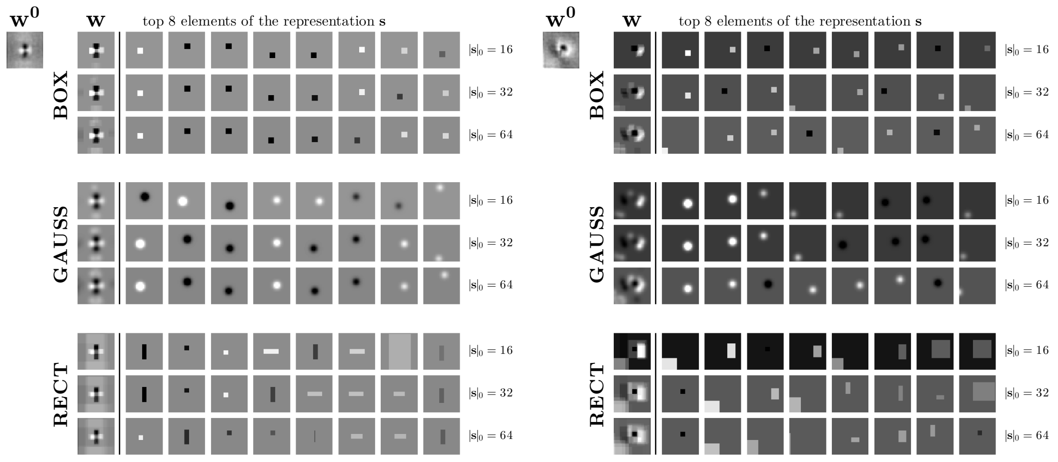
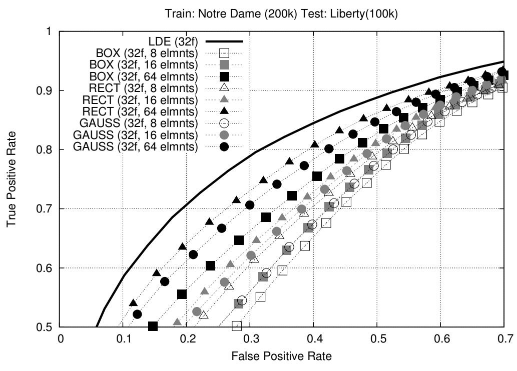
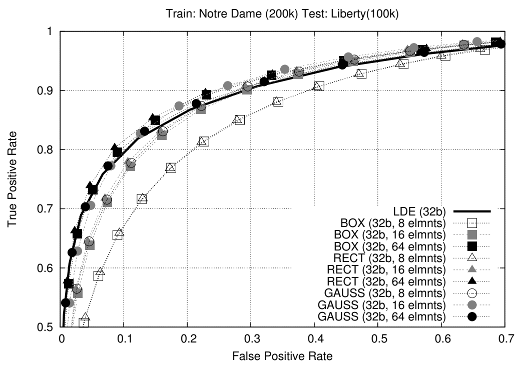


Figure 3.6: Projections obtained by optimizing Eq. (3.5), from two LDE projections obtained with Eq. (3.5). We varied the dictionaries and the maximal numbers of dictionary elements per projection. The columns on the right show the first 8 elements of the sparse representation **s**.

dictionary. This approach enables us to learn a compact and discriminative binary descriptor we call D-Brief. The method proposed in this chapter is not only limited to LDA projections, as one can imagine enabling efficient computation of many filter responses or projections, such as wavelets or PCA-generated projections, by approximating them with our approach. We compare the performances of D-Brief with respect to the state-of-the-art descriptors in Chapter 5.

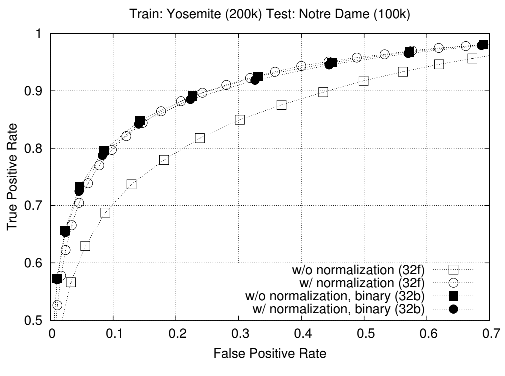


(a) Before binarization



(b) After binarization

Figure 3.7: Performances of the projections obtained by optimizing Eq. (3.5), from the top 32 LDE projections obtained with Eq. (3.5). We varied the dictionaries and the maximal numbers of dictionary elements per projection. We also give the results for the projections directly obtained with Eq. (3.5), referred to as LDE. In parentheses: Number of floating point (f) or binary (b) coordinates, and number of elements. Before binarization, the RECT dictionary provides the best approximation, followed by the GAUSS and BOX dictionary. While the approximation hurts the recognition performances before binarization, it is not true anymore after binarization. The minor improvement can be explained by the smoothing effect of the approximation applied on typically noisy LDE projections which makes the thresholds generalize better.



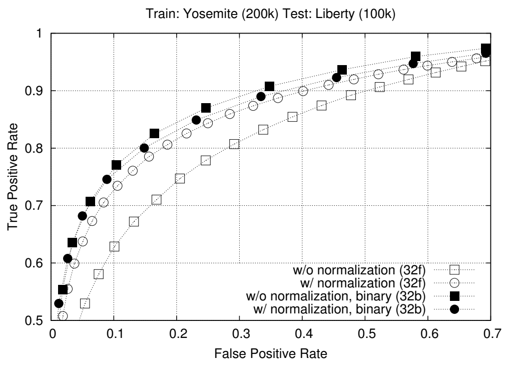


Figure 3.8: Influence of normalization applied on the D-Brief descriptors before (left) and after (right) binarization. In parentheses: Number of floating point (f) or binary (b) coordinates. Normalization improves the performance only for the real-valued coordinates. The thresholds applied in the binarization step adapt to the discriminative subspace and, we can hence skip the normalization steps with no loss of accuracy.

CHAPTER
FOUR

LEARNING DESCRIPTORS WITH BOOSTING

In this chapter, we propose another method to learn a compact binary local feature descriptor using machine learning technique called boosting. Although the binary descriptor presented in the previous chapter can be computed fastly with extremely low memory footprint, its performance can still be improved. Hence, we investigate here a more powerful boosting-inspired method of learning feature descriptors that is capable of modeling the non-linear nature of many unwanted image transformations.

Learning an invariant feature representation can be seen as finding an appropriate similarity measure which remains invariant to unwanted image transformations. Although several learning methods have been proposed in the literature [47, 98, 99], they have largely focused on finding a linear feature mapping in either the original input or a kernelized feature space. As a result, modeling non-linearities requires choosing an appropriate kernel function that maps the input features to a high-dimensional feature space where the transformations are assumed to be linear. However, selecting the right kernel, which is a crucial element of the algorithm, is often non-intuitive and generally constitutes a complex and challenging problem. Recent works introduce non-linear mappings by the means on sigmoidal activation functions in multi-layer neural network architectures [66, 67], however, they typically rely on existing representations such as pre-computed descriptors.

In this chapter, we propose a novel supervised learning framework that finds low-dimensional but highly discriminative descriptors. With our approach, image patch appearance is modeled using local non-linear filters that are selected with boosting.

Our work is inspired by Boosted Similarity Sensitive Coding (SSC) [98] which is the first application of boosting to learn an image similarity measure and was later extended in [116] to be used with a Hamming distance. We show how we can use Boosted SCC as a way to efficiently select features, from which we compute a compact representation. Analogous to the kernel-trick, our approach can be seen as applying a *boosting-trick* [25] to obtain a non-linear mapping of the input to a high-dimensional feature space. Unlike kernel methods, boosting allows for the definition of intuitive non-linear feature mappings that can share a close connection with existing, prevalent feature descriptors. Our learning approach is not limited to any pre-defined sampling pattern and provides a more general framework than previous training-based methods [17, 92, 103]. It also scales linearly with the number of training examples, making it more amenable to large scale problems, and results in highly accurate descriptor matching.

Nevertheless, as image databases grow in size, modern solutions to local feature-based image indexing and matching must not only be accurate but also highly efficient to remain viable. Binary descriptors are of particular interest as they require far less storage capacity and offer much faster matching times than conventional floating-point descriptors [21, 38, 54, 92, 111], or even quantized descriptors [17]. In addition, they can be used directly in hash table techniques for efficient Nearest Neighbor search [64, 80], and their similarity can be computed very quickly on modern CPUs based on the Hamming distance.

However, as our experiments show, state-of-the-art binary descriptors often perform worse than their floating-point competitors: some are built on top of existing representations such as SIFT or GIST by relying on training data [38, 111], and are therefore limited by the performance of the intermediate representation. Others start from raw image intensity patches, but focus on computation speed and rely on fast-to-compute image features [21, 54, 92], which limit their accuracy.

To address these shortcomings, we extend our learning framework to the binary case and we train a highly discriminative yet compact binary descriptor. This extension demonstrates the real advantage of our approach as it enables us to not only compress the descriptor, but also significantly decrease the processing cost and memory footprint. For each dimension of the resulting binary representation, we learn a hash function of the same form as an AdaBoost strong classifier, that is the sign of a linear combination of non-linear weak learners. The resulting binary descriptor, which we refer to as BinBoost, significantly outperforms its binary competitors and exhibits a similar accuracy to state-of-the-art floating-point or quantized descriptors at a fraction of the storage and matching cost. Furthermore, it is more complex to optimize, and we show how to efficiently optimize our hash functions using boosting.

Our method provides a general descriptor learning framework that encompasses, among many others, the state-of-the-art intensity-based [21, 54, 92] and gradient-based descriptors [10, 62, 111]. Our results show that the ability to effectively optimize over the descriptor filter configuration leads to a significant performance boost at no additional computational cost compared with the original hand-designed representation. Moreover, our approach is not restricted to local feature descriptors and we show that it can be applied to other types of image data, such as face images.

The rest of this chapter is organized as follows. In Section 4.1 we describe our method: we first show how to efficiently construct our set of weak learners, from which we compute a compact floating-point representation. We then explain how to extend this approach to build a binary local feature descriptor. In Section 4.2 we discuss different weak learner types and in Section 4.3 we describe the experimental setup of our method and its parameters. Finally, in Section 4.4, we showcase the application of our learning method on a set of face images.

4.1 Method

In this section we describe methods for learning local feature descriptors with boosting. We first formulate our problem by defining the exponential loss objective function we use to learn a similarity embedding between image patches. We then present different similarity measures which, when plugged into our boosting framework, can be used to train floating-point and binary descriptors.

4.1.1 Problem formulation

Given an image intensity patch $\mathbf{x}$, we look for a descriptor $C(\mathbf{x}) = [C_1(\mathbf{x}), \dots, C_D(\mathbf{x})]$ which maps the patch to a D-dimensional vector. This descriptor can be learned by minimizing the exponential loss with respect to a desired similarity function $f\left(C(\mathbf{x}), C(\mathbf{y})\right) =$

$f_C(\mathbf{x}, \mathbf{y})$ defined over image patch pairs:

$$\mathcal{L} = \sum_{i=1}^{N} \exp(-l_i f_C(\mathbf{x}_i, \mathbf{y}_i)) \tag{4.1}$$

where $\mathbf{x}_i, \mathbf{y}_i \in \mathcal{R}^p$ are training intensity patches and $l_i \in \{-1, 1\}$ is a label indicating whether it is a similar $(+1)$ or dissimilar (-1) pair. Minimizing Equation (4.1) finds an embedding which maximizes the similarity between pairs of similar patches, while minimizing it for pairs of different patches.

This formulation allows for numerous similarity functions f_C. We consider similarity functions of the form

$$f_C(\mathbf{x}, \mathbf{y}) = C(\mathbf{x})^T \, \mathbf{A} \, C(\mathbf{y}) \tag{4.2}$$

where $\mathbf{A} \in \mathcal{R}^{D \times D}$ is a symmetric matrix. This defines a general class of symmetric similarity measures that can be factorized to compute a feature descriptor independently over each input and used to define a wide variety of image descriptors. In what follows, we consider different choices of $\mathbf{A}$ and $C(\cdot)$ ordering them in increasing complexity.

4.1.2 Boosted Similarity Sensitive Coding (SSC)

The Boosted SSC method proposed in [98] considers a similarity function defined by a simply weighted sum of thresholded response functions $\{h_d(\cdot)\}_{d=1}^{D}$:

$$f_{SSC}(\mathbf{x}, \mathbf{y}) = \sum_{d=1}^{D} \alpha_d h_d(\mathbf{x}) h_d(\mathbf{y}) \, . \tag{4.3}$$

This function is the weighted Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ and corresponds to Equation (4.2) where $\mathbf{A}$ is restricted to be a diagonal matrix. The importance of each dimension d given by the α_d's and the resulting D-dimensional descriptor is a floating-point vector $C(\mathbf{x}) = [\sqrt{\alpha_d} h_d(\mathbf{x})]_{d=1}^{D}$, where α is constrained to be positive.

Substituting f_{SSC} for f_C in Equation (4.1) gives

$$\mathcal{L}_{SSC} = \sum_{i=1}^{N} \exp\left(-l_i \sum_{d=1}^{D} \alpha_d h_d(\mathbf{x}_i) h_d(\mathbf{y}_i) \right) \, . \tag{4.4}$$

In practice the space of h's is prohibitively large, possibly infinite, making the explicit optimization of $\mathcal{L}_{SSC}$ difficult, however, this constitutes a problem for which boosting is particularly well suited [33]. Although boosting is a greedy optimization scheme, it is an effective method for constructing a highly accurate predictor from a collection of weak predictors h. Due to its greedy nature, however, the weak learners found using Boosted SSC often remain highly redundant and hence inefficient. In what follows, we modify the similarity function $f_C(\mathbf{x}, \mathbf{y})$ so that it becomes better suited for learning low-dimensional, discriminative embeddings with boosting.

4.1.3 FPBoost

To mitigate the potentially redundant embeddings found by boosting we propose an alternative similarity measure f_{FP} that models the correlation between weak response functions:

$$f_{FP}(\mathbf{x}, \mathbf{y}) = \sum_{k,k'} \alpha_{k,k'} h_k(\mathbf{x}) h_{k'}(\mathbf{y}) = \mathbf{h}(\mathbf{x})^T \mathbf{A} \mathbf{h}(\mathbf{y}), \tag{4.5}$$

where $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \cdots, h_K(\mathbf{x})]$ and $\mathbf{A}$ is an $K \times K$ matrix of coefficients $\alpha_{k,k'}$. This similarity measure is a generalization of Equation (4.3). In particular, f_{FP} is equivalent to the Boosted SSC similarity measure in the restricted case of a diagonal $\mathbf{A}$.

Substituting the above expression into Equation (4.1) gives

$$\mathcal{L}_{FP} = \sum_{i=1}^{N} \exp\left(-l_i \sum_{k,k'} \alpha_{k,k'} h_k(\mathbf{x}_i) h_{k'}(\mathbf{y}_i) \right) . \tag{4.6}$$

We optimize $\mathcal{L}_{FP}$ using a two step learning strategy whereby we first apply AdaBoost to find weak learners $\{h_k\}_{k=1}^{K}$ by minimizing Equation (4.4) on the training samples. As shown by our experiments, this provides an effective way to select relevant h_k's. We then apply stochastic gradient descent to find an optimal weighting over the selected features that minimizes $\mathcal{L}_{FP}$. To guarantee that the similarity function f_{FP} remains symmetric, we restrict $\mathbf{A}$ to be symmetric.

The similarity function of Equation (4.5) defines an implicit feature mapping over example pairs. In order to compute the feature descriptors independently over each input, we need to factorize $\mathbf{A}$. As we constrain $\mathbf{A}$ to be symmetric, we can factorize it into the following form:

$$\mathbf{A} = \mathbf{B}\mathbf{W}\mathbf{B}^T = \sum_{k=1}^{K} w_k \mathbf{b}_k \mathbf{b}_k^T \tag{4.7}$$

where $\mathbf{W} = \mathrm{diag}([w_1, \cdots, w_K])$, $w_k \in \{-1, 1\}$, $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_k]$, $\mathbf{b} \in \mathcal{R}^K$. This factorization can be obtained after the eigendecomposition of $\mathbf{A}$ by simply premultiplying the eigenvectors by the squared root of the corresponding eigenvalues while storing the signs of the eigenvalues in $\mathbf{W}$.

Equation (4.5) can then be re-expressed as

$$f_{FP}(\mathbf{x},\mathbf{y}) = \sum_{d=1}^{D} w_d \left(\sum_{k=1}^{K} b_{d,k} h_k(\mathbf{x}) \right) \left(\sum_{k=1}^{K} b_{d,k} h_k(\mathbf{y}) \right) . \tag{4.8}$$

For $D < K$ (i.e., the effective rank of $\mathbf{A}$ is $D < K$) the factorization represents a smoothed version of $\mathbf{A}$ discarding the low-energy dimensions that typically correlate with noise, and in practice leading to further performance improvements. The factorization of Equation (4.8) defines a signed inner product between the embedded feature vectors and provides increased efficiency with respect to the original similarity measure[1]. Moreover, given that the first D eigenvalues of the $\mathbf{A}$ matrix are typically positive, we can drop the $\mathbf{W}$ matrix during the inner product computation and use the simple inner product instead. Under the assumption that the descriptors have comparable magnitudes, it is easy to show that the inner product is equivalent to the Euclidean distance. As shown in Fig. 4.1, this is the case in practice and, hence, we can leverage the existing methods for fast approximate nearest neighbor search which rely on Euclidean distances.

The final embedding $C(\mathbf{x}) = \mathbf{B}^T\mathbf{h}(\mathbf{x})$ results in a D-dimensional floating-point descriptor based on K weak learners that we call FPBoost$_K$-D. The projection matrix $\mathbf{B}$ defines a discriminative dimensionality reduction optimized with respect to the exponential loss objective of Equation (4.6). As seen in our experiments, in the case of redundant weak learners this results in a considerable feature compression, and therefore offering a more compact description than the original input patch. Although compact and highly discriminant, the descriptors learned using FPBoost are real-valued and, hence, can be slow to match and costly to store. Next, we consider a modification to FPBoost that as we show can be used to learn a highly accurate and efficient binary descriptor.

[1]Matching two sets of descriptors each of size N is $\mathcal{O}(N^2K^2)$ under the original measure and $\mathcal{O}(2NKD + N^2D)$ provided the factorization, resulting in significant savings for reasonably sized N and K, and $D \ll K$.

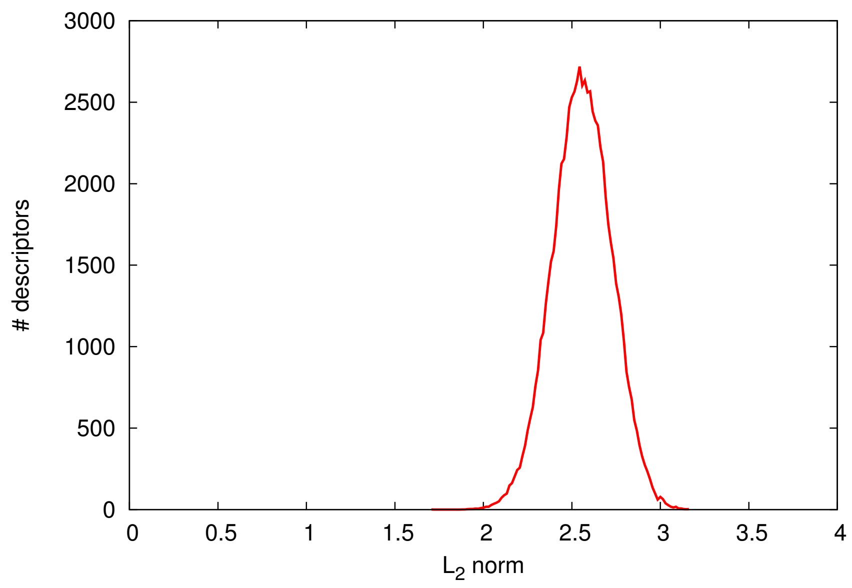


Figure 4.1: Histogram of the L_2 norms of 75k FPBoost descriptors extracted from the images of Mikolajczyk dataset [71]. The L_2 norms are upper-bounded by $\sqrt{\sum_{d=1}^{D} |\mathbf{b}_d|_1^2}$ which equals 4 in this case. A significant portion of the descriptors have a comparable magnitude and, hence, we can also use Euclidean distance in place of the equivalent inner product to measure descriptor similarity.

4.1.4 BinBoost

To learn a binary descriptor we propose a modified similarity measure that extends f_{FP} to operate within a D-dimensional Hamming space:

$$\begin{aligned} f_B(\mathbf{x}, \mathbf{y}) &= \sum_{d=1}^{D} \operatorname{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x})\right) \operatorname{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{y})\right) \\ &= \sum_{d=1}^{D} C_d(\mathbf{x}) C_d(\mathbf{y}) \end{aligned} \tag{4.9}$$

where $C_d(\mathbf{x}) = \operatorname{sgn}\left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x})\right)$ and $\mathbf{h}_d(\mathbf{x}) = [h_{d,1}(\mathbf{x}) \dots h_{d,K}(\mathbf{x})]^T$ are K weak learners weighted by the vector $\mathbf{b}_d = [b_{d,1} \dots b_{d,K}]^T$. The resulting binary descriptor, which we call BinBoost$_K$-D, is a D-dimensional binary vector built using K weak learners by applying $C(\mathbf{x}) = \{C_d(\mathbf{x})\}_{d=1}^{D}$.

Substituting f_B for f_C in Equation (4.1) gives

$$\mathcal{L}_B = \sum_{n=1}^{N} \exp\left(-\gamma\, l_n \sum_{d=1}^{D} C_d(\mathbf{x})C_d(\mathbf{y})\right) . \tag{4.10}$$

This optimization problem is closely related to Equation (4.4), but instead of weighting the dimensions with different α_d values we use a constant weighting factor γ. This enables us to compute the similarity more efficiently as it is now equivalent to the Hamming distance. More importantly, the $\{C_d(\cdot)\}$ functions are much more complex than the weak learners h_d as they are thresholded linear combinations of weak learner responses. The resulting optimization is discontinuous and non-convex and in practice the space of all possible weak learners h is discrete and prohibitively large. In what follows we develop a greedy optimization algorithm to solve this difficult problem and jointly optimize over the weak classifiers of each bit, $\mathbf{h}_d$ and their associated weights $\mathbf{b}_d$.

We first proceed as in regular AdaBoost. We optimize the $\{C_d(\cdot)\}$ functions iteratively, and at iteration d, the $C_d(\cdot)$ function that minimizes Equation (4.10) is also the one that maximizes the weighted correlation of its output and the data labels [96]. Using this fact, at iteration d, the optimal $\mathbf{b}_d$ and $\mathbf{h}_d$ can be taken as

$$\underset{\mathbf{b}_d,\mathbf{h}_d}{\arg\max} \sum_{n=1}^{N} l_n\, W_d(n) C_d(\mathbf{x})C_d(\mathbf{y}) , \tag{4.11}$$

where

$$W_d(n) = \exp\left(-\gamma l_n \sum_{d'=1}^{d-1} C_{d'}(\mathbf{x})C_{d'}(\mathbf{y}))\right) \tag{4.12}$$

is a weighting that is very similar to the one used in regular Adaboost. This means that pairs that are incorrectly classified by the previous iterations are assigned a higher weight, whereas the weight of those correctly classified is decreased.

The sign function in $C_d(\cdot)$ is non-differentiable, and Equation (4.11) is thus still hard to solve. We therefore apply the spectral relaxation trick [60, 127] and approximate the sign function using its signed magnitude, $\text{sgn}(x) \approx x$. This yields:

$$\begin{aligned}
&\operatorname*{arg\,max}_{\mathbf{b}_d,\mathbf{h}_d} \sum_{n=1}^{N} l_n\, W_d(n) C_d(\mathbf{x}) C_d(\mathbf{y}) \\
\approx\;& \operatorname*{arg\,max}_{\mathbf{b}_d,\mathbf{h}_d} \sum_{n=1}^{N} l_n\, W_d(n) \left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x}_n)\right) \left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{y}_n)\right) \\
=\;& \operatorname*{arg\,max}_{\mathbf{b}_d,\mathbf{h}_d} \sum_{n=1}^{N} l_n\, W_d(n) \left(\mathbf{b}_d^T \mathbf{h}_d(\mathbf{x}_n)\right) \left(\mathbf{h}_d(\mathbf{y}_n)^T \mathbf{b}_d\right) \\
=\;& \operatorname*{arg\,max}_{\mathbf{b}_d,\mathbf{h}_d} \mathbf{b}_d^T \left(\sum_{n=1}^{N} l_n\, W_d(n) \mathbf{h}_d(\mathbf{x}_n) \mathbf{h}_d(\mathbf{y}_n)^T \right) \mathbf{b}_d \,.
\end{aligned} \tag{4.13}$$

We first select a vector $\mathbf{h}_d(\mathbf{x})$ of suitable weak classifiers by minimizing Equation (4.4) on the training samples initially weighted by the $W_d(n)$ weights. The sign function in the expression of $C_d(\cdot)$ makes $\mathbf{b}_d$ defined only up to a scale factor, and given an estimate for $\mathbf{h}_d(\mathbf{x})$, we solve for $\mathbf{b}_d$ by looking for

$$\operatorname*{arg\,max}_{\mathbf{b}_d} \mathbf{b}_d^T \mathbf{M} \mathbf{b}_d, \text{ s.t. } \|\mathbf{b}_d\|_2 = 1 \tag{4.14}$$

where

$$\mathbf{M} = \sum_{n=1}^{N} l_n\, W_d(n) \mathbf{h}_d(\mathbf{x}_n) \mathbf{h}_d(\mathbf{y}_n)^T \,. \tag{4.15}$$

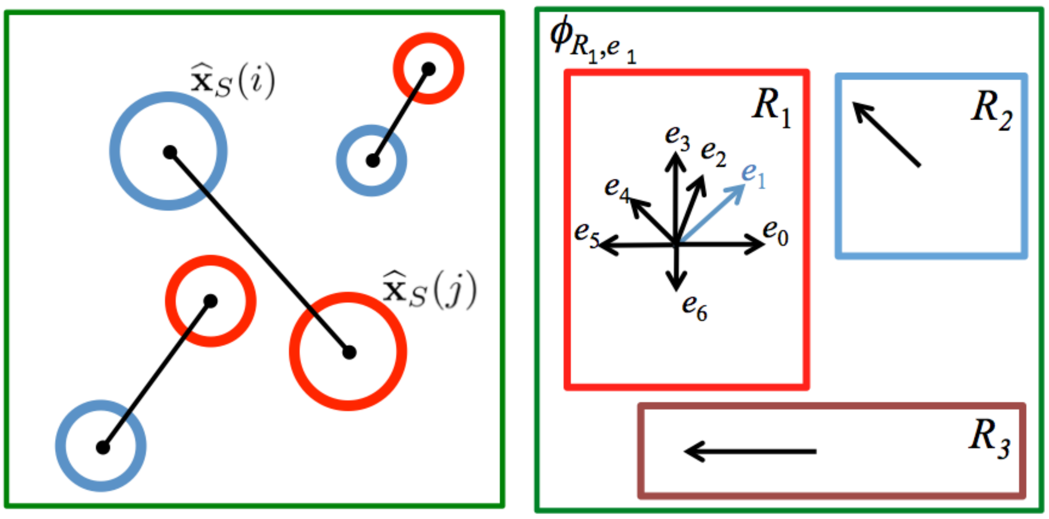
Eq. (4.14) defines a standard eigenvalue problem and the optimal weights $\mathbf{b}_d$ can therefore be found in closed-form as the eigenvector of $\mathbf{M}$ associated with its largest eigenvalue.

Although not globally optimal, this solution returns a useful approximation to the solution to Eq. (4.11). Moreover, thanks to our boosting scheme even a sub-optimal selection of $C_d(\cdot)$ allows for an effective minimization.

We still have to explain how we choose the γ parameter. Note that its value is needed for the first time at the end of the first iteration, and we set this parameter after finding C_1 using the formula from regular Adaboost. We use the rule $\gamma = \nu \cdot \frac{1}{2} \log \frac{1+r_1}{1-r_1}$ where $r_1 = \sum_{n=1}^{N} W_1(n)\, l_n\, C_1(\mathbf{x}_n) C_1(\mathbf{y}_n)$ and ν is a shrinkage parameter used to regularize our optimization as described in [43]. In practice, we use $\nu = 0.4$.

4.2 Weak Learners

The employed weak learner family encodes specific design choices and desired descriptor properties. In this section we present two weak learner types inspired from existing,



(a) Intensity-based (b) Gradient-based

Figure 4.2: Overview of the intensity and gradient-based weak learners. To compute the responses of intensity-based weak learners, we compare the image intensity values after Gaussian smoothing at two locations i and j. Using boosting, we optimize both the locations and Gaussian kernel sizes, S. The gradient-based learners consider the orientations of gradients normalized within a given region. Boosting allows us to find the pooling configuration of the gradient regions and optimize the values of the corresponding thresholds.

prevalent keypoint descriptors. The simpler, yet less discriminative weak learners are based on pixel intensities. The more complex and computationally expensive weak learners rely on gradient images. Below we provide a detailed description of each along with their parameters.

4.2.1 Intensity-based learners

The intensity-based weak learners rely on BRIEF-like comparisons of pre-smoothed image intensities. More precisely, we define the output of our weak learner as:

$$h(\widehat{\mathbf{x}}_S; i, j, S) = \begin{cases} 1 & \text{if } \widehat{\mathbf{x}}_S(i) \leq \widehat{\mathbf{x}}_S(j) \\ -1 & \text{otherwise} \end{cases} \tag{4.16}$$

where $\widehat{\mathbf{x}}_S(i)$ is the pixel intensity of $\mathbf{x}$ pre-smoothed with a Gaussian kernel of size $S \in \{3, 5, 7, \ldots, 15\}$ at position i.

The above formulation allows us to optimize the selection of the sampling points as it was done, *e.g.* in [92], except we minimize a loss function with boosting rather than the responses' corellation with a stochastic algorithm.

Inspired by the sampling scheme of BRISK [54] and FREAK [2], we also optimize the value of the Gaussian kernel size S which defines the amount of smoothing applied to the image before comparing the intensity values, in addition to the positions i and j. This adds an additional degree of freedom to our optimization framework and, therefore, encompasses the formulation of many recently proposed binary feature descriptors, such as BRISK, FREAK and ORB.

4.2.2 Gradient-based learners

The gradient-based weak learners consider the orientations of intensity gradients over image regions [4]. They are parameterized by a rectangular region R over the image patch $\mathbf{x}$, an orientation e, and a threshold T, and are defined as

$$h(\mathbf{x}; R, e, T) = \begin{cases} 1 & \text{if } \phi_{R,e}(\mathbf{x}) \leq T \\ -1 & \text{otherwise} \end{cases} , \tag{4.17}$$

with

$$\phi_{R,e}(\mathbf{x}) = \sum_{m \in R} \xi_e(\mathbf{x}, m) \;/ \sum_{e' \in \Phi, m \in R} \xi_{e'}(\mathbf{x}, m) , \tag{4.18}$$

and

$$\xi_e(\mathbf{x}, m) = \max(0, \cos(e - o(\mathbf{x}, m))) , \tag{4.19}$$

where $o(\mathbf{x}, m)$ is the orientation of the image gradient in $\mathbf{x}$ at location m. The orientation e is quantized to take values in $\Phi = \{0, \frac{2\pi}{q}, \frac{4\pi}{q}, \cdots, (q-1)\frac{2\pi}{q}\}$ with q is the number of quantization bins. As noted in [4] this representation can be computed efficiently using integral images.

4.3 Experiments

In this section, we first show the results obtained for different types of weak learners, as described in Section 4.2. We then present a set of initial experiments which validate our approach and allow us to select the correct parameters for our descriptors. Our approach improves over the state-of-the-art mostly with the binary version of our boosted descriptors, and we focus here on this version. Nevertheless the optimized parameters remain valid also for the floating-point descriptor. In all the experiments, we use the

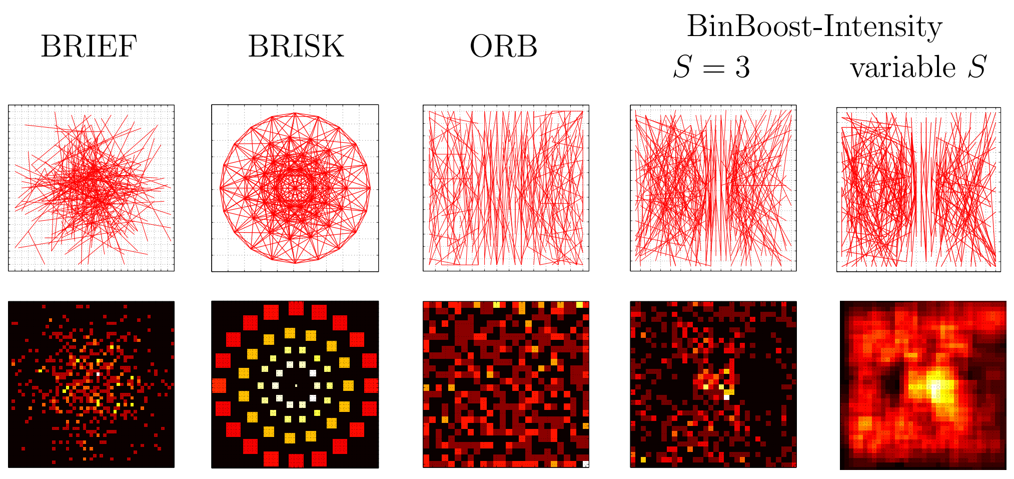


Figure 4.3: Visualization of the intensity tests (first row) and spatial weight heat maps (second row) employed by BRIEF, ORB, BRISK and our BinBoost$_1$-256 descriptor trained with intensity-based weak learners on rectified patches from the Liberty dataset. BRIEF picks its intensity tests from an isotropic Gaussian distribution around the center of the patch, while the sampling pattern of BRISK is deterministic. The intensity tests of ORB are selected to increase the variance of the responses, while reducing their correlation. This results in a pronounced vertical trend which can also be seen in the case of BinBoost and can be traced back to the fact that the patches used for training ORB and BinBoost are orientation rectified. Nevertheless, the heat maps show that the tests for BinBoost-Intensity are — similarly to BRIEF — more dense around the center of the patch while the ones used in ORB present a more uniform distribution.

Brown datasets along with the corresponding evaluation protocol, described in details in Section 5.1.

4.3.1 Weak learner types

To analyze the impact of the weak learner type on descriptor performances, we train a BinBoost$_1$-256 descriptor where each bit corresponds to one weak learner. Other configuration of the BinBoost descriptor are tested in the following sections. For our gradient-based descriptor we use $q = 8$ orientation bins, as this is equal to the number of bins proposed for SIFT.

First, we compared the sampling patterns employed in the state-of-the-art binary intensity-based descriptors, such as BRIEF, BRISK and ORB, with the pooling learned with our framework when using intensity-based weak learners. Fig. 4.3 shows the

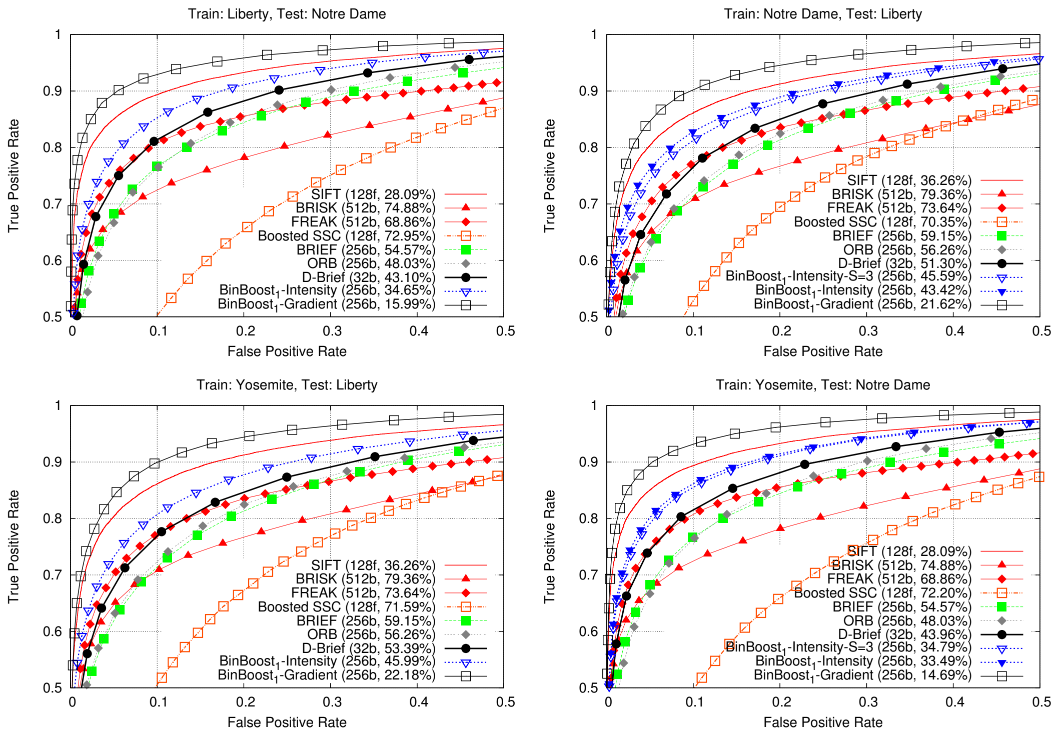


Figure 4.4: Performance of BinBoost$_1$-256 with different weak learner types compared with the state-of-the-art binary descriptors and SIFT as a baseline. Out of all descriptors based on intensity tests, BinBoost-Intensity performs the best. This shows that our framework allows to effectively optimize over the other state-of-the-art binary descriptors and boost their performances at no additional computational cost. Nevertheless, the performance of BinBoost-Intensity cannot match that of floating-point SIFT which is outperformed when using the more discriminative gradient-based weak learners (BinBoost-Gradient).

visualization of intensity tests and heat maps of the spatial weighting employed by each descriptor. For BRIEF, intensity tests are from an isotropic Gaussian distribution with the origin of the coordinate system located at the patch center [21]. By contrast, the sampling pattern of BRISK is deterministic. The intensity tests of ORB are selected to increase the variance of the responses, while reducing their correlation. This results in a pronounced vertical trend which can also be seen in the case of BinBoost. Nevertheless, the heat maps show that the tests for BinBoost-Intensity are more dense around the center of the patch, similar to BRIEF, while the ones used in ORB present a more uniform distribution.

To evaluate the influence of the weak learner type on performance, in Fig. 4.4 we compared the results obtained with BinBoost-Intensity and BinBoost-Gradient with those of Boosted SSC, BRIEF, ORB, BRISK, D-Brief and SIFT. The performance of Boosted SSC remains inferior to the other descriptors as the weak learners proposed in [98] rely on thresholding single pixel intensity values and do not provide enough discriminative power. Our experiments show that even though BinBoost-Intensity with variable Gaussian kernel size performs the best out of all the intensity-based descriptors, it is only slightly better than BinBoost-Intensity with filter size equal to 3. As shown in Fig. 4.5, our learning framework does not find a clear correlation between the size of the smoothing kernel and the distance to the patch center, contrary to the sampling pattern of BRISK. Interestingly, even though the optimized sampling scheme of ORB resembles this of BinBoost-Intensity, our framework improves the results over BRIEF much more than ORB. This may be explained when looking at the spatial weighting employed by BinBoost and ORB, where we can see that certain parts of the patch are much more densely sampled in the case of BinBoost, whereas the sampling scheme of ORB is rather uniform.

Nevertheless, BinBoost-Intensity cannot match the performance of SIFT as the discriminative power of the underlying weak learners is not sufficient. When using gradient-based weak learners, we are able to outperform 128-dimensional floating-point SIFT with only 256 bits. Since the performance of gradient-based weak learners remains superior to the intensity-based learners, we use only the former to compute our final BinBoost descriptor. However, BinBoost-Intensity remains a highly efficient and more powerful alternative to the state-of-the-art intensity-based binary descriptors such as BRIEF or ORB, especially in the case of real-time mobile applications, where computing image gradients fast enough may not be feasible.

4.3.2 Numerical parameters

Our boosting framework defines a generic optimization strategy that unlike many previous approaches, such as [17], does not require fine tuning of multiple parameters. BinBoost has only three main parameters that provide a clear trade-off between the performance and complexity of the final descriptor: the number of orientation bins used by the weak learners, the number of weak learners, and the final dimensionality of the

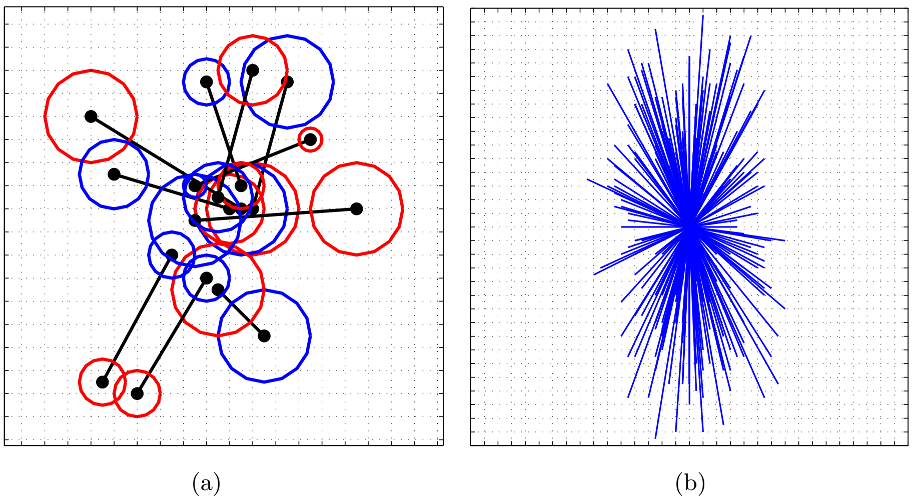


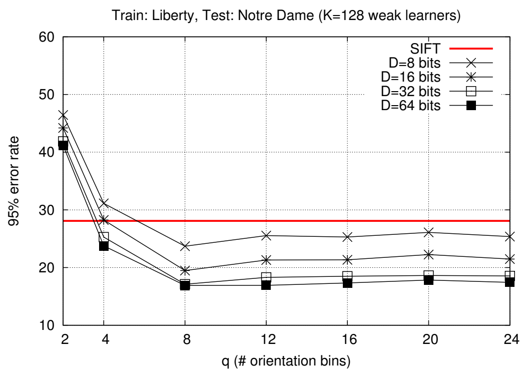
Figure 4.5: (a) Visualization of the first ten intensity-based weak learners with variable kernel size S trained on Liberty dataset. When optimizing on both the pixel positions and the sizes of the Gaussian kernels, our boosting framework does not yield a clear pattern, in particular there is no clear correlation between the size of the smoothing kernel and the distance to the patch center, contrary to the sampling pattern proposed for BRISK. It nevertheless outperforms BRISK in our experiments. (b) Visualization of the distances between the sampling points and orientations and their orientations.

descriptor. We study below the influence of each of them on the performance of our descriptor.

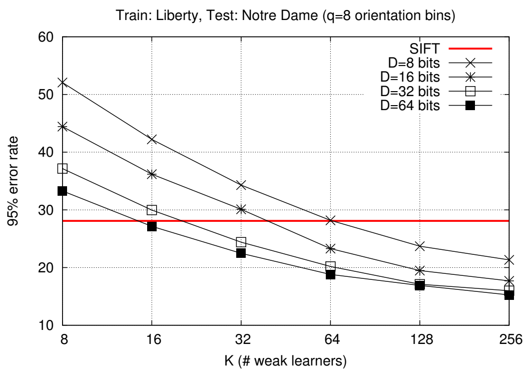
Number of orientation bins q defines the granularity of the gradient-based weak learners. Fig. 4.6(a) shows the results obtained for different values of q and D. For most values of D, the performance is optimal for $q = 8$ as finer orientation quantization does not lead to any performance improvement and we keep $q = 8$ in the remaining experiments. Interestingly, this is also the number of orientation bins used in SIFT.

Number of weak learners K determines how many gradient-based features are evaluated per dimension and in Fig. 4.6(b) we show the 95% error rates for different values of K. Increasing the value of K results in increased computational cost and since performance seems to saturate after $K = 128$, we keep this value for our final descriptor.

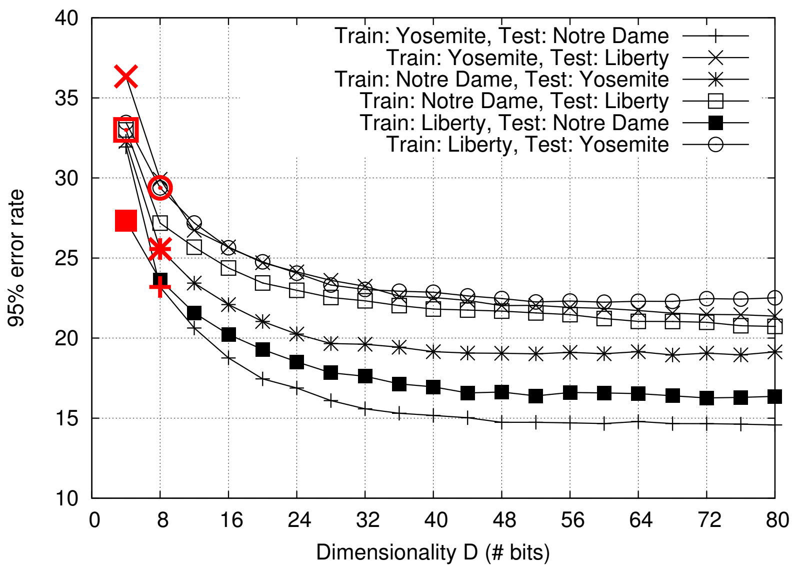
Dimensionality D is the number of bits of our final descriptor. Fig. 4.6(c) shows



(a)



(b)



(c)

Figure 4.6: Influence of **(a)** the number of orientation bins q and **(b)** the number of weak learners K on the descriptor performance for dimensionalities $D = 8, 16, 32, 64$ bits. The performances are optimal with $q = 8$ orientation bins, which is also the number used in SIFT. Increasing the number of weak learners K from $K = 128$ to $K = 256$ provides only a minor improvement—at greatly increased computational cost—and, hence, we choose for our final descriptor $K = 128$. **(c)** Performance for different dimensionalities D. With $D = 64$ bits, BinBoost reaches its optimal performance as increasing the dimensionality further does not seem to improve the results. In bold red we mark the dimensionality for which BinBoost starts outperforming SIFT.

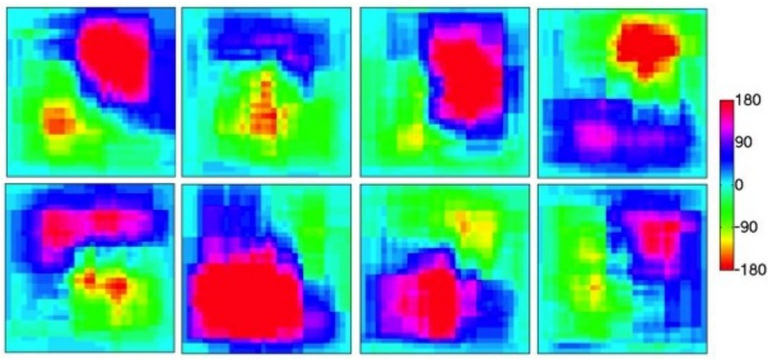


Figure 4.7: Visualization of the selected weak learners for the first 8 bits learned on 200k pairs of 32×32 patches from the Notre Dame dataset. For each pixel of the figure we show the average orientation weighted by the weights of the weak learners $\mathbf{b}_d$. For different bits, the weak learners cluster about different regions and orientations illustrating their complementary nature.

that with $D = 64$ bits, our descriptor reaches its optimal performance as increasing the dimensionality further does not seem to improve the results.

Using these parameters we trained our compact BinBoost descriptor on the Notre Dame dataset. A visualization of the learned weighting and pooling configuration is shown in Fig. 4.7 for the first 8 bits of the descriptor. The weak learners of similar orientations tend to cluster about different regions for each bit thus illustrating the complementary nature of the learned hash functions.

4.3.3 BinBoost and Binarized FPBoost

To verify if learning binary BinBoost proves to be a better approach than simply binarizing the floating-point FPBoost descriptor, we compare the performances of BinBoost with those of several binarization techniques applied to FPBoost. Results are displayed in Fig. 4.8. Binarizing the FPBoost coordinates by thresholding them at an optimal threshold found as in [111] results in large binarization errors significantly decreasing the accuracy of the resulting binary representation. This error can be reduced using Iterative Quantization [38], however, the orthogonality constraints used in this approach largely limit the extent to which it can be minimized. In contrast, sequential projection learning (S3PLH) [127] can find non-orthogonal projections that more faithfully mitigate binarization error, however, it requires a fairly large number of bits to re-

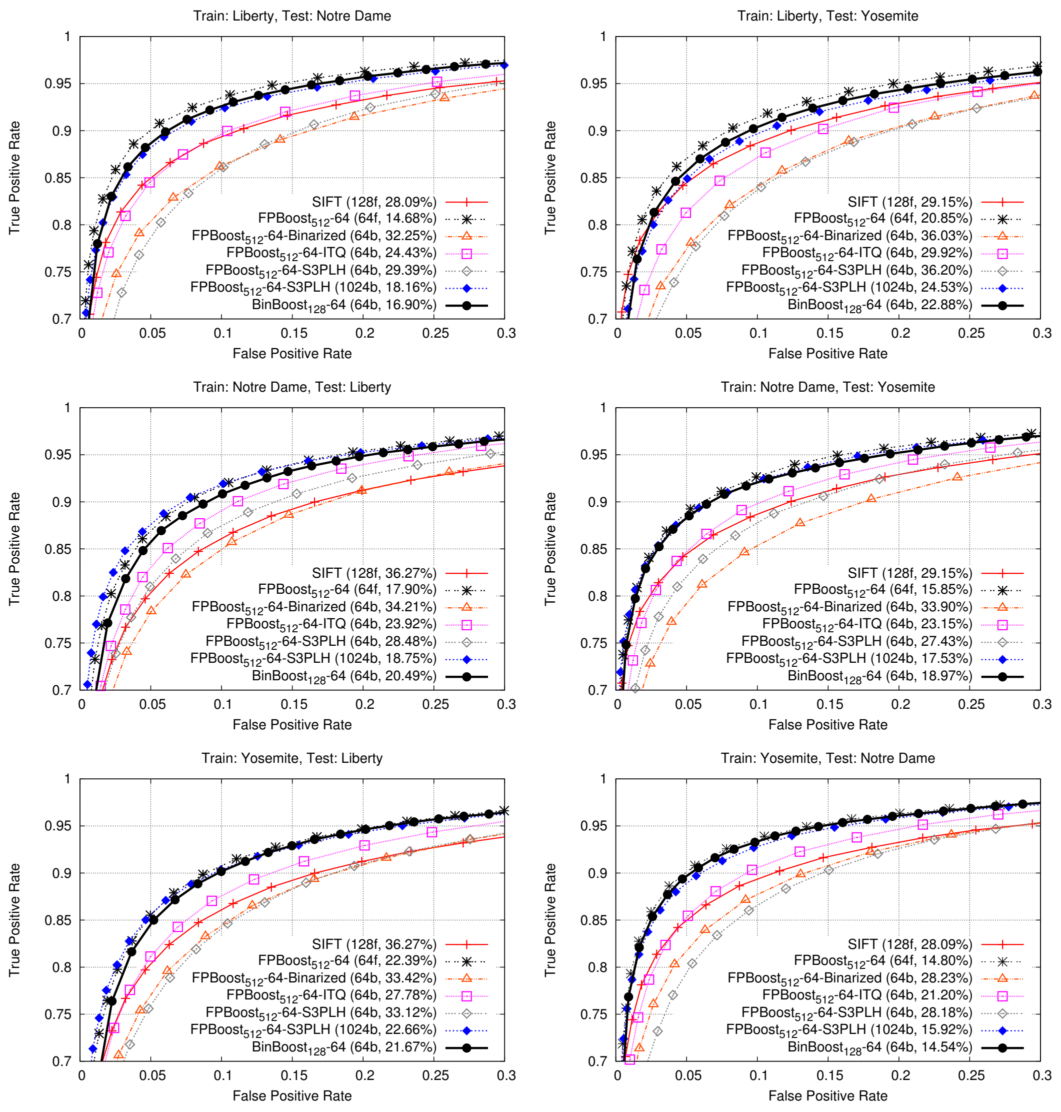


Figure 4.8: Performance of our BinBoost descriptor compared with different binarization methods applied on FPBoost. Binarizing the discriminative projections found with FPBoost either by simple thresholding or with Iterative Quantization (ITQ) results in large binarization errors significantly reducing its accuracy. On the other hand, the sequential projection learning of S3PLH requires a fairly large number of bits to recover the original performance of FPBoost. In contrast, by jointly optimizing over the feature weighting and pooling strategy of each bit, our BinBoost approach results in a highly compact and accurate binary descriptor whose performance is similar with FPBoost but at a fraction of the storage cost.

cover FPBoost's original performance. Unlike these methods, by effectively combining multiple weak learners within each hash function, learning BinBoost results in a more accurate predictor with far fewer bits.

4.4 Face Descriptors

To show that the boosting-based method proposed in this chapter is generic and can be easily adapted to new applications, in this section we evaluate it in a significantly different application, namely for matching face images. This constitutes a rather different problem than modeling the appearance of local features and it proves well that our method is not limited to learning only the feature descriptors. For our evaluation we used a dataset of face images [135] that consists of faces imaged under different viewpoints. We randomly subsampled this dataset to create two sets of 100k and 200k pairs of images. Similarly to Liberty, Notre Dame and Yosemite datasets, each set is balanced and contains an equal number of image pairs belonging to the same person as those of different people. We used the 200k dataset to train our descriptors and the 100k set to test them.

Fig. 4.9 compares the learned spatial weightings obtained with the Brown and Faces datasets. When we train our BinBoost descriptor on the images extracted around interest points, the weak learners clearly concentrate around the center of the patch. In fact, the obtained weighting closely resembles the Gaussian weighting employed by SIFT. In contrast, for face images the weak learners concentrate about the lower and upper image regions that correspond to the location of the eyes and mouth, and as also observed in [1] constitute discriminative facial features. This further demonstrates the flexibility of our approach and its ability to adapt to new types of image data.

Fig. 4.11 shows the qualitative results obtained using BinBoost$_1$-256. BinBoost remains largely invariant to the significant viewpoint and intensity changes present in this dataset, while still being able to discriminate between different people. Most of the mis-classifications are due to occlusions and extreme viewpoint variation such as side views.

In Fig. 4.10 we plot the quantitative results of BinBoost$_1$-256, FPBoost$_{256}$-64 and BinBoost$_{128}$-64 descriptors compared with LBP, a widely used face descriptor [1]. Our

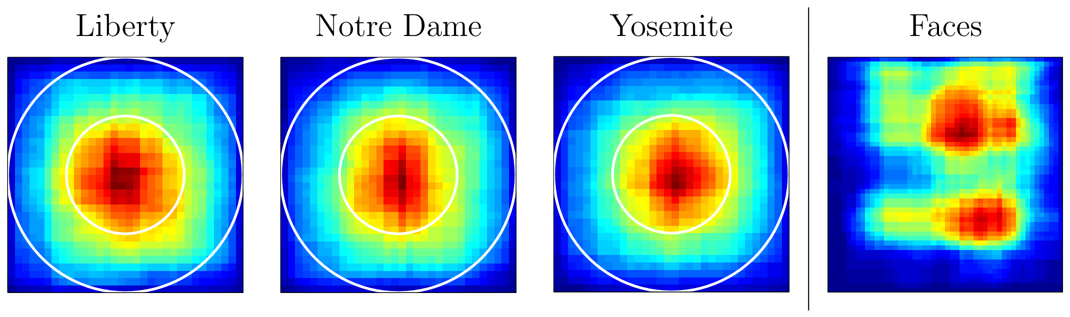


Figure 4.9: Learned spatial weighting obtained with BinBoost$_1$-256 trained on the (a) Liberty, (b) Notre Dame, (c) Yosemite and (d) Faces datasets. For the first three datasets, the learned weighting closely resembles the Gaussian weighting employed by SIFT (white circles indicate $\sigma/2$ and σ used by SIFT). However, the learned spatial weighting on the Faces dataset is focused about the eyes and mouth that constitute discriminative facial features.

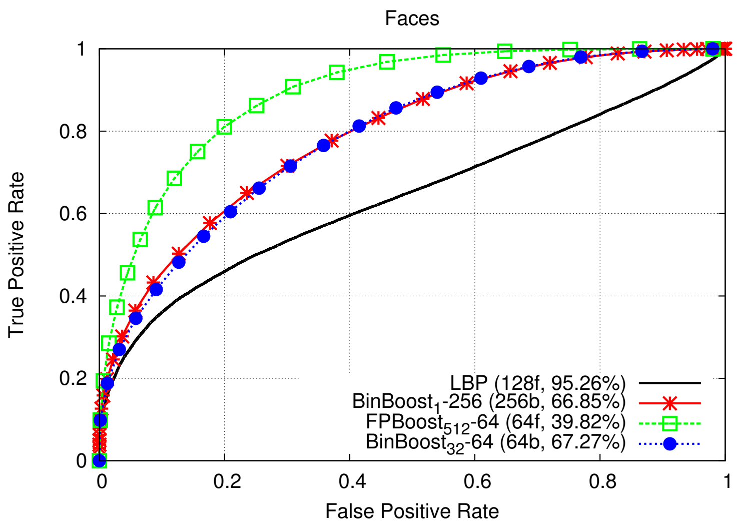


Figure 4.10: The performance of our boosted descriptors on the Faces dataset compared with the commonly used LBP face descriptor [1]. BinBoost$_1$-256 significantly outperforms LBP. Similarly to the results obtained for local feature descriptors, we can see that BinBoost$_{128}$-64 performs equally to BinBoost$_1$-256, but with only 64 bits per descriptor. FPBoost performs even better with the 95% error rate reduced by more than twice compared with the LBP baseline.

boosted descriptors result in a significant improvement over the baseline. Furthermore, compared with LBP our FPBoost descriptor achieves a reduction in 95% error rate by more than a factor of 2. Similar to [22, 130] this demonstrates the potential advantages of exploiting image data to learn a face descriptor. More importantly, it illustrates the flexibility of our approach beyond local keypoint descriptors.

4.5 Conclusion

In this chapter we presented an efficient framework to train highly discriminative and compact local feature descriptors that leverages the boosting-trick to simultaneously optimize both the weighting and sampling strategy of a set of non-linear feature responses. We first showed how boosting can be used to result in an accurate yet compact floating-point descriptor. We then considered a binary extension of our approach that shares a similar accuracy but operates at a fraction of the matching and storage cost. We explored the use of both intensity- and gradient-based features within our learning framework and performed an evaluation across a variety of descriptor matching tasks. Finally, we tested our method on a new application domain, such as faces.

As BinBoost is the second binary descriptor we proposed, in the next chapter we show the performance analysis of both approaches and their comparison with the state of the art.

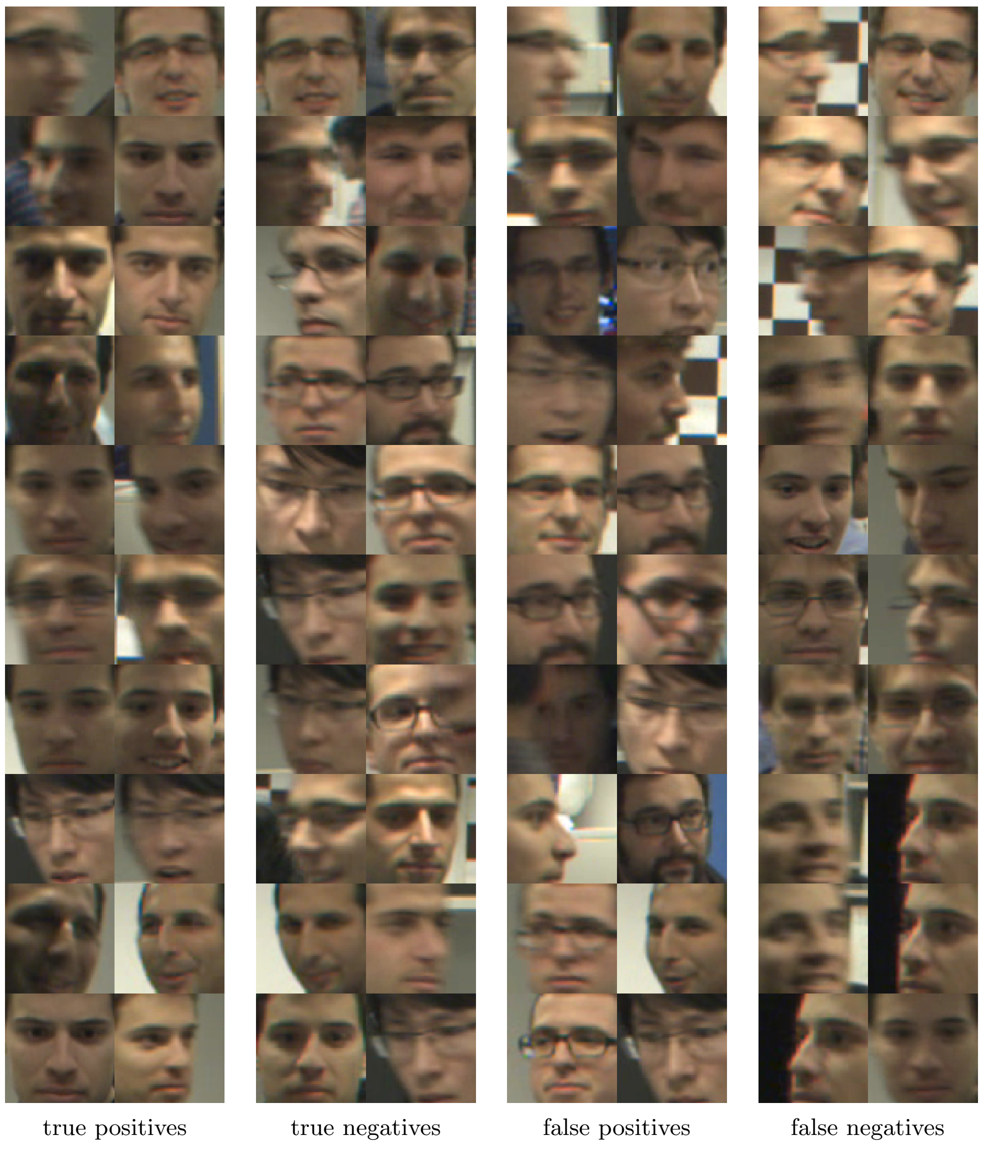


Figure 4.11: Matching results on the Faces dataset using our 256-bit BinBoost$_1$-256 at the 95% error rate, *i.e.* when 95% of the positive image pairs are correctly classified. BinBoost remains robust to significant viewpoint changes and motion blur. The misclassified examples are mostly due to occlusion and extreme variations in viewpoint such as side views.

CHAPTER
FIVE

DESCRIPTOR EVALUATION

In this chapter we provide an extensive comparison of our methods against the state-of-the-art descriptors on the Brown [17] and Mikolajczyk [71] datasets. We also show the performance our descriptors when performing visual search on the UKBench dataset [78]. Finally, we show two examples of real-life applications that use D-Brief and BinBoost descriptors.

To provide an exhaustive evaluation of the descriptor performances, we compare our approach against SIFT [62], SURF [10], the binary LDAHash descriptor [111], Boosted SSC [98], the binary ITQ descriptor applied to SIFT [38], and the fast binary BRIEF [21], ORB [92], FREAK [3] and BRISK [54] descriptors.

For SIFT, we use the publicly available implementation of A. Vedaldi [125]. For LDAHash, and ITQ we use the implementation available from their authors. As a result, we do not re-learn the discriminative projections used by LDAHash to binarize SIFT descriptors and we rely on the pre-trained ones available in the authors' implementation. For SURF, BRIEF, BRISK, ORB and FREAK we use the implementation available with OpenCV[1]. For the other methods, we use our own implementation or we report the results from the literature. For Boosted SSC, we use 128 dimensions as this obtained the best performance. When matching the descriptors we use a fast `popcount`-based implementation for computing Hamming distances between binary descriptors and matched floating-point descriptors using their Euclidean distance, unless stated otherwise.

[1]`http://opencv.org/`

5.1 Datasets and Evaluation Protocol

We compare the descriptors using three datasets along with the corresponding evaluation protocol.

5.1.1 Brown datasets

We evaluate the performance of our methods using three publicly available datasets: Liberty, Notre Dame, and Yosemite [17]. Each of them contains over 400k scale- and rotation-normalized 64×64 patches. These patches are sampled around interest points detected using Difference of Gaussians and the correspondences between patches are found using a multi-view stereo algorithm. The resulting datasets exhibit substantial perspective distortion and changing lighting conditions. Fig. 5.1 shows a selected set of patches from the Brown datasets. The ground truth available for each of these datasets describes 100k, 200k and 500k pairs of patches, where 50% correspond to match pairs, and 50% to non-match pairs. In our experiments, we use sub-sampled patches of size 32×32 and the descriptors are trained on each of the 200k datasets and we use the held-out 100k dataset for testing. We report the results of the evaluation in terms of ROC curves and 95% error rate which is the percent of incorrect matches obtained when 95% of the true matches are found, as in [17].

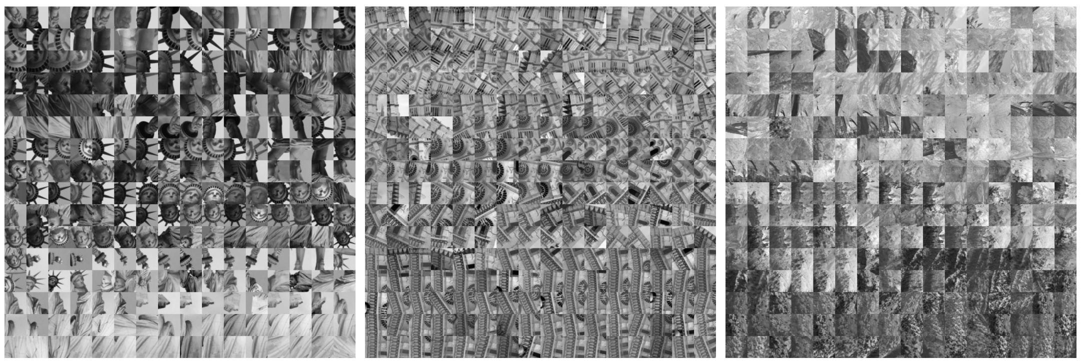
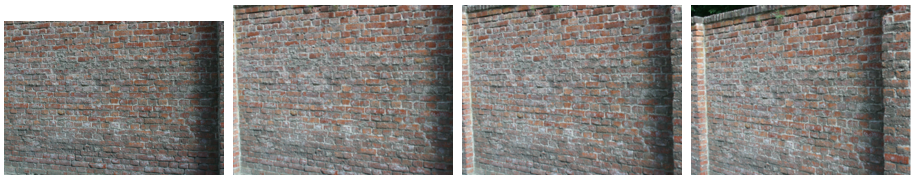
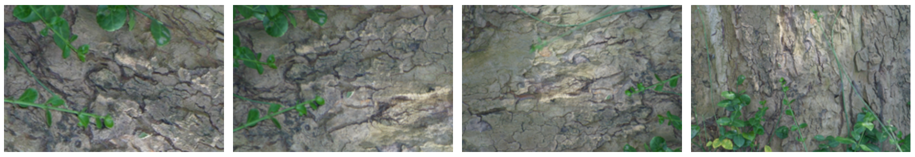

(a) Liberty (b) Notre Dame (c) Yosemite

Figure 5.1: Selected patches from the Brown datasets.

(a) `graf`: Viewpoint Change

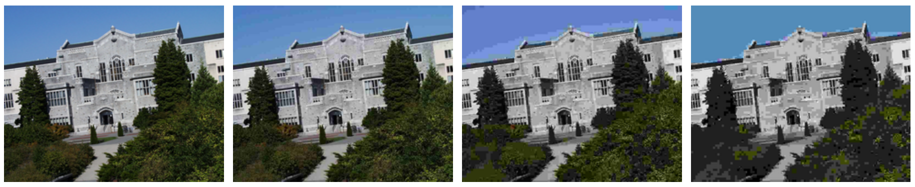


(b) `wall`: Viewpoint Change

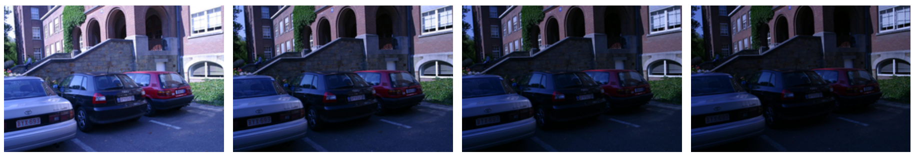


(c) `bark`: Zoom + Rotation Change

(d) `bikes`: Image Blur



(e) `ubc`: JPEG compression



(f) `leuven`: Illumination Change

Figure 5.2: The selected images from the Mikolajczyk Dataset.

5.1.2 Mikolajczyk dataset

To analyse the generalization of the methods proposed in this thesis we verify the performance of our descriptors trained on Notre Dame dataset from the Brown datasets by performing a set of experiments on a significantly different and publicly available Mikolajczyk dataset [71]. The Mikolajczyk dataset is designed to test the robustness of the descriptor against a specific type of transformation, such as blurring or viewpoint change. This dataset consists of eight sets of 5 images with each of the sets providing a different type of typical image disturbances:

- `bark`: zoom and rotation change.
- `bikes`: image blurring.
- `boat`: zoom and rotation change.
- `graf`: viewpoint change.
- `leuven`: illumination change.
- `trees`: image blurring.
- `ubc`: JPEG compression artifacts.
- `wall`: viewpoint change.

Fig. 5.2 shows a subset of images from the Mikolajczyk Dataset.

We followed the evaluation protocol of [71] that compares descriptors using the same keypoint detector, and used the OpenCV SURF Hessian-based detector. To verify the performance of our descriptors when coupled with different feature detectors, we also performed a set of experiments using OpenCV ORB detector. For each image pair we detect 1000 keypoints per image and match them using exhaustive search. We then filter outliers using a distance ratio threshold of 0.8 as in [62]. We evaluate each descriptor in terms of the recognition rate which is the number of correctly matched keypoints according to the homography matrices provided as the ground truth with the dataset.

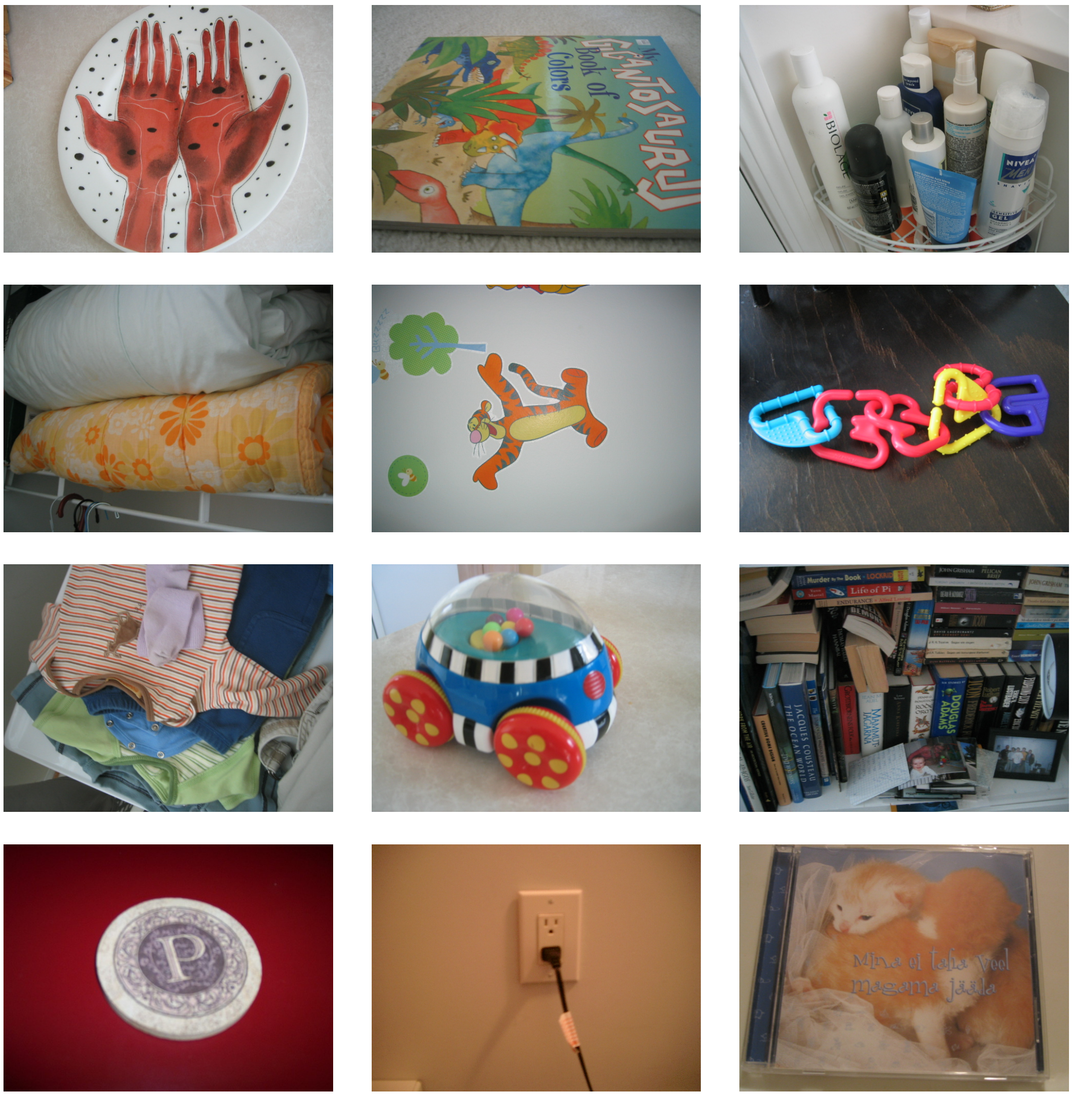


Figure 5.3: Selected images from the UK-Bench Dataset.

5.1.3 UK-Bench dataset

To evaluate the performance of the descriptors in the real-life scenario, we implemented a visual search engine using the University of Kentucky Benchmark (UK-Bench) dataset [78] that contains over 10k images of 2600 objects, each object being depicted in 4 images taken from different viewpoints. Fig. 5.3 shows a set of sample images from the UK-Bench dataset. As in other approaches, we first build a database of the almost one million descriptors extracted from all the dataset images. For each query image, we then search for the nearest neighbors in the database using their associated keypoint descriptors to vote for the most similar images in the database. Finally, we sort the database images according to the number of votes, or so-called hits, they receive and retrieve those associated with the highest number of votes. Although this approach is different than the widely used bag-of-words framework [105], it allows us to measure the influence of the descriptor type on the quality of search without the visual word quantization step, which might introduce other effects.

5.2 Results

In this section we present the results obtained for each of the evaluation frameworks discussed above.

Brown datasets

We first compare our methods using the Brown Datasets: Liberty, Notre Dame and Yosemite. Fig. 5.4 shows the ROC curves for D-Brief, BinBoost and their binary competitors. The numerical results in terms of 95% error rates are also summarized in Tab. 5.1. Both show that D-Brief outperforms its intensity-based fast-to-compute binary descriptors such as BRIEF, BRISK and FREAK, its performance, however, is inferior to this of ITQ-SIFT. This is not the case for BinBoost$_1$-64 and BinBoost$_{128}$-64 which outperform all the presented binary descriptors with only 64 bits (8 bytes). Only the recent work of Simonyan *et al.* [104] on the binarized version of their previously proposed descriptors [103] present the performances competitive to those of our BinBoost descriptor.

We also compare our binary descriptors with the state-of-the-art floating point ones and we show the results in Fig. 5.5 and Tab. 5.2. Although much shorter than its competitors, BinBoost clearly outperforms its well-known competitors such as SIFT and SURF. Moreover, it provides similar results to the state-of-the-art floating-point descriptors of [17] and [103], even though the memory footprint of their descriptors is almost 4 times greater. The real advantage of BinBoost, however, is its binary nature which allows for extremely fast similarity computation using the Hamming distance, whereas the descriptors of [17] and [103] are floating-point and cannot benefit from the same optimization, even when quantized very coarsely. As presented in Fig. 5.6, this results in a speedup of over 2 orders of magnitude in terms of similarity search.

A similar argument can be made for D-Brief: despite the fact that its performance is clearly inferior than the performances obtained for its floating-point competitors, its small memory footprint combined with fast matching is highly advantageous when used in low-cost mobile platforms.

Since the dimensionality of the binary descriptors can typically be varied depending on the required performance quality, we show in Fig. 5.7 the 95% error rates of various descriptors for different numbers of bits used. D-Brief outperforms the other intensity-based descriptors, *i.e.* BRIEF and BRISK, when their length is shorter than 64 bits. As discussed in Section 3.3, the additional bits of D-Brief introduce noise that is related to the low energy dimensions of LDE projections. On the other hand, BinBoost clearly outperforms all the competitors across all dimensions. However, the biggest improvement can be seen for lower dimensionality.

Mikolajczyk dataset

We tested the generalization performance of our descriptors when trained on the Brown datasets and evaluated on the significantly different Mikolajczyk dataset [71].

Figures 5.8, 5.9 and 5.10 show the results obtained for the `bark`, `bikes`, `boat`, `graf`, `leuven`, `trees`, `ubc` and `wall` sequences. The results obtained for D-Brief show that it can provide a very compact alternative for longer binary descriptors, such as BRIEF or BRISK, while often performing similarly to the floating-point SURF. Nevertheless, it is clear that in all the sequences BinBoost$_1$-256 and FPBoost$_{512}$-64 outperform the other descriptors. BinBoost$_{128}$-64 does not perform as well as when evaluated on the Brown datasets, which indicates that there is an inherent efficiency tradeoff when training on a different condition. Nonetheless, the extended BinBoost$_{128}$-128 and BinBoost$_{128}$-256 descriptors outperform the other methods while being shorter or of the same length.

To verify the performance of the descriptors when coupled with different feature detector, we also performed a set of experiments on the features extracted using ORB detector. The results, shown in Figures 5.11, 5.12 and 5.13, confirm that the D-Brief and BinBoost descriptors perform well even when trained on local patches extracted using different feature detectors.

Train	**Test**	BinBoost$_{128}$-64 *8 bytes*	BinBoost$_1$-64 *8 bytes*	D-Brief *4 bytes*	Simonyan *et al.* [104] *8 bytes*	ITQ-SIFT [38] *8 bytes*	LDAHash [111] *16 bytes*	BRIEF [21] *32 bytes*	FREAK [3] *64 bytes*	BRISK [54] *64 bytes*
Yosemite	Notre Dame	14.54	26.80	43.96	14.37	30.56	51.58	54.57	68.86	74.88
Liberty		16.90	29.60	43.10	15.2	31.07				
Yosemite	Liberty	21.67	33.54	53.39	23.48	37.31	49.66	59.15	73.64	79.36
Notre Dame		20.49	31.90	51.30	20.35	36.95				
Notre Dame	Yosemite	18.97	30.58	46.22	18.46	34.34	52.95	54.96	67.25	73.21
Liberty		22.88	38.13	47.29	24.02	34.43				

Table 5.1: 95% error rates for different training and testing configurations and the corresponding results for BinBoost with 8 bytes and its binary competitors. For the descriptors that do not depend on the training data, we write one result per testing dataset, for others we give the results for two different training datasets. Below the descriptor names we write the number of bytes used to encode them. D-Brief provides better results than its binary intensity-based competitors, such as BRIEF or FREAK. BinBoost significantly outperforms its binary competitors, while performing equally well as the binarized descriptor of Simonyan *et al.* [104].

Train	**Test**	BinBoost$_{128}$-64 *8 bytes*	BinBoost$_1$-64 *8 bytes*	D-Brief *4 bytes*	SURF *64 bytes*	SIFT *128 bytes*	FPBoost$_{512}$-64 *64 bytes*	Brown *et al.* [17] *29 bytes*	Simonyan *et al.* [104] *32 bytes*
Yosemite	Notre Dame	14.54	26.80	43.96	45.51	28.09	14.80	11.98	9.99
Liberty		16.90	29.60	43.10			14.68	-	9.07
Yosemite	Liberty	21.67	33.54	53.39	54.01	36.27	22.39	18.27	16.7
Notre Dame		20.49	31.90	51.30			17.90	16.85	14.26
Notre Dame	Yosemite	18.97	30.58	46.22	43.58	29.15	15.85	13.55	13.4
Liberty		22.88	38.13	47.29			20.85	-	14.32

Table 5.2: 95% error rates for different training and testing configurations and the corresponding results for BinBoost with 8 bytes and its floating-point competitors. For the descriptors that do not depend on the training data, we write one result per testing dataset, for others we give the results for two different training datasets. Below the descriptor names we write the number of bytes used to encode them. For the floating point descriptors (SIFT, SURF, FPBoost, Brown *et al.* [17], Simonyan *et al.* [104]) we assume 1 byte per dimension, as this quantization was reported as sufficient for SIFT [125]. While performance of D-Brief remains inferior due to its simplicity, BinBoost performs similarly to the best floating-point descriptors even though it is shorter and binary which enables a significant speedup in matching time as shown in Fig. 5.6.

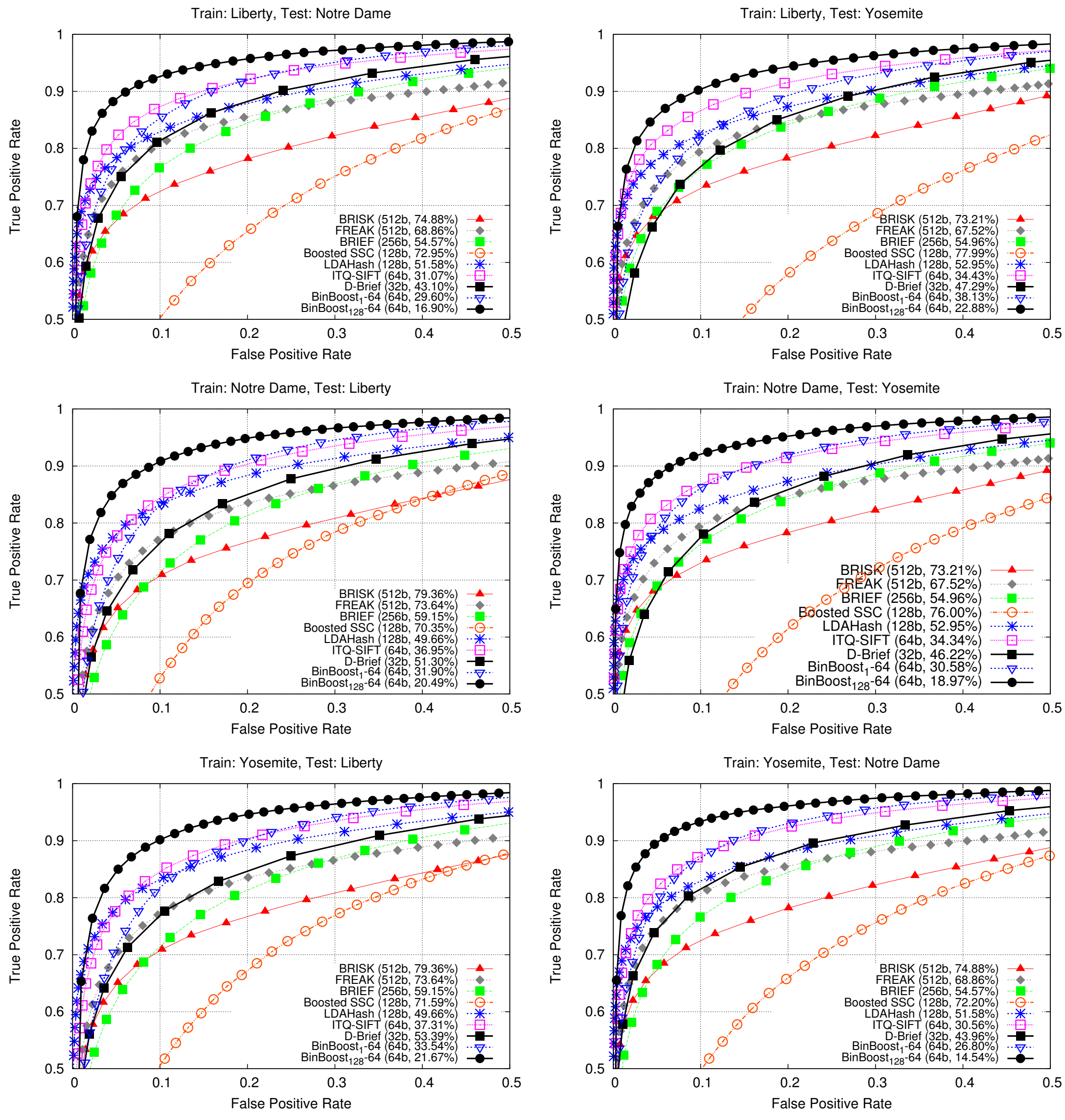


Figure 5.4: Comparison of our BinBoost descriptor to the state-of-the-art binary descriptors. In parentheses: the number of binary (b) dimensions and the 95% error rate. Our BinBoost descriptor significantly outperforms its binary competitors across all false positive rates.

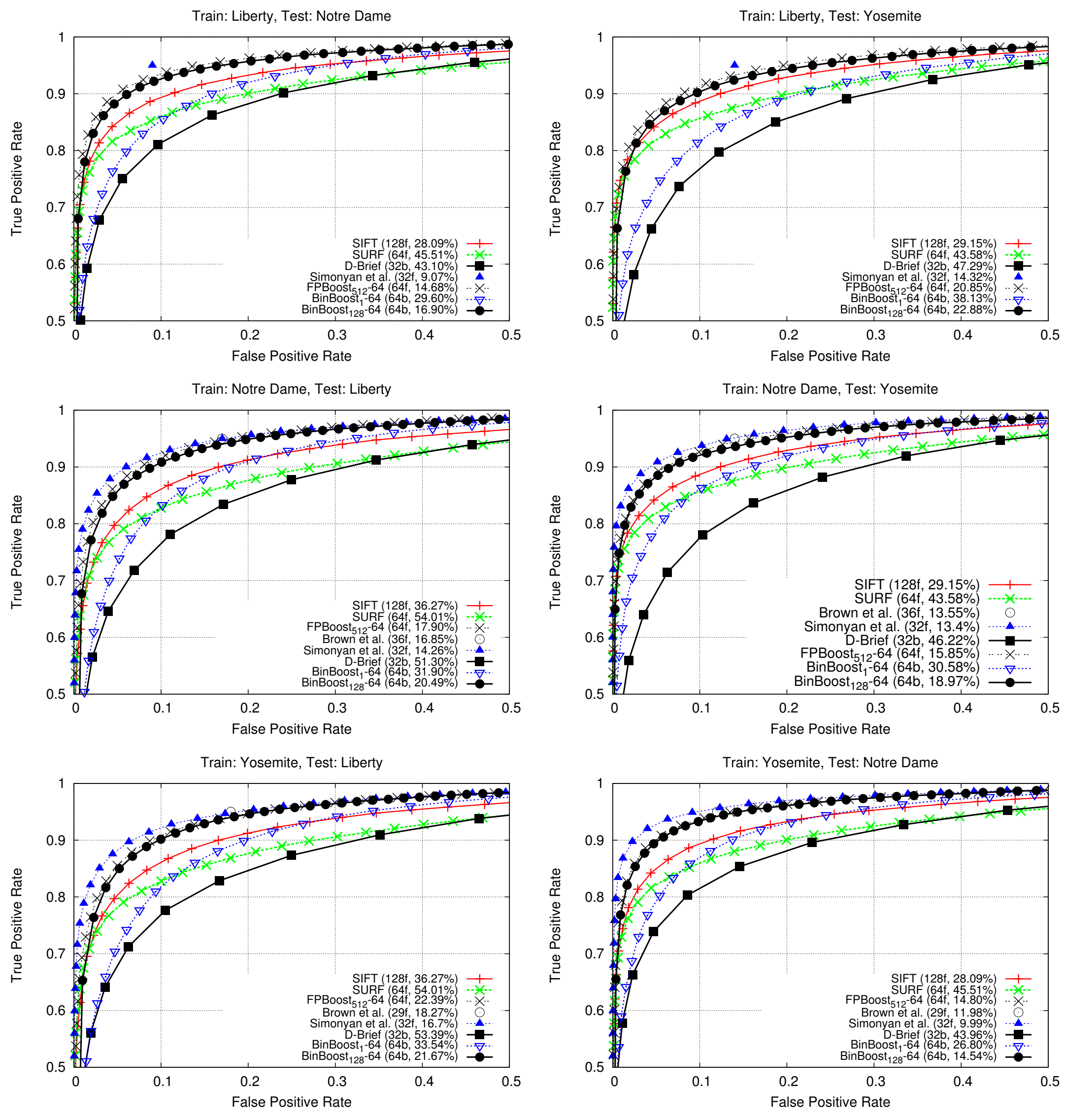


Figure 5.5: Comparison of our BinBoost descriptor to the state-of-the-art floating-point descriptors. In parentheses: the number of floating-point (f) or binary (b) dimensions and the 95% error rate. The curves for Simonyan *et al.* are available only for the subset of train-test configurations, in the other cases only a single reported 95% error rate is plotted. Our BinBoost descriptor outperforms SIFT and provides similar performances to the recent floating-point descriptors, even though it is much faster to match and has a lower memory footprint.

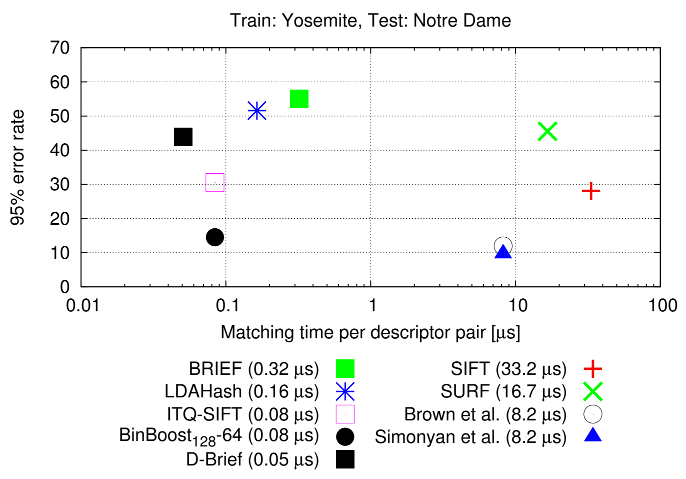


Figure 5.6: Descriptor performances as a function of their matching times. The reported times were computed from 100k test pairs (*i.e.* 100k distance computations were performed) on a Macbook Pro with an Intel i7 2.66 GHz CPU using the `popcount` instruction and averaged over 100 runs. To make the comparison fair, we optimized the matching strategy for floating-point descriptors by representing them with unsigned characters. The advantage of binary descriptors, out of which BinBoost performs the best in terms of 95% error rate, is clear.

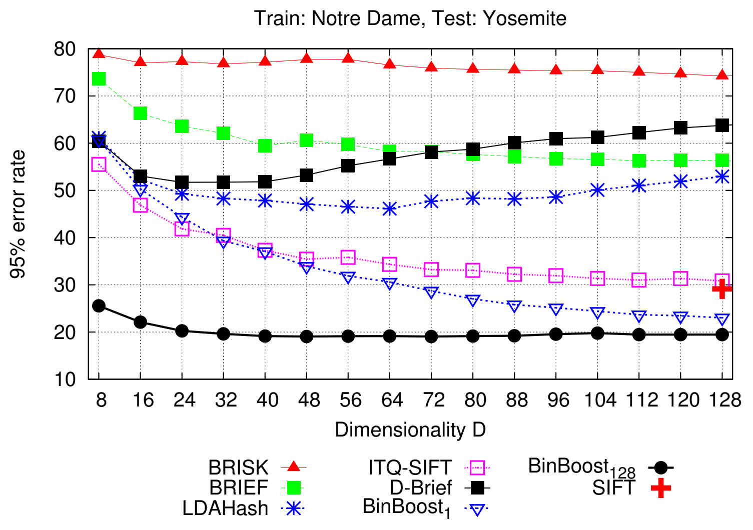


Figure 5.7: 95% error rates for binary descriptors of different dimensionality. For reference, we plot the results obtained with SIFT. BinBoost outperforms the state-of-the-art binary descriptors and the improvement is especially visible for lower dimensionality.

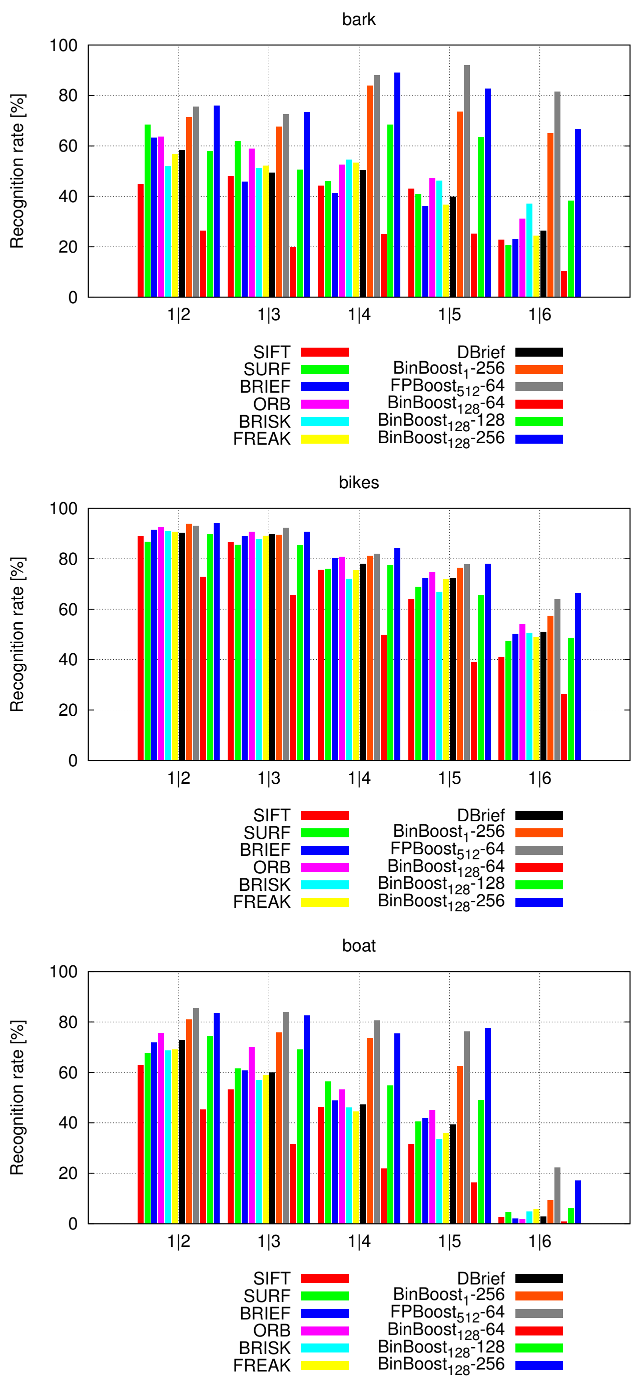
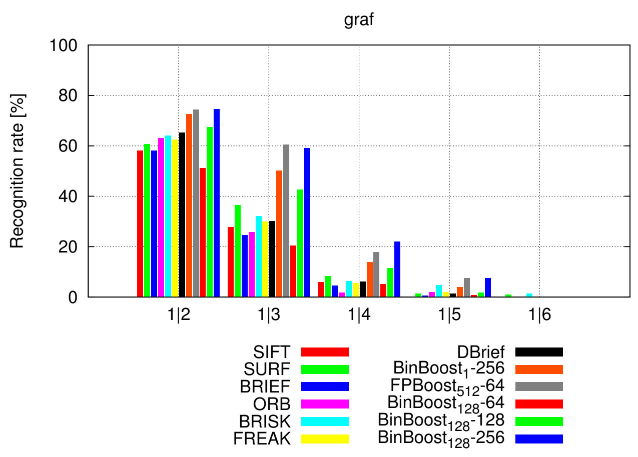
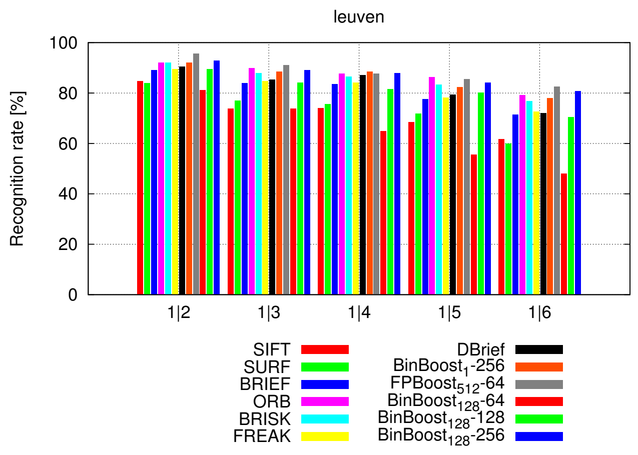


Figure 5.8: Recognition rates for the `bark`, `bikes` and `boat` sequences from the Mikolajczyk dataset (SURF detector, 1000 features extracted).





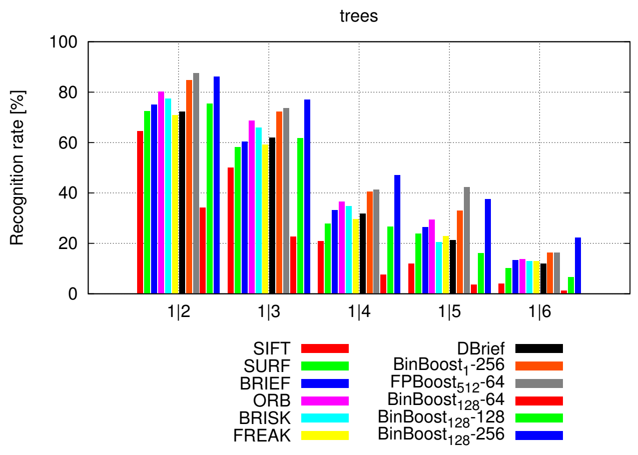
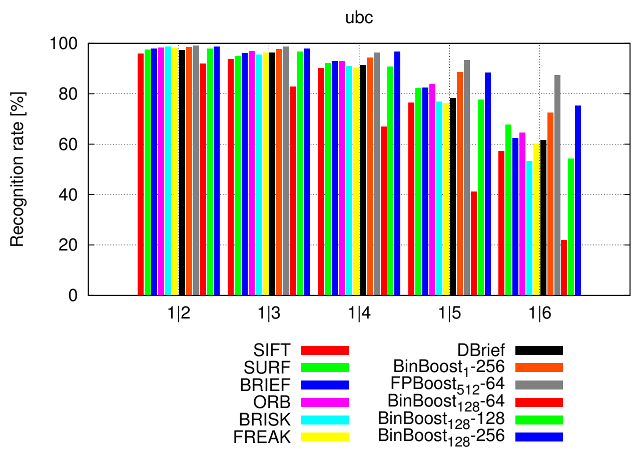


Figure 5.9: Recognition rates for the `graf`, `leuven` and `trees` sequences from the Mikolajczyk dataset (SURF detector, 1000 features extracted).



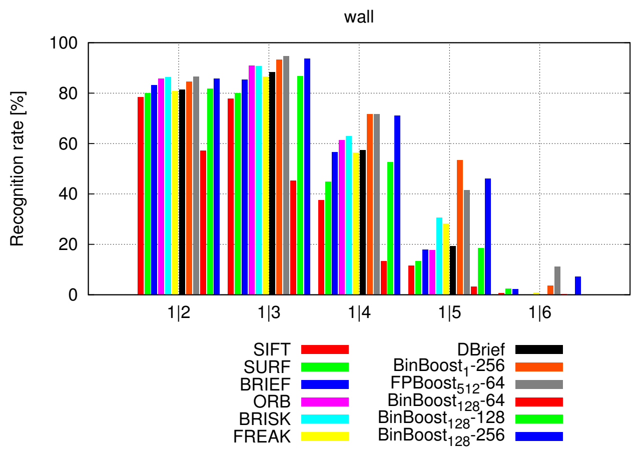
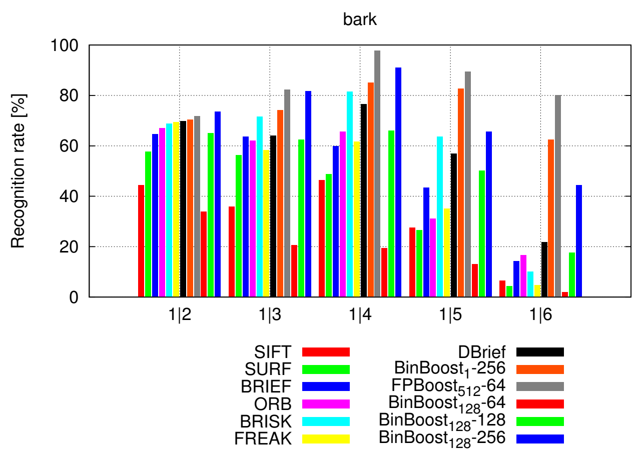


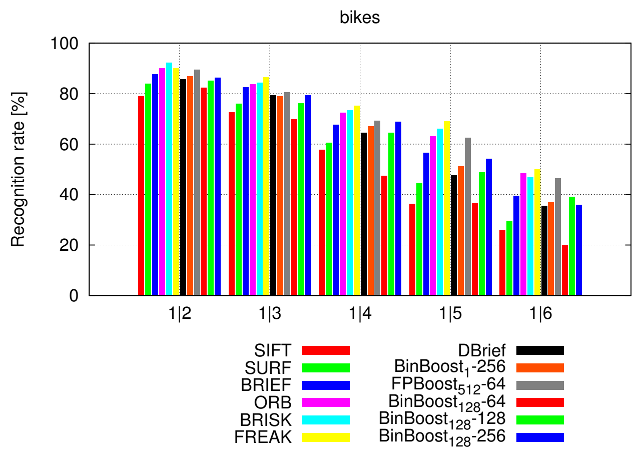
Figure 5.10: Recognition rates for the `ubc` and `wall` sequences from the Mikolajczyk dataset (SURF detector, 1000 features extracted).

UK-Bench dataset

We further evaluate our descriptors in the real-life application of visual search using the UK-Bench dataset.

Table 5.3 summarizes the results we obtained for different descriptors. To evaluate the performance we report mean average precision (mAP) and percentage of correct number of images retrieved at the top of the list (Correct@1). Similarly to the results





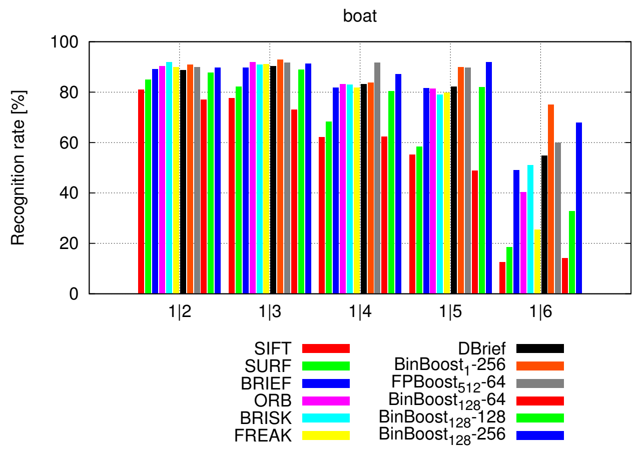


Figure 5.11: Recognition rates for the `bark`, `bikes` and `boat` sequences from the Mikolajczyk dataset (ORB detector, 1000 features extracted).

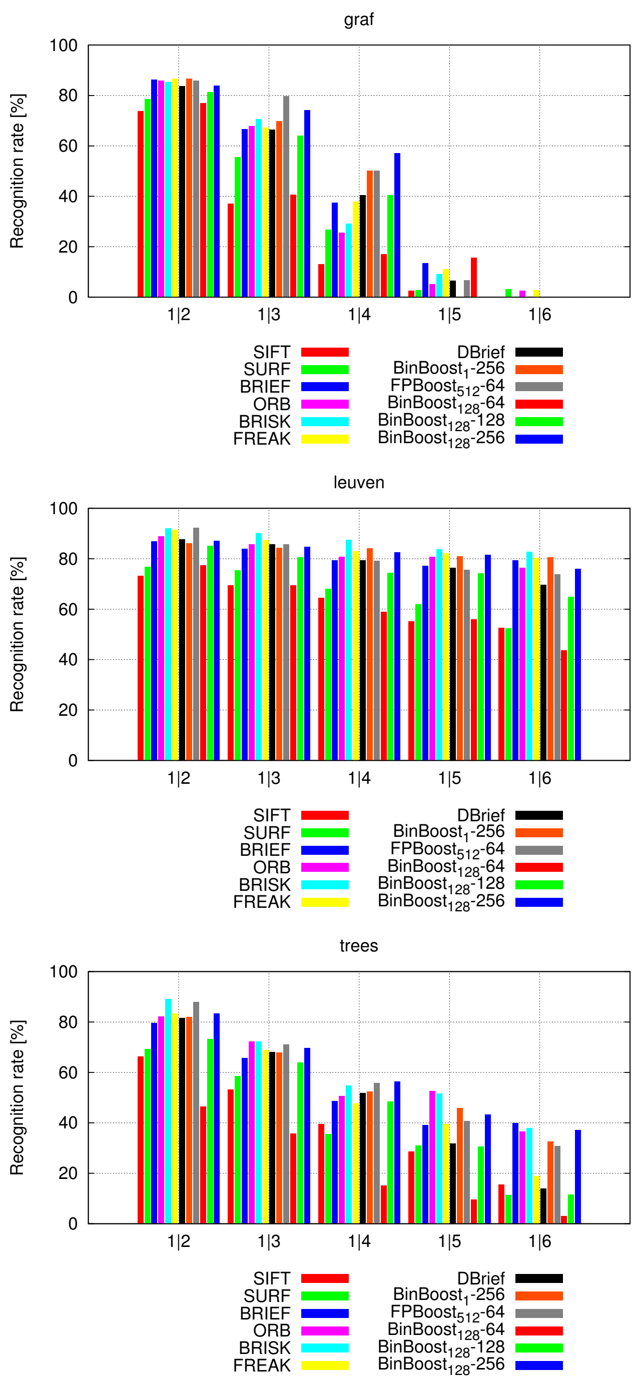
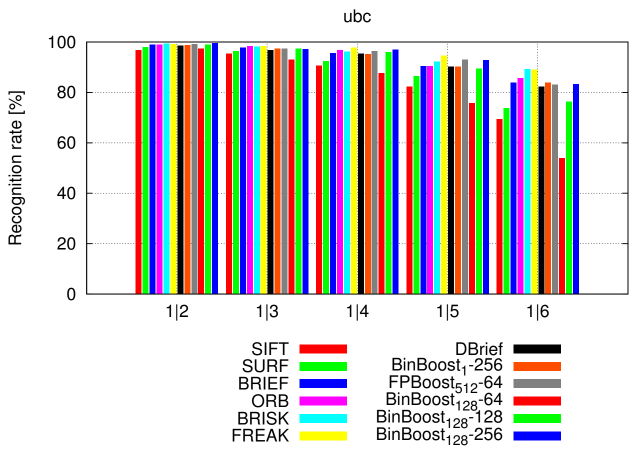


Figure 5.12: Recognition rates for the `graf`, `leuven` and `trees` sequences from the Mikolajczyk dataset (ORB detector, 1000 features extracted).



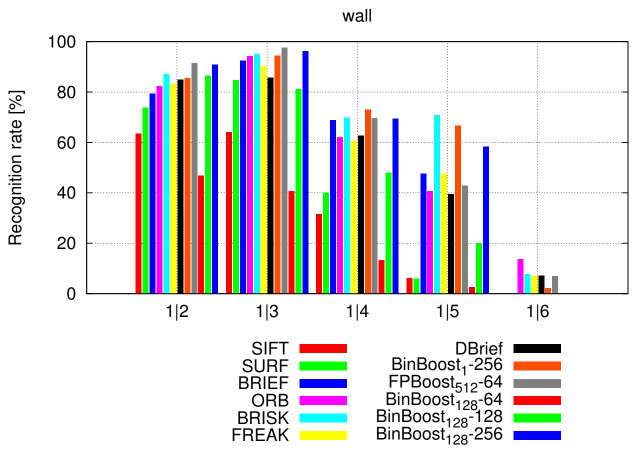


Figure 5.13: Recognition rates for the `ubc` and `wall` sequences from the Mikolajczyk dataset (ORB detector, 1000 features extracted).

presented in the previous sections, the results obtained for D-Brief are superior to those of BRISK and FREAK, although in this scenario BRIEF performs well above average, beating not only D-Brief but also SIFT. Nonetheless, out of all the evaluated descriptors BinBoost$_{128}$-256 performs the best followed by BinBoost$_1$-256. FPBoost performs slightly worse than BinBoost, while still outperforming SIFT and other intensity-based binary descriptors. Overall, the boosted keypoints descriptors provide the best performance of all the tested descriptors, even though they were trained on a significantly different dataset.

Descriptor	mAP $\pm\ \sigma$	Correct@1 $\pm\ \sigma$
BRISK	0.402 ± 0.006	61.830 ± 0.884
ORB	0.418 ± 0.005	64.902 ± 1.931
FREAK	0.429 ± 0.004	66.325 ± 0.843
D-Brief	0.432 ± 0.007	67.365 ± 2.043
SIFT	0.455 ± 0.008	68.235 ± 2.183
BRIEF	0.457 ± 0.014	68.562 ± 0.493
BinBoost$_{128}$-64	0.463 ± 0.043	68.967 ± 1.317
FPBoost$_{512}$-64	0.476 ± 0.006	70.000 ± 1.709
BinBoost$_{128}$-128	0.493 ± 0.017	72.222 ± 2.747
BinBoost$_{1}$-256	0.533 ± 0.010	76.144 ± 2.467
BinBoost$_{128}$-256	**0.556 ± 0.008**	**79.216 ± 1.870**

Table 5.3: Results of visual search on the UKBench dataset [78]: mean average precision (mAP) and percentage of correctly retrieved images at the first position (Correct@1) are reported. Average results are shown across three random train and test splits along with the standard deviation. BinBoost$_{128}$-256 outperforms the other descriptors, even though it is trained on the Notre Dame dataset. The other learned descriptors, namely BinBoost$_{1}$-256 and FPBoost$_{512}$-64, achieve worse results, though their performance is still better than SIFT and the other intensity-based descriptors.

5.3 Time Complexity Evaluation

In this section we evaluate the time performance of our D-Brief and BinBoost descriptors and their state-of-the-art competitors, namely SIFT, SURF, BRIEF, ORB, BRISK and FREAK. This measurement is crucial to determine the feasibility of the proposed descriptors as many computer vision applications that use local feature descriptors in their pipeline assume real-time or almost-real-time processing time. Hence, if the extraction and matching times for the descriptors is too long, their application in many real-life problems remains highly limited.

To compare the timings of the descriptors we extract 1000 keypoints using ORB feature detector from the 1000×700 `wall1.ppm` image from the Mikolajczyk dataset. We then measure the timings to compute and exhaustively match the above mentioned descriptors to themselves and average them over 100 runs. The timings were obtained using single-threaded code on a computer with two Intel Xeon E5620 2.4 GHz CPUs.

Table 5.4 shows the results of this experiment. First of all, the advantage of binary descriptors with respect to their floating-point competitors is evident, as they provide a speedup of over 2 orders of magnitude in terms of total extraction and matching times. Although slower than the other binary descriptors, FPBoost is still much faster to extract and match than SIFT or SURF. Out of the binary descriptors, D-Brief proves to be the fastest to match and it is as fast as BRIEF to compute. Moreover, our binary descriptors are also able to provide very fast extraction and matching. Since our implementation of boosted descriptors makes use of integral images and the *Single Instruction on Multiple Data* (SIMD) instructions on Intel processors, we are able to perform gradient-based BinBoost$_1$-256 descriptor construction almost 25% faster than the intensity-based FREAK and over 20 times faster than SIFT, while reducing the memory footprint and improving the descriptor performance, as shown in the previous sections.

Fig. 5.14 visualizes the computation times for the descriptors and clearly show that binary descriptors significantly speed up feature extraction process. Finally, Fig. 5.15 plots the relative timings and shows that the faster the descriptor computation, the more important feature detection step becomes. In the case of floating-point descriptors, detection time is negligible, while in the case of D-Brief it accounts for over 20% of the extraction time.

5.4 Real-life Applications

In this section we present two applications that incorporate binary feature descriptors proposed in this thesis. We first outline a simple object detection application where the user is asked to select the object that will be then detected in frames coming from a webcam. This application aims at real-time performance and hence we use our compact and efficient D-Brief descriptor to match incoming images with the one selected by the user. We then describe a more complex, mobile application of a treasure hunt game.

5.4.1 Object Detection with D-Brief

To demonstrate the real-time performance of D-Brief we implemented a simple real-time application for planar object detection[1]. The user can select the object of interest by drawing a rectangle around it in a reference view. The application then extracts feature points using FAST [91] and builds a database of D-Brief descriptors for these

[1]This work was based on the framework publicly available at `http://cvlab.epfl.ch/research`.

	Description [ms]	Matching [ms]	**Total** [ms]
Floating-point descriptors			
SIFT_{128f}	1021.789	173.644	1195.433
SURF_{64f}	358.822	85.840	444.662
$\text{FPBoost}_{512}\text{-}64_{64f}$	195.637	86.225	281.862
Binary descriptors			
FREAK_{512b}	64.771	12.642	77.413
ORB_{256b}	18.708	7.444	26.152
BRISK_{512b}	11.042	13.935	24.977
BRIEF_{256b}	6.928	7.441	14.369
D-Brief_{32b}	7.200	1.837	9.037
$\text{BinBoost}_{1}\text{-}256_{256b}$	52.133	7.416	59.549
$\text{BinBoost}_{128}\text{-}64_{64b}$	96.911	3.361	100.272
$\text{BinBoost}_{128}\text{-}128_{128b}$	137.100	4.163	141.263
$\text{BinBoost}_{128}\text{-}256_{256b}$	230.735	7.637	238.372

Table 5.4: Description and matching timings for SIFT, SURF, BRIEF, ORB, BRISK and FREAK and our D-Brief and BinBoost descriptors. Results computed for 1000 descriptors extracted from the `wall1.ppm` image of Mikolajczyk dataset and averaged over 100 runs. To compare the matching times, we extract another set of 1000 descriptors from `wall2.ppm` image of Mikolajczyk dataset and perform exhaustive matching using `popcount` instruction. The subscripts of the descriptor names indicate its number of binary or floating-point coordinates.

feature points in 18 rotated views at 3 scales, totaling up to 54 views. This is a simple way to make our descriptor invariant to scale and rotation changes, and was used for BRIEF in [21]. Alternatively, one could estimate the scale and orientation of the feature points, and compute the descriptors on the rectified patches as was done in ORB for example.

At run-time, for each input image, the application simply extracts feature points, computes their D-Brief descriptors, matches them against all the descriptors of the database, and finally computes the homography between the reference view and the input image using RANSAC. Some screenshots are shown in Fig. 5.16. One shall note that the projection matrix and thresholds of D-Brief are learnt on images from Notre Dame dataset, whose quality differs significantly from the quality of webcam images. Despite of that, the matching results are very consistent and the correct homography

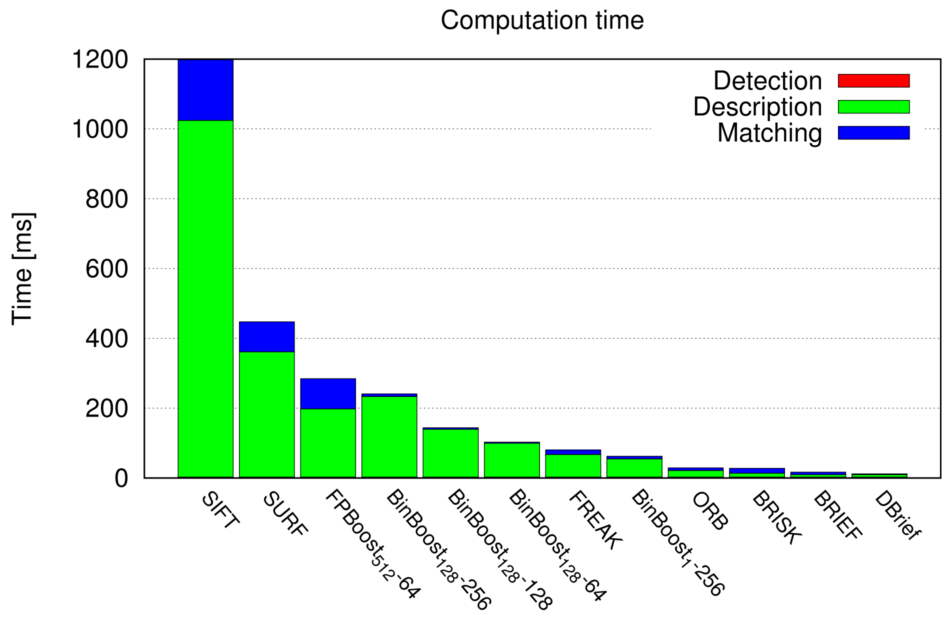


Figure 5.14: Visualization of the computation times from Tab. 5.4 for different descriptors. The detection of 1000 feature points was done with ORB detector and it took approximately 3ms. Clearly, the description and matching times for the floating-point descriptors are orders of magnitude higher than those of binary descriptors. Out of binary descriptors, D-Brief and BRIEF are the fastest to compute, while BinBoost description time depends on the number of weak learners and output dimensions. Nevertheless, BinBoost$_1$-256, *i.e.* the simplest version of our boosted descriptor, can be computed as fast as the intensity-based FREAK descriptor, while providing better recognition performances, as shown in the previous sections.

is easily found.

5.4.2 Treasure Hunt with BinBoost

We demonstrate the potential of compact binary boosted descriptors by integrating them into a mobile application for treasure hunt game[1]. It relies on an offline visual search engine system inspired by the Video Google approach [105]. The application was developed for the iOS system and tested with both an iPhone 4 (with 512MB RAM, a 1GHz ARM Cortex-A8 processor underclocked to 800MHz) and an iPad mini (with

[1]This work was done in collaboration with Aniruddha Loya, a Master student of CVLab, who merged BinBoost code into a visual search engine.

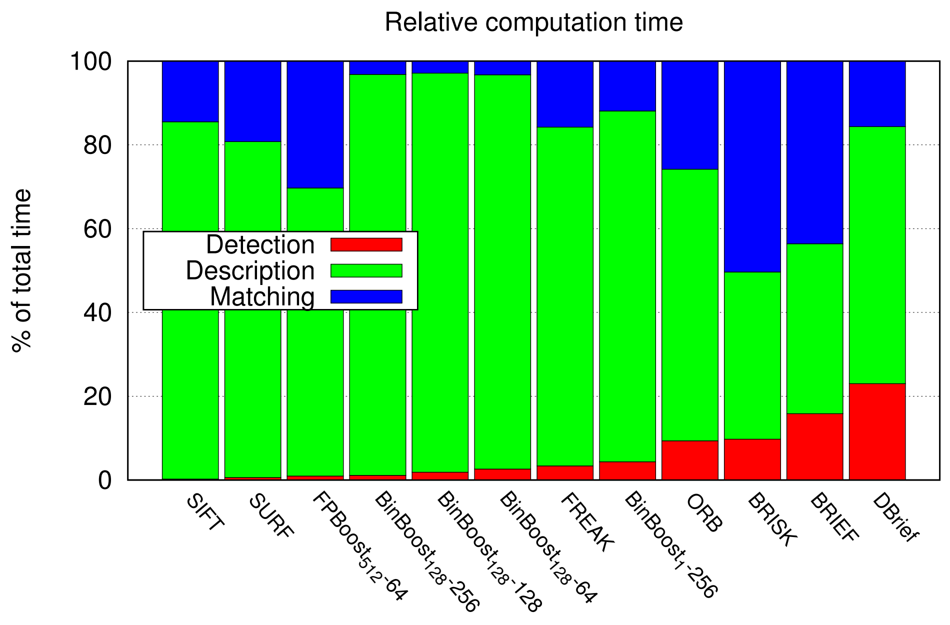


Figure 5.15: Relative detection-description-matching times as a percentage of total feature extraction time. As the descriptor computation time decreases, detection and matching times become more and more significant. This is especially visible for the fastest D-Brief descriptor whose detection and matching times account for 50% of the extraction time.

512 MB RAM, a 1GHz dual core ARM Cortex-A9 processor).

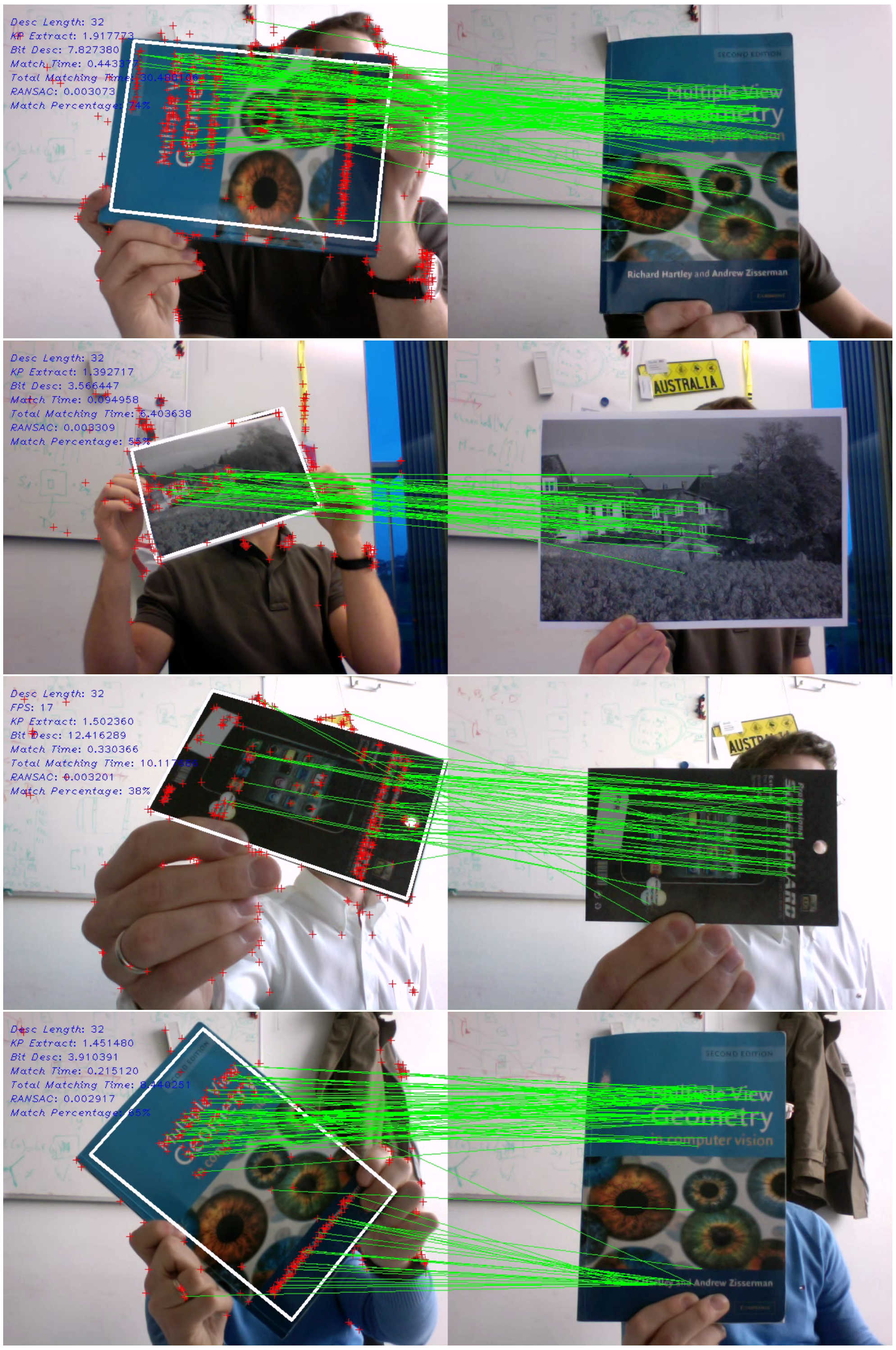
A sample set of screenshots is shown in Fig. 5.17. The user is presented with a clue about the next place or object where he should go. Following the clue should lead the user to a place where he can capture an image of the requested object with the camera (in the demo version the user can also select an existing image save in the memory of his phone). Pressing the verify button sends the selected image as a query to our visual search engine. The top result from visual search is matched against the ground truth to determine the success or failure. If the user selects the picture of the object that corresponds to the clue, a next clue is displayed and he continues the game. Otherwise, he needs to look for another object.

The visual search engine database consists of 1000 images from the UK-Benchmark dataset, discussed in Sec. 5.1.3, and a set of 10 ”clue images” which correspond to the clues presented to the user. The maximum resolution of the images is 800 $\times$ 600. To build the database, for each input image we first detect 1000 features using the OpenCV

SURF detector and then compute a set of corresponding BinBoost$_{128}$-256 descriptors. Following the Video Google approach, the database images are then clustered into a *tf-idf* structure by quantizing the binary descriptors into 5-byte *visual words*. To quantize the descriptor, we simply concatenate the first 5×8 bits of the descriptor and use the resulting value as the index of the quantized descriptor. Selecting the number of concatenated bytes used to generate visual words allows us to control the granularity of the visual word vocabulary. Nevertheless, it has the disadvantage of the exponential growth of the memory needed to store all the generated visual words with respect to the number of bits used. We therefore use a sparse image representation in the space of visual words and keep in the memory only those visual words that have at least one descriptor assigned. In other words, instead of storing in memory the entire histogram of visual words for each image with most of the fields equal to zero, we only keep the list of indices of the visual words that have at least one descriptor assigned. Furthermore, a simple stop-list mechanism was implemented to decrease the number of possible visual words by discarding the visual words of the weight lower than a pre-defined threshold. These modifications allowed us to store the entire database of the image descriptors and the corresponding indexing scheme in less than 30 MB.

When the user queries the database with an image, it is preprocessed in the same manner as the database images and a set of binary descriptors is computed and quantised. We then rank the database images according to the dot product between the histograms of the visual words for the query image and the database images. Finally, we verify the spatial consistency of the matches between the query image and top ten database images using RANSAC and re-rank the list according to the number of verified matches. Once the final ranking of the database images is computed, we check the final score of the top database image retrieved and inform the user about the success or failure of the query.

During our experiments we observed a surprising performance quality drop when increasing the vocabulary size through concatenating more bits of our binary descriptors. After careful analysis, we discovered that this problem can be explained by the harsh quantisation of binary descriptors to visual words, also known as the *hard assignment*. A typical solution of this problem in the case of floating-point descriptors relies on a hierarchical clustering scheme and on assigning several visual words to one descriptor with weights corresponding to the distance between the descriptor and the nearest clusters' centers [85]. This approach, however, is not feasible in binary spaces due to the peculiarities of the binary vectors' distribution along the Voronoi boundaries and we will explain it in more details in the following section.



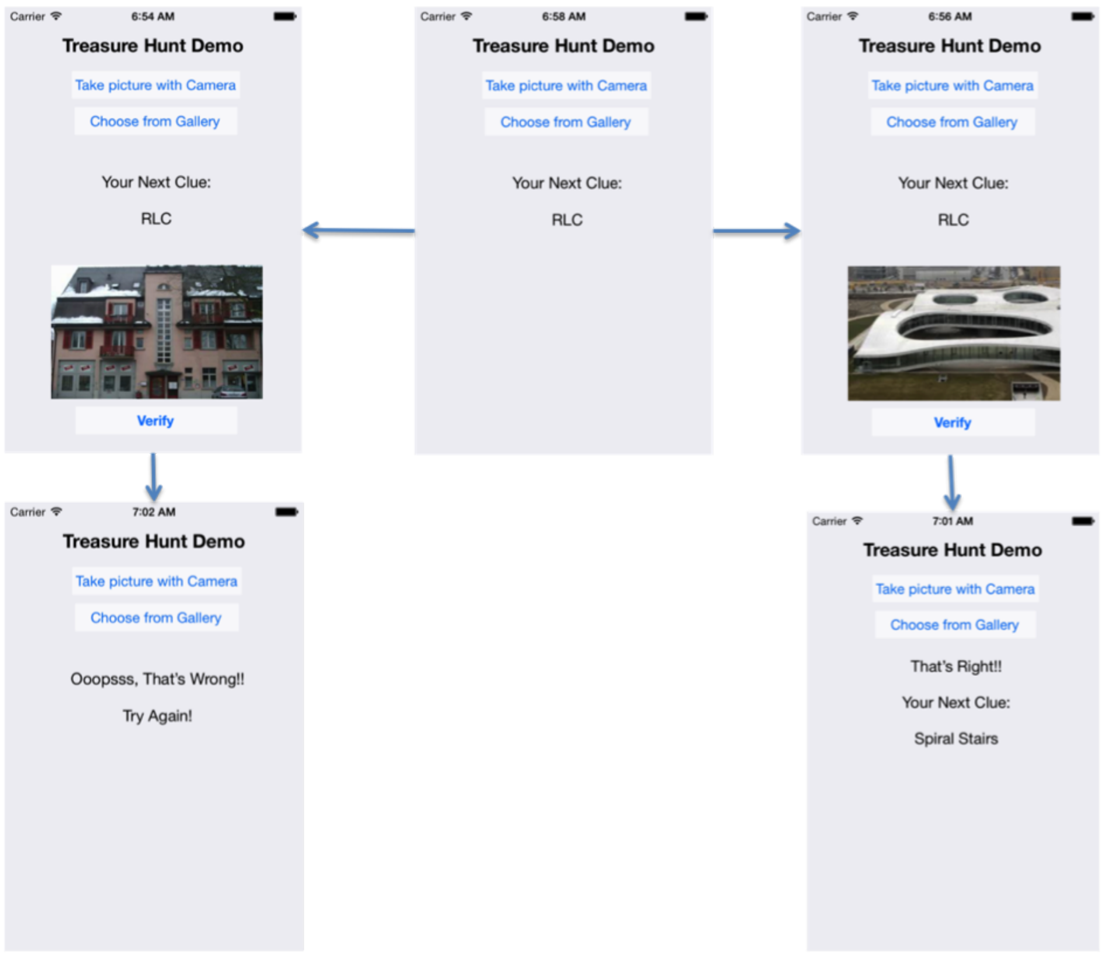
```
Desc Length: 32
KP Extract: 1.917773
Bit Desc: 7.827380
Match Time: 0.443377
Total Matching Time: 30.488106
RANSAC: 0.003073
Match Percentage: 74%
```

```
Desc Length: 32
KP Extract: 1.392717
Bit Desc: 3.566447
Match Time: 0.094958
Total Matching Time: 6.403638
RANSAC: 0.003309
Match Percentage: 55%
```

```
Desc Length: 32
FPS: 17
KP Extract: 1.502360
Bit Desc: 12.416289
Match Time: 0.330366
Total Matching Time: 10.117686
RANSAC: 0.003201
Match Percentage: 38%
```

```
Desc Length: 32
KP Extract: 1.451480
Bit Desc: 3.910391
Match Time: 0.215120
Total Matching Time: 8.440251
RANSAC: 0.002917
Match Percentage: 65%
```

Figure 5.16: Screenshots of the object detection application. The user first draws a rectangle around the target. The invariance to large scale changes and rotation is obtained by computing the feature point descriptors under different scales and orientations. This is performed on-the-fly and detecting the target runs in real-time with 27 frames per second.



Your Next Clue:

RLC

Ooopsss, That's Wrong!!

Try Again!

That's Right!!

Your Next Clue:

Spiral Stairs

Figure 5.17: Screenshots from the mobile treasure hunt application based on BinBoost descriptors showcasing a sample runtime scenario. The user is presented with a clue about the next place where he should go. If he follows the clue by taking a picture of the correct object or place and verifying it (scenario on the right), a next clue is displayed. Otherwise, the user is asked to choose another picture (scenario on the left).

Part II

Matching Binary Descriptors

CHAPTER

SIX

BINARY APPROXIMATE NEAREST NEIGHBOR SEARCH

The problem of matching high-dimensional descriptors against large databases is pervasive in Computer Vision, *e.g.* in image-retrieval, or pose-estimation. When there are millions of such descriptors, linear search becomes prohibitively expensive, even after dimensionality reduction [17, 50].

Approximate Nearest Neighbor (ANN) search constitutes one effective approach to overcoming this limitation and there are many algorithms that can handle real-valued descriptors such as the Scale Invariant Feature Transform (SIFT) [62] or Speeded Up Robust Feature (SURF) [10] descriptors. These algorithms rely on modified kd-trees [8, 12], multiple randomized kd-trees [100], hierarchical k-means (HKM) trees [35, 78], spill trees [59], vantage-point trees [134], or hashing functions [6]. A different approach to speeding up nearest-neighbor search is to binarize the real-valued descriptors using techniques such as Boosting [98], hashing [6, 51], Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) based methods [87, 111], quantization [37] and Semantic or Spectral Hashing [94, 131]. Because the similarity between the resulting binary vectors can be evaluated using the Hamming distance, which can be computed much faster than the Euclidean one on modern CPUs, linear search is more efficient but remains too slow for large-scale applications. In favorable cases, the binary vectors can be used as indices to directly access their nearest neighbors [131] which provides sub-linear complexity of the search. Unfortunately, this stops being possible when the typical Hamming distance between nearest neighbors is larger than a few units. Only recently an exact efficient algorithm to find nearest neighbors in binary spaces has been proposed [80], but its assumptions about uniform data distribution might not be correct in practice.

To get the best of both worlds under general conditions and to exploit the potential of binary descriptors, ANN search is necessary. Little attention has been paid to the performance of ANN algorithms on binary, as opposed to real-valued, vectors. Some of the algorithms discussed above such as Spectral Hashing are not adapted to binary vectors because they involve a PCA decomposition. Other methods can be used by treating binary descriptors as vectors of zeros and ones encoded as floating-point numbers. Even with the same search-accuracy, this encoding negates the advantages of binary vectors over real-valued ones: their compactness and the fact that the Hamming distance can be computed faster than the Euclidean one. Finally, there are algorithms such as vantage-point trees and HKM that can be modified to only deal with binary vectors and use the Hamming distance as a similarity measure. However, as we show, their accuracy is much lower.

In this chapter we first show that this performance loss can be traced to the fact that in Hamming spaces, unlike in Euclidean ones, the number of points that lie at the same distance from two random points, *i.e.* the points lying at the boundary of a Voronoi diagram, encompass a large proportion of the space. In other words, the Voronoi diagram has thick boundaries. This breaks the assumption made by many ANN algorithms that points can be unambiguously clustered with their closest neighbors. This phenomenon is different from the well-known *curse of dimensionality* which becomes apparent only for the high dimensional data [45]. In the case of binary spaces, the thick boundaries of the Voronoi diagram influence the search regardless of the data dimensionality.

We then present an effective way to overcome the above mentioned problems inherent to Hamming spaces by creating multiple randomized data structures. Randomization produces structures that are independent from each other and therefore complementary. It solves the thick boundary problem and yields results similar to those obtained by converting the vectors to floating point values, but at a fraction of the computational cost. We instantiate this idea in two different ways, the first inspired by HKM trees and the second by the Locality-Sensitive Hashing scheme originally proposed for integer vectors [36]. In the first case, we replace the cluster centroids computed at each level of the tree by randomly chosen points and create multiple trees in this manner. In the second, we introduce an improved mechanism for selecting random subsets of coordinates used to index the vectors.

6.1 Related Work

Matching local feature descriptors is a fundamental step in a multitude of Computer Vision applications. Typical solutions for this problem in the case of floating-point descriptor rely on approximating the distances between the descriptors using k-means clustering or tree-based methods [73]. A recent trend focuses on representing the descriptors in the binary spaces and looking for the similarities in terms of the Hamming distance [77, 111], which is much more efficient to compute than the Euclidean one. The resulting binary representation can be computed through binarization of the real-valued descriptors using various forms of hashing, *e.g.* Locality Sensitive Hashing [6], Semantic or Spectral Hashing (SH) [94, 131], or LDAHash [111]. In particular, SH was designed to create binary vectors that can be used as table indices to directly access their nearest neighbors. However, as shown in Fig. 6.1, applying it to SIFT descriptors yields too large average Hamming distances between nearest neighbors to be practical. LDAHash [111] produces average distances that are smaller but still too large.

In short, even when using sophisticated binary descriptors, quickly querying large databases still requires effective ANN methods. Recently, an interesting approach for exact binary nearest neighbour search has been proposed [80], but it provides sub-linear run-time only for uniformly distributed binary vectors, which might not always be the case. Another approach is to use hashing methods that yield low average Hamming distances as it leads to higher recall values while providing faster look-up time [40], for instance by enforcing sparsity [67]. On the other hand, many efficient approximate algorithms have been proposed for large-scale search. According to a recent comparative study [73], the best ones for querying large databases are the randomized kd-trees [100] and hierarchical k-means tree algorithm [35, 78].

The randomized kd-trees [100] are a recent modification of the original kd-trees [34], which involved building a tree by recursively splitting in half along the dimension in which it exhibits the greatest variance. This performs well in low-dimensional spaces but looses its effectiveness as dimensionality increases [5]. To prevent this, *sets* of randomized kd-trees can be built by recursively splitting along dimensions randomly

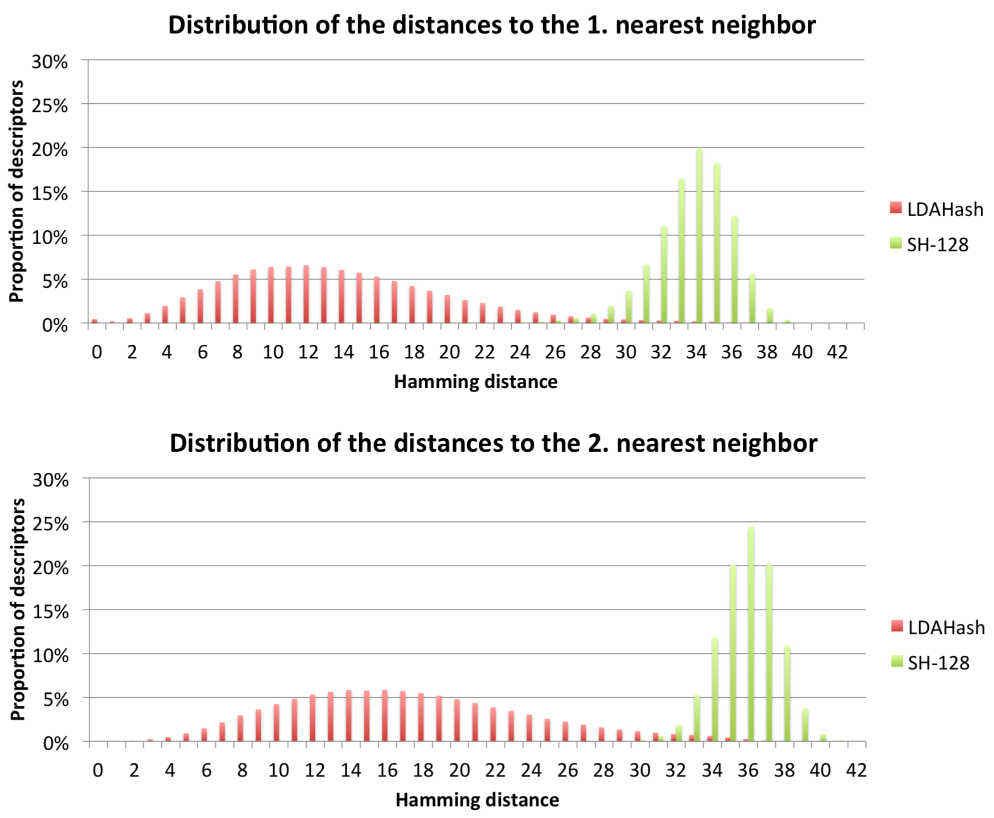


Figure 6.1: Comparison of the distributions of distances from the descriptor to its first and second nearest neighbors in the Hamming spaces generated using LDAHash and SH on our 500k database. The average Hamming distance between descriptors is larger than 1 for both LDAHash and SH-generated 128-bit descriptors. However, because the distances are spread more widely for LDAHash vectors, all ANN algorithms tend to perform better on those, which is not all that surprising since SH was designed for a different purpose.

chosen among the first D dimensions of greatest variance. Combining several trees with different splits mitigates the effects of quantization errors. Unfortunately, as we show in Section 6.2, this is a brittle technique when applied to binary vectors because a query vector can be moved to the wrong branch if only one of its bit is flipped, *e.g.* due to noise.

The hierarchical k-means tree [35, 78] represents another successful alternative to brute force search. It recursively uses the k-means algorithm to split the data into k clusters. At run-time, a query vector follows the branch that corresponds to the closest centroid and back-tracking can be invoked to explore several leaves. Hierarchical k-means rely on means of vectors, which is problematic when dealing with binary vectors as we also see in Section 6.2.

Vantage-point trees [134] avoid the need to compute means by recursively picking a single vector among the data that reaches a node and splitting the others into those that are closer and those that are further. As we will show in Section 6.2, this method also performs poorly on binary vectors.

In short, state-of-the-art ANN techniques work well on real-valued vectors but not on binary ones. Furthermore, there is little work in connection to the latter. In [23], an Additive Binary Tree (ABT) is associated to each binary vector. Each one of its nodes contains the frequency of 1's in a sub-part of the vector and this structure is used to stop the computation of the distance between two vectors early when the match is not promising. This approach, however, is still linear in the size of the database, and the speed gain is not clear compared to the full computation of the Hamming distance on modern hardware. In [72], the database is represented by a 256-ary tree in which each node corresponds to one byte of the vector. The parts of the tree that contain only one vector are pruned and replaced by a single leaf. This approach is sensitive to noise as changing a single bit may change how the branches are explored. In [26], vectors are represented by a number of random permutations of bits. For each permutation, the vectors are sorted in a lexicographic order and when the query comes, the binary search is used to find the closest vectors. Although this method provides a sub-linear complexity, the memory required to store the sorted lists is a multiple of the dataset size. Moreover, it is reported to provide identical performance to the original LSH [36] which is much more memory efficient.

Perhaps the most related to our work is a method of [74] where a tree-based structure is proposed to solve the binary approximate nearest neighbor problem. Although it resembles our HKM-inspired solution, it provides a more complex mechanism to aggregate the results of the search in multiple trees.

6.2 Thick Borders and Performance Loss

In this section, we first demonstrate that state-of-the-art ANN algorithms directly applied to binary vectors perform worse than when applied to floating-point ones. We

Precision for	**first position**		**second position**	
	SIFT descriptors	binary descriptors	SIFT descriptors	binary descriptors
hierarchical k-means with real-valued centroids	0.94	0.82	0.93	0.79
kd-trees	0.98	0.78	0.97	0.75
hierarchical k-means with binary centroids	-	0.32	-	0.29
vantage-point trees	0.35	0.17	0.31	0.15
parc-trees	0.94	0.91	0.91	0.92
Original LSH for binary vectors [36]	-	0.92	-	0.95
Uniform LSH	-	0.93	-	0.96

Table 6.1: Precisions when looking for the first and second nearest neighbors for different methods and comparable query times, approximately 0.2 ms per query, for a dataset of 500k descriptors. The performances of state-of-the-art methods drop when they are applied on the binary 128-vectors obtained by running LDAHash [111] on SIFT vectors. This is especially noticeable for the HKM algorithm when the centroids are forced to be binary vectors. The Parc-trees and Uniform LSH are two methods we introduce in Section 6.3 to avoid this loss of accuracy.

then show that Voronoi diagrams have thick boundaries in binary spaces, which is what causes this performance drop.

6.2.1 ANN on Binary Vectors

To perform the comparison, we collected many images of Venice from the Flickr [1] database and created a first dataset containing 500k feature points and their SIFT descriptors. We then binarized these using the publicly available implementation of LDAHash into 128-bits vectors [111] whose length was shown to provide a good compromise between accuracy of mapping between the original and Hamming space and the length of the codeword. We used exhaustive linear-search to find the closest neighbors of each descriptor and we use this information as a ground-truth.

Table 6.1 summarizes our precision results for the first and second positions. The first simply is the the percentage of correct nearest neighbor that are retrieved. The second is computed by retrieving two nearest neighbors and checking whether both, only one, or none are the correct first two nearest neighbors of the query. The average

[1] `http://www.flickr.com`

proportion of correct matches divided by 2 is then taken to be the precision at the second position.

The results for the kd-trees and HKM algorithms were obtained using the publicly available code of the FLANN library [73], which automatically optimizes the algorithms parameters. We used our own implementation of the vantage-point trees.

The kd-trees and vantage-point trees can work on binary vectors without any modification since they do not involve averaging. By contrast, HKM involve computing centroids. We therefore tested two different versions of the algorithm, either rounding the coordinates of the centroids so that they remain binary vectors or using the floating-point coordinates.

As a baseline, we plot in the first column the results for matching the SIFT floating-point vectors. In the second column, we plot the systematically worse equivalent results using binary vectors. The degradation is noticeable for kd-trees and HKM, even if we treat binary vectors as floating-point ones. The performance drop is even more noticeable for the vantage-point trees.

As a sanity check, even though Spectral Hashing [131] is not truly designed to produce vectors that can be searched by ANN but rather used as indices to directly access their nearest neighbors, we ran the same series of tests on the vectors it produces. The ANN precision rates are globally lower but we observed the same behavior.

6.2.2 Interpretation

That kd-trees perform poorly on binary vectors is not that surprising since the splits are performed one dimension at a time and binary vectors can take only two values per dimension. Hence this method is sensitive to flip noise.

To understand the performance drops for the HKM and the vantage-point tree, one must consider that the topology of the Hamming space is different than the Euclidean one. This is because of the discrete nature of the binary spaces where many vectors are *equidistant* to two random points.

This affects the vantage-point trees because many vectors may lie on the splitting sphere: If the dimensionality of the binary space is L and the sphere radius is D, the proportion of uniformly distributed vectors that lie on the sphere boundary is $\frac{1}{2^L}\binom{L}{D}$.

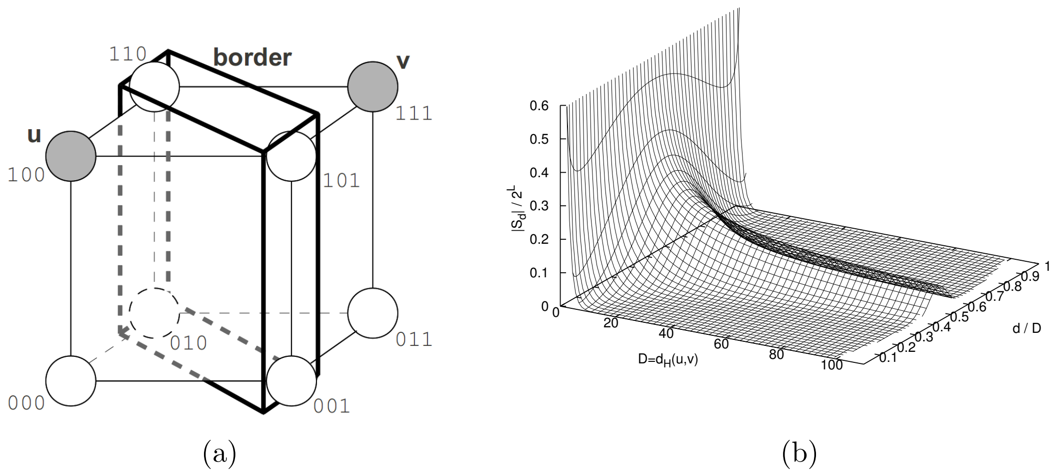


Figure 6.2: Thick borders of Voronoi diagrams in binary space. (a) A significant proportion of the space is equidistant from two arbitrary points. In this example, four vectors are equidistant to the vectors u and v which accounts for half of the population. This makes the HKM algorithm fail on binary vectors. (b) Proportion of the binary vectors w that belong to the sets S_d defined in Eq. (6.1) as a function of the distance $D = d_H(\mathtt{u}, \mathtt{v})$ and d. It is maximal for $d = D/2$, which corresponds to the set of vectors equidistant to u and v, and remains large even for large values of D. This phenomenon differs from the *curse of dimensionality* as it affects the data regardless of its dimensionality.

For example, for $L = 16$ and $D = 8$, this represents 20% of the binary space, an enormous fraction. This is problematic because the algorithm depends on the assumption that the splits separate the data well.

The same thing happens with the HKM trees, especially when one binarizes the centroids. As explained below, the boundaries of the Voronoi diagram defined by such binarized centroids contain a significant proportion of the binary space. This is detrimental to the algorithm because points in those thick boundaries can be arbitrarily assigned to one or the other cluster and can fall down the wrong branch of the tree at run-time.

Let us consider two L-dimensional binary centroids u and v. We would like to evaluate the number of vectors around the boundary defined by u and v, that is, around the hyperplane made of w vectors equidistant from u and v. To this end, let us consider the cardinality of the sets S_d defined by:

$$S_d = \{\mathtt{w} \text{ such that } d_H(\mathtt{v}, \mathtt{w}) = d_H(\mathtt{u}, \mathtt{w}) + D - 2d\} \,, \tag{6.1}$$

where $d_H(\cdot, \cdot)$ is the Hamming distance, and D the Hamming distance between $\mathtt{u}$ and $\mathtt{v}$. The S_d family spans the Hamming space, with $\mathtt{u} \in S_0$, $\mathtt{v} \in S_D$, and $S_{D/2}$ the set of vectors vectors equidistant from $\mathtt{u}$ and $\mathtt{v}$.

$\mathtt{u}$ and $\mathtt{v}$ have $L - D$ bits in common and D bits that are different. Let us first consider the case when D is even. For a vector $\mathtt{w}$ to belong to S_d, d bits among the D different bits must be changed between $\mathtt{u}$ and $\mathtt{w}$. In addition any number n of bits among the $L - D$ common bits can also be flipped between $\mathtt{u}$ and $\mathtt{w}$: We then have $d_H(\mathtt{u}, \mathtt{w}) = n + d$ and $d_H(\mathtt{v}, \mathtt{w}) = n + D - d$, and $\mathtt{w}$ indeed belongs to S_d.

The number of possible such $\mathtt{w}$ vectors is therefore $2^{L-D}\binom{D}{d}$ and their proportion of the full space is $\frac{2^{L-D}}{2^L}\binom{D}{d} = \frac{1}{2^D}\binom{D}{d}$. Remarkably this expression does not depend on the dimension L of the space but only on D and d. We plot its values in Fig. 6.2(b).

This expression reaches its maximum for $d = D/2$, that is, for the set of vectors that lie at equal distance from vectors $\mathtt{u}$ and $\mathtt{v}$. Using Stirling's approximation [108], this expression for $d = D/2$ can be approximated by $\sqrt{\frac{2}{\pi D}}$ when D increases, and therefore slowly decreases towards 0 (see Fig. 6.2(b)). For example, when $D = 2$, 50% of the space lies at equal distance from the 2 centroids! For $D = 64$, 10% of the space is still equidistant from the centroids. As a result, the borders of the Voronoi diagram defined by $\mathtt{u}$ and $\mathtt{v}$ contain a significant proportion of the binary space which leads to a severe performance drop of the HKM algorithm.

When D is odd, no vector is equidistant from $\mathtt{u}$ and $\mathtt{v}$. However, we can derive a similar expression for the number of points for which the distances to $\mathtt{u}$ and $\mathtt{v}$ differ by 1. The number of such points remains large. The borders of the Voronoi diagram defined by the centroids in the HKM algorithm therefore contain a significant proportion of the binary space.

6.3 Randomized Data Partitioning

In this section, we address the above-mentioned shortcoming of state-of-the-art ANN search algorithms in Hamming spaces, and describe two simple yet effective ANN search methods that work by randomizing data partitioning.

6.3.1 Parc-Trees

This first algorithm relies on multiple trees. Like the nodes of a hierarchical k-means (HKM) [35, 78] tree, the parc-tree nodes split the data into k parts by storing k vectors we call *centroids*, and associating the data with the closest centroid. Each non-terminal node has k children, corresponding to the different parts. By contrast with HKM, we do not optimize on the centroids but randomly select them among the data vectors that reach the node, except those which have previously been used. The recursion stops when the number of data vectors is less than k. Because of the randomization, the trees are independent from each other.

At run-time, a query vector recursively follows the branch associated to the closest node vector until it reaches a leaf, as in HKM. In HKM however, the leaves have to store all the data that reach them, and a linear search over this data is required to find the vector closest to the query vector as the candidate nearest neighbor. In parc-trees, the centroids belong to the dataset and when the query vector reaches a leaf, we already computed its distances to the centroids of the nodes it visited. Hence, a candidate nearest neighbor is chosen to be the closest vector among those in the leaf and the centroids of the visited nodes.

The query operation is repeated over all T trees and the best match is retained. This allows the parc-trees to mitigate the quantization error introduced by the thick Voronoi boundary: We can find the correct nearest neighbor even if it is present in only one visited node among all the trees.

The influence of parameters T and k on the obtained precision and computational time can be seen in Fig. 6.3. The performance increases with the number of trees T, until it saturates, linearly with T. Increasing the branching factor k also improves the performance. The average tree depth then decreases, but the computation time still increases: A tree of depth d contains $k+k^2+k^3+\dots k^d \approx k^d$ vectors, so a tree of S data vectors is approximatively of depth $\log S/\log k$. The number of distance computations required when dropping a query vector into the tree is therefore $k \log S/\log k$, which increases with k sublinearly.

Most of the operations involved by this approach are Hamming distance computations. They amount to an `xor` operation followed by a `popcount` instruction present

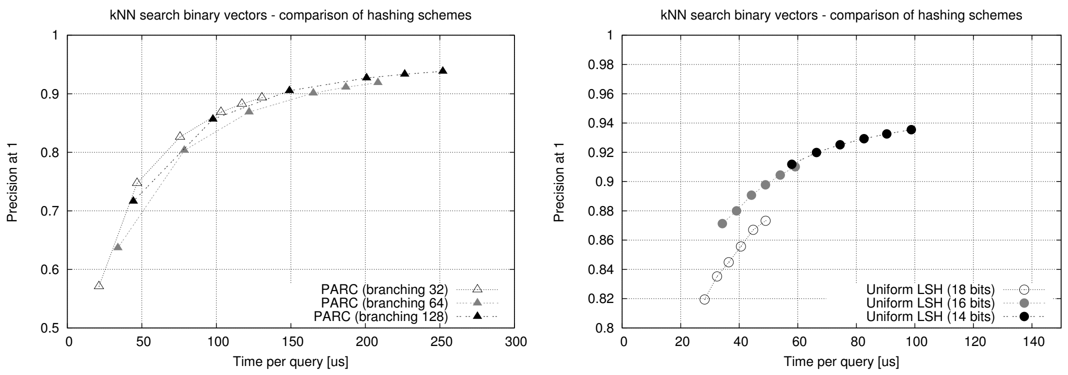


Figure 6.3: Comparison of precision for first position with different parameters for the parc-trees and Uniform LSH on the 500k binary vectors dataset. For PARC trees (left) we plot the results for $T = 8, 16, 24, 32, 36, 40$ trees and different branching factors. For Uniform LSH (right) we plot the results for $30, 35, 40, 45, 50, 55, 60$ hashing tables with various key lengths.

on modern CPUs, and are much faster to evaluate than the Euclidean distance between floating-point vectors. Moreover, the T trees can be simultaneously queried on a multi-core machine, which means we incur only a limited penalty for using several trees.

6.3.2 Uniform LSH

We also developed an ANN method inspired by the Hashing-based method of [36], which involves converting integer vectors into binary ones and randomly selecting and concatenating bits from them to generate multiple hashing keys. A query vector is then matched against the vectors in the buckets corresponding to its keys values by linear search. The smaller the lengths of the keys, the greater the size of the buckets becomes, which yields higher precision at the cost of increased computational time. This simple scheme performs well, as our experiments show. As for parc-trees, most of the operations are Hamming distance evaluations, which can be performed efficiently, and searches, which can be parallelized.

However, the random selection of coordinates may lead to unnecessary overhead, as some coordinates may be selected more frequently than others, whereas some of them may not be picked at all. This problem can be solved by increasing the number of keys, but to run fast, it is desirable to use as few keys as possible.

To resolve this dilemma, we optimize the keys so that the bits selected to generate

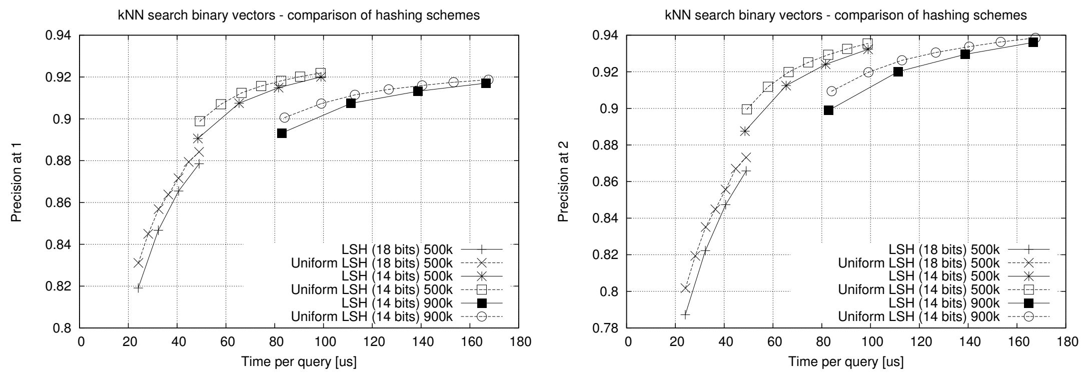


Figure 6.4: .
Comparison of precision for first (left) and second (right) positions for the original LSH and Uniform LSH with hashing keys optimized as explained in Section 6.3.2. We varied the number of keys, their lengths and the sizes of the datasets. While the improvement is limited, this demonstrates that the hashing keys can be optimized in Hamming spaces.

the keys are distributed more uniformly. This way, the keys generate more various partitioning of the database. More formally, we can define the keys K_i as sets containing the selected bits coordinates: $K_i = \{c_{ij} \in [1; L] \mid j \in [1; n]\}$ where L is the dimensionality of the binary vectors, n is the number of bits in a key. We also define N_k as the number of times a given coordinate is used in a key: $N_k = |\{c_{ij} = k \mid i \in [1; m] \text{ and } j \in [1; n]\}|$, where m is the number of keys. Then we optimize the keys to minimize

$$\min_{\{K_i\}} \sum_k (N_k - N)^2 \tag{6.2}$$

with $N = n \cdot m/L$, the ideal number of times a coordinate should be picked. To do this, we use a simple greedy algorithm that generates the keys one by one, by randomly selecting the bits among those which were used less often for the previous keys. We call this modification Uniform LSH as the distribution of the bits used in the keys is optimized to be more uniform and hence partition the dataset better.

We experimented with different numbers of keys and numbers of bits per key used. The results of those experiments are shown in Fig. 6.3. Computation times increase linearly with the number of keys, but the precision of the search also increases. Similarly, the lower the number of bits per key used, the bigger the data partitions and hence the longer the search time. However, since we perform a linear search within the selected data partition, shorter keys (and bigger data partitions) lead to performance improvement.

Overall, as it can be seen in Fig. 6.4, the resulting Uniform LSH algorithm improves performances over those of the original LSH.

6.4 Results

In this section, we first compare parc-trees and uniform LSH against the state-of-the-art ANN methods, namely kd-trees and HKM trees. To that end we use datasets of binary local feature descriptors created using LDAHash and BinBoost methods. We conclude this section by applying our algorithms to real-life application of aerial triangulation.

6.4.1 Evaluation Protocol

To evaluate our approach to solving the thick Voronoi boundary, we first create several datasets using two binary descriptors. We generate 128-bit binary LDAHash by binarizing the SIFT descriptors extracted from a set of additional Flickr images of Venice. Together with the dataset described in Section 6.2.1 we then have three datasets that contain 500k, 900k and 1.5M descriptors. Moreover, we use BinBoost$_1$-128t descriptor proposed in this thesis to obtain two other datasets of sizes 500k and 1M. The BinBoost descriptors are created from Liberty dataset using a set of weak learners trained on the Notre Dame dataset. One should note that those test datasets are larger than the ones used to evaluate the recent FLANN library [73] which mostly contained only 100k vectors. Furthermore, datasets of comparable sizes are frequently used for real-life applications, such as image-based 3D reconstruction.

To analyze the performances of various methods, we plot *precision* vs *query time* curves by setting the parameters of all algorithms so that the query time is approximately the same. To produce the different points in the plots, the number of hashing tables of LSH varied from 30 to 60, the number of parc-trees varied from 8 to 32, while for kd-trees and HKM we limited number of visited leaves. The results presented here are the average over three runs. The computation times were evaluated on a computer with two Intel Xeon E5620 2.4 GHz CPUs and 48 Gb RAM.

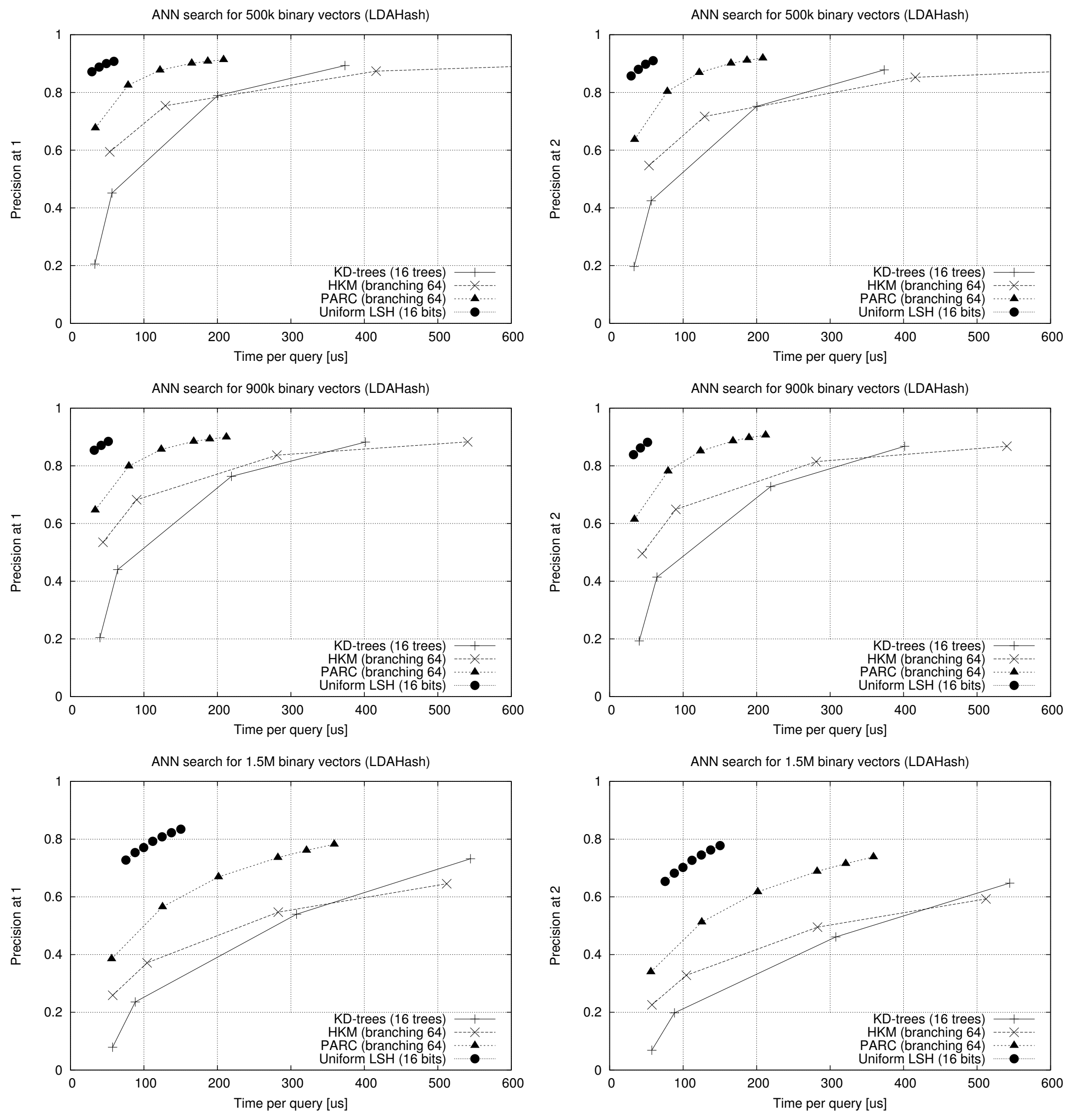


Figure 6.5: Comparison of precision for first and second positions for different ANN search algorithms on 500k, 900k and 1.5M binary LDAHash descriptors. Uniform LSH outperforms the parc-trees, which themselves outperform all the other state-of-the-art methods for all configurations.

6.4.2 Results for LDAHash descriptors

Fig. 6.5 presents the comparison of different ANN search algorithms applied to binary LDAHash descriptors. LSH outperforms all tree-based methods. Out of those, parc-trees remain the best. The speed-up of LSH and parc-trees over the other algorithms is especially visible for higher precision levels. For instance, for 500k dataset KD-trees needs approximately $300\mu s$ to reach the precision at second position equal to 0.85, whereas it takes parc-trees and Uniform LSH less than $100\mu s$ and $50\mu s$, respectively.

As the dataset size grows, it takes more time to find the nearest neighbors, but the relative ordering of the performances remains the same for all the methods: LSH performs the best, followed by the parc-trees. For the 1.5M dataset, LSH achieves a precision of 0.7 at first position about an order of magnitude faster than KD-trees and HKM.

6.4.3 Results for BinBoost Descriptors

Using the same evaluation protocol, as in the previous section, we then perform an additional set of experiments on 500k and 1M datasets of BinBoost$_1$-128 created from Liberty dataset. The results of the evaluation are shown in Fig. 6.6. Similarly to the case of LDAHash descriptors, also in this case parc-trees and Uniform-LSH outperform the other methods. Contrary to the results obtained using LDAHash descriptors, HKM performs worse than KD-trees on BinBoost descriptors, which may be explained by different distribution of the bits of BinBoost and LDAHash. As shown in Fig. 6.7, mean values of binary dimensions are much closer to 0.5 for LDAHash, than for BinBoost.

6.4.4 Memory Requirements

Finally, we perform a set of experiments to analyze the memory consumption of the ANN algorithms. To that end, we measure the memory allocated when searching through BinBoost descriptor datasets of size 500k and 1M. We report here the memory reserved for the structures and the stored data.

As we show in this chapter, randomization solves the thick Voronoi boundary, although it comes at a price of generating redundant data partitions. Fig. 6.8 shows the impact of this overhead on memory consumption. Not surprisingly, increasing number of parc-trees and LSH hashing tables leads to better search quality, but also higher memory allocation. As a matter of fact our unoptimized implementation quickly exceeds the memory reserved by FLANN implementation of both kd-trees and HKM.

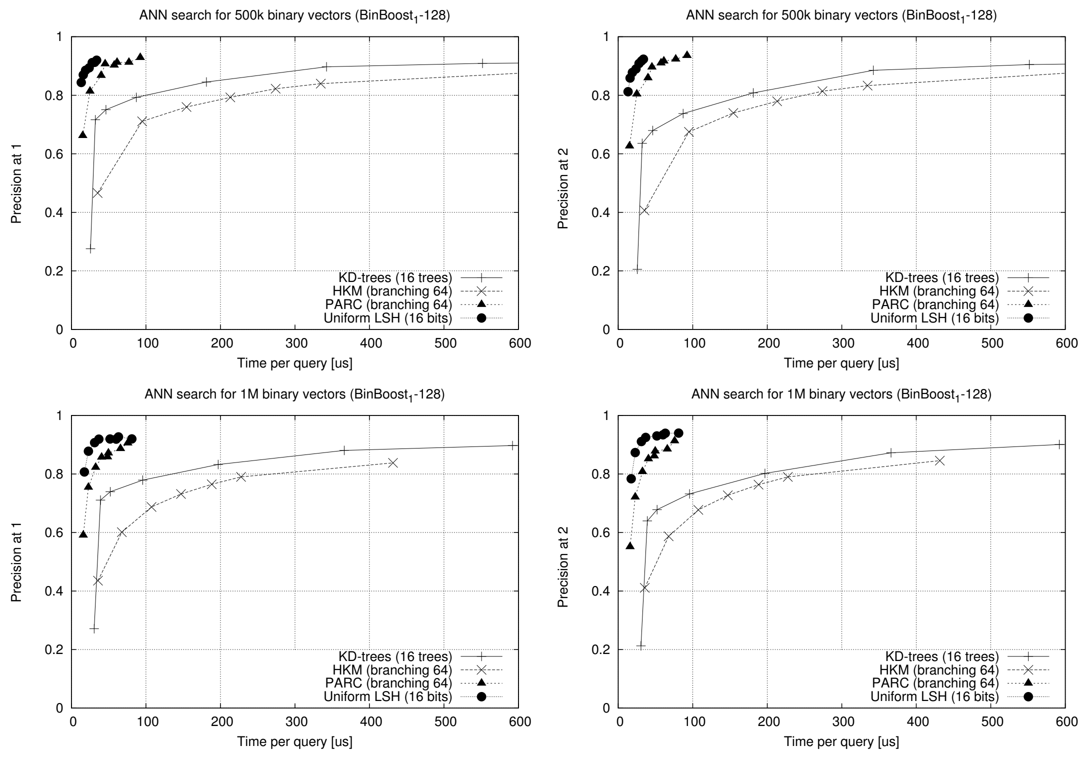


Figure 6.6: Comparison of precision for first and second positions for different ANN search algorithms on 500k and 1M binary BinBoost$_1$-128 descriptors. Uniform LSH outperforms the parc-trees, which themselves outperform all the other state-of-the-art methods for all configurations.

This problem has been addressed in the literature and an extension of LSH scheme was proposed under the name of Multiprobe LSH [63]. In short, Multiprobe LSH replicates query vectors, perturbs them with random noise and uses this newly created expanded set of queries to look for the approximate nearest neighbors. This allows us to reduce the number of generated hashing tables as we effectively increase the chances of finding the correct database vector by visiting more buckets. Fig. 6.8 shows that extending our Uniform-LSH algorithm with multi-probing leads to significant memory savings, while Fig. 6.9 proves that this comes at virtually no cost in terms of performance.

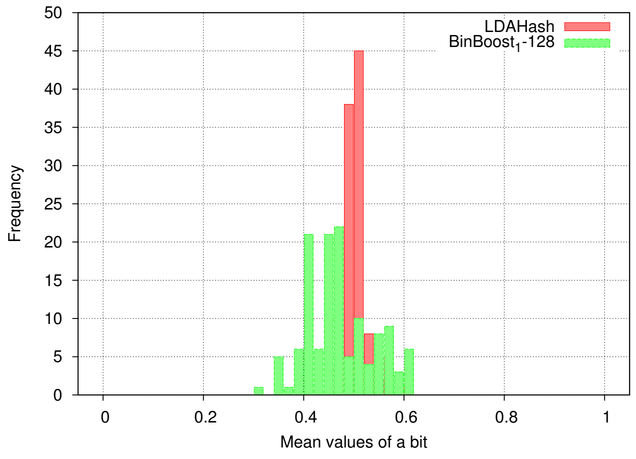


Figure 6.7: Histograms of means for binary dimensions of BinBoost$_1$-128 and LDAHash descriptors.

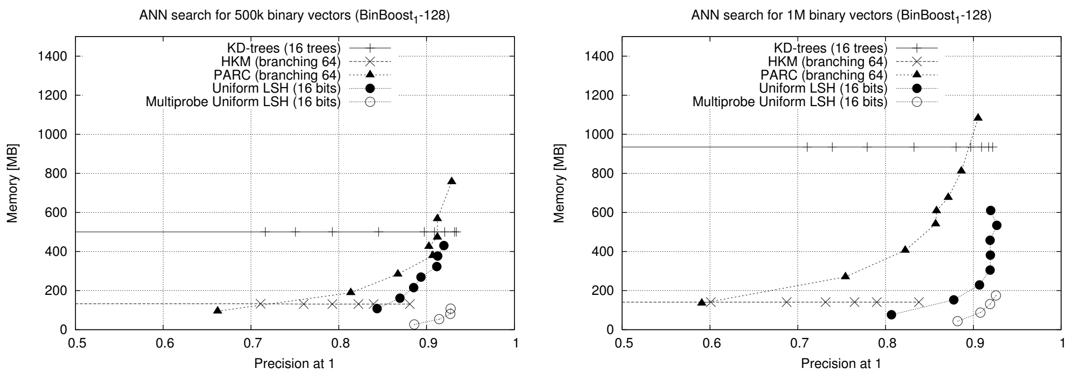


Figure 6.8: Comparison of memory used for different ANN search algorithms on 500k and 1M binary BinBoost$_1$-128 descriptors.

6.4.5 Aerial Triangulation

To verify our approach, we applied it to Aerial Triangulation. We extracted feature points from aerial images, and match them using the same binary descriptors as before. Matched points correspond to the same 3D points, and we use these matches to jointly optimize the 3D points and the camera parameters by bundle block adjustment [42].

We tested our binary search strategy on two datasets of large aerial images. The first dataset contains 25 high resolution (13824×7680) images of Marseilles[1], two of

[1]The images are available at `http://eurosdrbenchmarkofimagematching.ign.fr/` under "Bench-

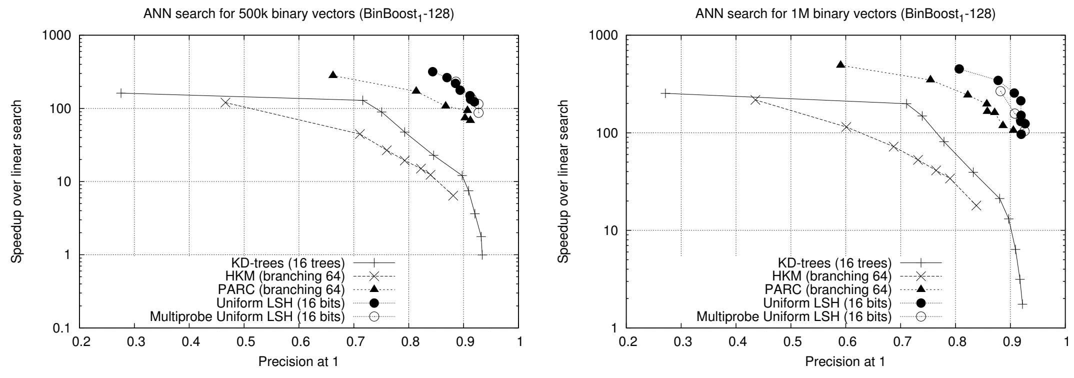


Figure 6.9: Comparison of speedup with respect to the linear search for different ANN search algorithms on 500k and 1M binary BinBoost$_1$-128 descriptors.

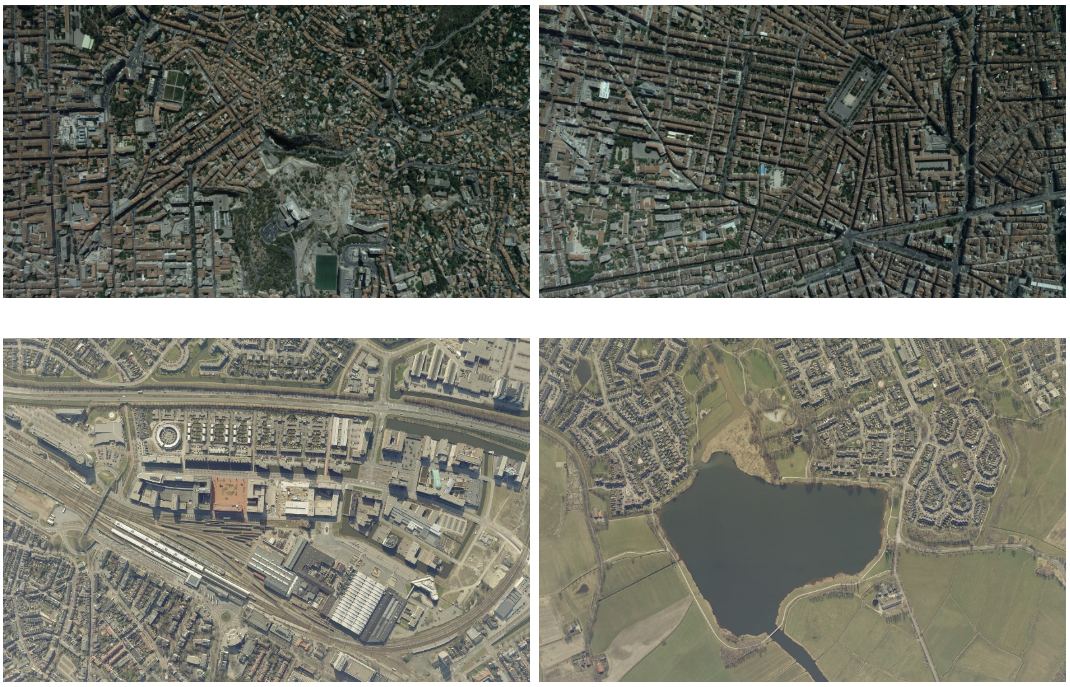


Figure 6.10: Top: Two out of the 25 13824×7680 pixels images from the Marseilles dataset. **Bottom:** Two out of the 68 11500×7500 pixels images from the Zwolle dataset.

which are shown in Fig. 6.10. The second dataset consists of 68 11500×7500 aerial images of the Dutch city of Zwolle.

Each image contains approximately 400k binary keypoints which makes exhaustive

marking of Image Matching Approaches for DSM computation" project.

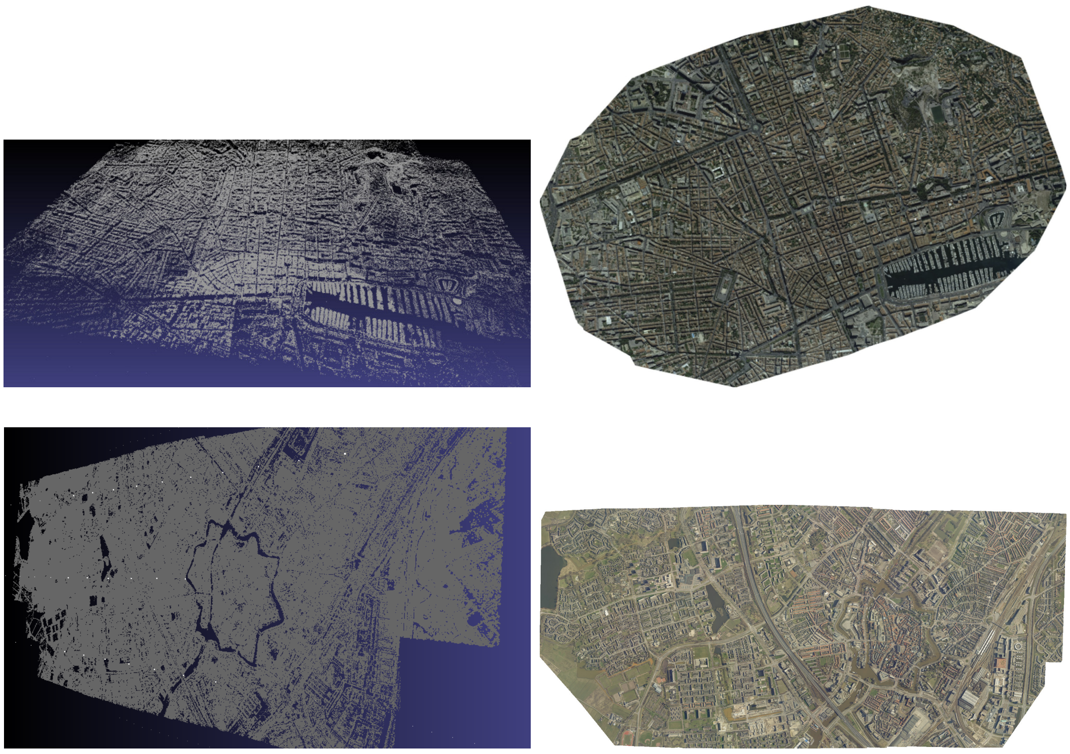


Figure 6.11: Top left: The aerial triangulation of 1.1M 3D points and **top right:** the generated ortho-image for the Marseilles dataset. **Bottom left:** The aerial triangulation of 2.1M 3D points and **bottom right:** the ortho-image made of 68 individual images (right) for the Zwolle dataset.

feature matching, even on binary vectors, excessively slow. Our approach reduces the matching time by a factor 20 over linear search with a 95% accuracy, which is consistent with the results reported in Section 5. The final aerial triangulations and the combined ortho-images for the Marseilles and Zwolle datasets are shown in Fig. 6.11.

6.5 Conclusion

In this chapter we showed that Voronoi diagrams in Hamming spaces have thick borders, which reduces the precision of many state-of-the-art ANN algorithms. We then proposed two techniques that rely on randomized data partitioning to overcome this problem and yield precisions that are comparable to those obtained using floating-point vectors at a fraction of the computational cost.

Part III

Final Words

CHAPTER
SEVEN

CONLUDING REMARKS

In this thesis, we presented two methods to construct robust binary local feature descriptors. We proposed to use machine learning techniques and learn the necessary descriptor invariance towards various image transformations directly from the data. We focused our efforts on binary descriptors, as they enable faster processing at lower memory footprint. The performances of previous binary descriptors, such as BRIEF or BRISK, tend to be inferior to those of the competing floating-point descriptors, such as SIFT or SURF. We showed that this performance loss can be mitigated by using data-driven approaches while preserving all the benefits of binary representations.

We first proposed an extremely compact and quick to compute binary descriptor, D-Brief, that is built by projecting an image patch to a more discriminative subspace and thresholding the resulting coordinates. Applying complex projections directly to the image patch can be computationally demanding, and hence we introduced an efficient method for learning discriminative projections so that they can be decomposed into a small number of simple filters, such as box or Gaussian filters. Our approach allows for the selection of the dictionary of filters that are used to construct the projections and number of elements, thus controlling the trade-off between computational complexity and the projection quality. The resulting 32-bit D-Brief descriptor outperforms its binary intensity-based competitors such as BRIEF, ORB or BRISK, while reducing the memory footprint over 8 times and decreasing the description-matching cycle time. It is therefore mainly designed for low-cost mobile platforms and real-time application with a significant emphasis on fast processing.

Although compact and very fast to compute and match, D-Brief does not provide the discriminative power and robustness that is required for multiple large-scale applications, such as 3D reconstruction or visual search. Hence, we extended our work on learning binary feature descriptors by proposing a more complex and powerful binary descriptor called BinBoost. We leveraged the *boosting-trick* to efficiently train a compact binary descriptor that is extremely robust to image transformations and viewpoint changes. Each bit of BinBoost is computed with a boosted binary hash function and we showed how to efficiently optimize different hash functions so that they complement each other, which is key to compactness and robustness. As we do not put any constraints on the weak learner configuration underlying each hash function, our general framework allows us to optimize the sampling patterns and encompasses the formulation of various hand-crafted descriptors. Hence, the resulting descriptor significantly outperforms the state-of-the-art binary and floating-point descriptors at a fraction of the matching time and memory footprint.

Since feature descriptors are typically used in a more complex systems where they have to be matched, we also investigated various approximate nearest neighbor search (ANN) methods and checked their applicability to binary vectors. Our findings showed that the state-of-the-art methods for approximate nearest neighbor search do not perform well and we attributed this behavior to the specificity of binary spaces, namely to the high concentration of the binary vectors along the Voronoi borders. This phenomenon turns to have a tremendous impact on the performance of the widely used ANN methods, such as hierarchical k-means or kd-trees. We therefore discussed the consequences of the thick Voronoi boundaries and gave a simple recipe to alleviate the above mentioned problems. More precisely, we introduced two algorithms which postulate to build the ANN search structures by randomizing data partitioning. The first one, called parc-trees, constructs a set of complimentary search trees by randomly selecting cluster centers from within the data points. The second method, improves the performance of the Local Sensitive Hashing (LSH) scheme by greedily optimizing over the selection of dimensions used to create hashing tables. In both cases, we showed that the performances of the proposed methods are similar to those based on floating-point number computations, but exploiting the characteristics of binary spaces allows us to significantly reduce the computational cost.

Future Research

The binary feature descriptors proposed in this thesis, although providing state-of-the-art performances, do not benefit from all the recent advances in machine learning and, therefore, it is expected that new algorithms to learn robust local feature descriptors will emerge. This becomes even more probable, when one takes into account the immense growth in the visual data that is being gathered by professionals, academics and regular people around the world. The massive amount of visual content is also prone to inspire more research on the topic of fast similarity computation and approximate nearest neighbor algorithms. We believe that the research presented in this thesis provides only the introduction to the topic of data-driven methods to learn and match binary feature descriptors and below we outline three paths that could extend this work.

Neural networks have been recently rediscovered and today they are considered one of the most well-known machine learning algorithms, mainly due to their general framework and impressive results [52, 53]. Moreover, Convolutional Neural Networks, a particular instance of Neural Networks, have successfully been used to train local feature descriptors [46, 83]. Nevertheless, the complexity of the learning framework limits the feasibility of the resulting descriptors, as the extraction time remains several orders of magnitude higher than those of the state-of-the-art descriptors available on the market. As a matter of fact, the initial learning objective of D-Brief, Eq. 3.2, was recently cast as a neural-network problem [66] which resulted in a significant performance improvement with respect to the LDA-type relaxed version of this objective. This leads to a conclusion that neural networks might help to re-learn image patch representations without referring to approximated and sub-optimal solutions. Moreover, the learning framework proposed for the BinBoost descriptor yields a close connection to a two-layer neural network and, hence, investigating the relation between neural networks and our boosting algorithm can potentially become an interesting research outlook.

Learning the feature detector can be seen as a natural extension of our work, as it complements the extraction pipeline of our learned descriptors. Nevertheless, the problem of training a repeatable detector remains very challenging due to the ambiguity of the keypoint concept as well as due to the differences between the appearances of the keypoints. Although several attempts to address this problem have been made, they typically focus on post-processing of the results of the initial detection for task-specific keypoint detection [110] or to decrease the computational burden [90, 91]. On the other hand, our research shows that feature descriptor can be learned directly from the intensity patches, without any intermediate step. A similar approach could potentially lead to a truly data-driven feature detection mechanism, although the variance in the training samples for this problem is significantly higher.

Quantizing binary feature descriptors constitutes yet another interesting research topic, as quantization of feature descriptors into visual words remains one of the most crucial elements of a typical visual search engine [105]. In the case of floating-point descriptors, hierarchical k-means has been successfully employed for feature quantisation [78]. As we showed in Chapter 6, however, hierarchical k-means does not perform well when clustering binary vectors. It is therefore necessary to develop other methods for binary descriptor quantization. A straight forward approach is by using only a subset of the dimensions to generate the indices of the quantization buckets. This approach resembles the locality sensitive hashing (LSH) framework discussed in Chapter 6, although with only one hashing table. Nonetheless, quantizing the descriptor into a single bucket can lead to significant performance losses as a single bit flip. *e.g.* due to noise, would result in an erroneous assignment. Hence, multiple assignment, also known as soft assignment [85], may present a good starting point to resolve this issue and can potentially lead to much faster visual search that is based on binary descriptors such as BinBoost.

REFERENCES

[1] Ahonen, T., Hadid, A. & Pietikïinen, M. (2006). Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**, 2037–2041. iv, 61, 62

[2] Alahi, A., Jacques, L., Boursier, Y. & Vandergheynst, P. (2011). Sparsity Driven People Localization with a Heterogeneous Network of Cameras. *Journal of Mathematical Imaging and Vision*. 53

[3] Alahi, A., Ortiz, R. & Vandergheynst, P. (2012). FREAK: Fast Retina Keypoint. In *Conference on Computer Vision and Pattern Recognition*. 1, 2, 11, 18, 22, 23, 65, 72

[4] Ali, K., Fleuret, F., Hasler, D. & Fua, P. (2012). A Real-Time Deformable Detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**, 225–239. 53

[5] Amit, Y., Geman, D. & Jedynak, B. (1998). Efficient Focusing and Face Detection. *Face Recognition: From Theory to Applications*. 95

[6] Andoni, A. & Indyk, P. (2008). Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Communications of the ACM*, **51**. 12, 93, 95

[7] Andreasson, H., Duckett, T. & Lilienthal, A. (2007). Mini-Slam: Minimalistic Visual SLAM in Large-Scale Environments Based on a New Interpretation of Image Similarity. In *International Conference on Robotics and Automation*. 5

[8] Arya, S., Mount, D., Netanyahu, N., Silverman, R. & Wu, A. (1998). An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. *Journal of the ACM*, **45**, 891–923. 93

[9] Bach, F., Jenatton, R., Mairal, J. & Obozienski, G. (2011). Optimization with Sparsity-Inducing Penalties. Tech. rep., INRIA. 32, 33

[10] Bay, H., Tuytelaars, T. & Van Gool, L. (2006). SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*. 1, 17, 18, 21, 45, 65, 93

[11] Bay, H., Ess, A., Tuytelaars, T. & Van Gool, L. (2008). SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, **10**, 346–359. 10, 11

[12] Beis, J. & Lowe, D. (1997). Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In *Conference on Computer Vision and Pattern Recognition*, 1000–1006. 93

[13] Bergamo, A., Torresani, L. & Fitzgibbon, A. (2011). Picodes: Learning a Compact Code for Novel-Category Recognition. In *Advances in Neural Information Processing Systems*, 2088–2096. 32

[14] Boix, X., Gygli, M., Roig, G. & Gool, L. (2013). Sparse Quantisation for Patch Description. In *Conference on Computer Vision and Pattern Recognition*. 11

[15] Bosch, S., Lacroix, S. & Caballero, F. (2006). Autonomous Detection of Safe Landing Areas for a UAV from Monocular Images. In *International Conference on Intelligent Robots and Systems*. 18

[16] Brown, M. & Lowe, D. (2007). Automatic Panoramic Image Stitching Using Invariant Features. *International Journal of Computer Vision*, **74**, 59–73. 18

[17] Brown, M., Hua, G. & Winder, S. (2011). Discriminative Learning of Local Image Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1, 2, 12, 25, 29, 30, 32, 33, 36, 38, 39, 44, 56, 65, 66, 70, 72, 93

[18] Cai, H., Mikolajczyk, K. & Matas, J. (2011). Learning Linear Discriminant Projections for Dimensionality Reduction of Image Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **33**, 338–352. 29

[19] Calonder, M. (2010). *Robust, High-Speed Interest Point Matching for Real-Time Applications*. Ph.D. thesis, EPFL. 11, 23

[20] Calonder, M., Lepetit, V., Strecha, C. & Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision*. 18, 22

[21] Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C. & Fua, P. (2012). BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**, 1281–1298. 1, 2, 11, 21, 23, 30, 44, 45, 55, 65, 72, 84

[22] Cao, Z., Yin, Q., Tang, X. & Sun, J. (2010). Face Recognition with Learning-Based Descriptor. In *Conference on Computer Vision and Pattern Recognition*. 63

[23] Cha, S.H. & Srihari, S. (2000). Nearest Neighbor Search Using Additive Binary Tree. In *Conference on Computer Vision and Pattern Recognition*. 97

[24] Chandrasekhar, V., Takacs, G., Chen, D., Tsai, S., Grzeszczuk, R. & Girod, B. (2009). CHoG: Compressed Histogram of Gradients a Low Bit-Rate Feature Descriptor. In *Conference on Computer Vision and Pattern Recognition*. 11

[25] Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y. & Tseng, B. (2010). Boosted Multi-Task Learning. *Machine Learning*. 25, 44

[26] Charikar, M. (2002). Similarity Estimation Techniques from Rounding Algorithms. In *STOC*, 380–388. 97

[27] Chatfield, K., Lempitsky, V., Vedaldi, A. & Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference*. 10

[28] Chli, M. & Davison, A. (2008). Active Matching. In *European Conference on Computer Vision*, 72–85. 5

[29] Davison, A.J., Reid, I., Molton, N. & Stasse, O. (2007). Monoslam: Real-Time Single Camera Slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**, 1052–1067. 5

[30] Dollar, P., Rabaud, V., Cottrell, G. & Belongie, S. (2005). Behavior Recognition via Sparse Spatio-Temporal Features. In *VS-PETS*, 65–72. 18

[31] Dollár, P., Tu, Z., Perona, P. & Belongie, S. (2009). Integral Channel Features. In *British Machine Vision Conference*. 25

[32] Edelman, S., Intrator, N. & Poggio, T. (1997). Complex Cells and Object Recognition. In *Advances in Neural Information Processing Systems*. 19

[33] Freund, Y. & Schapire, R. (1995). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *European Conference on Computational Learning Theory*, 23–37. 25, 46

[34] Friedman, J., Bentley, J. & Finkel, R. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, **3**. 95

[35] Fukunaga, K. & Narendra, P. (1975). A Branch and Bound Algorithm for Computing K-Nearest Neighbors. *IEEE Transactions on Computing*, **24**, 750–753. 93, 95, 96, 102

[36] Gionis, A., Indik, P. & Motwani, R. (1999). Similarity Search in High Dimensions via Hashing. In *International Conference on Very Large Databases*. 11, 94, 97, 98, 103

[37] Gong, Y. & Lazebnik, S. (2011). Iterative Quantization: A Procrustean Approach to Learning Binary Codes. In *Conference on Computer Vision and Pattern Recognition*. 12, 93

[38] Gong, Y., Lazebnik, S., Gordo, A. & Perronnin, F. (2012). Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 24, 44, 59, 65, 72

[39] Grauman, K. & Darrell, T. (2005). The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *International Conference on Computer Vision*, 1458–1465. 18, 25

[40] Grauman, K. & Fergus, R. (2013). Learning binary hash codes for large-scale image search. In *Machine Learning for Computer Vision*, 49–87. 95

[41] Harris, C. & Stephens, M. (1988). A Combined Corner and Edge Detector. In *Fourth Alvey Vision Conference*. 13, 14

[42] Hartley, R. & Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge University Press. 109

[43] Hastie, T., Tibshirani, R. & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer. 51

[44] Hirschmüller, H. & Scharstein, D. (2009). Evaluation of Stereo Matching Costs on Images with Radiometric Differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**, 1582–1599. 10

[45] Indyk, P. & Motwani, R. (1998). Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the 30th Symposium on Theory of Computing*, 604–613. 94

[46] Jahrer, M., Grabner, M. & Bischof, H. (2008). Learned Local Descriptors for Recognition and Matching. In *Computer Vision Winter Workshop*. 117

[47] Jain, P., Kulis, B., Davis, J. & Dhillon, I. (2012). Metric and Kernel Learning Using a Linear Transformation. *Journal of Machine Learning Research*. 24, 43

[48] Jégou, H., Douze, M. & Schmid, C. (2011). Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **33**, 117–128. 11

[49] Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P. & Schmid, C. (2012). Aggregating Local Image Descriptors into Compact Codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**, 1704–1716. 11, 25

[50] Ke, Y. & Sukthankar, R. (2000). PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Conference on Computer Vision and Pattern Recognition*, 111–119. 10, 24, 93

[51] Kulis, B. & Darrell, T. (2009). Learning to Hash with Binary Reconstructive Embeddings. In *Advances in Neural Information Processing Systems*, 1042–1050. 24, 93

[52] Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J. & Ng, A. (2012). Building High-Level Features Using Large Scale Unsupervised Learning. In *International Conference on Machine Learning*. 117

[53] LeCun, Y., Huang, F.J. & Bottou, L. (2004). Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *Conference on Computer Vision and Pattern Recognition*. 117

[54] Leutenegger, S., Chli, M. & Siegwart, R. (2011). BRISK: Binary Robust Invariant Scalable Keypoints. In *International Conference on Computer Vision*. 1, 2, 11, 18, 22, 23, 30, 44, 45, 53, 65, 72

[55] Lindeberg, T. (1993). Detecting Salient Blob-Like Image Structures and Their Scales with a Scale-Space Primal Sketch: A Method for Focus-Of-Attention. *International Journal of Computer Vision*, **11**, 134–141. 14

[56] Lindeberg, T. (1995). Direct Estimation of the Affine Image Deformations Using Visual Front-End Operations with Automatic Scale Selection. In *International Conference on Computer Vision*. 14

[57] Lindeberg, T. (1998). Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, **30**, 79–116. 13, 14, 17

[58] Liu, C., Yuen, J. & Torralba, A. (2011). SIFT Flow: Dense Correspondence across Scenes and its Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **33**. 10

[59] Liu, T., Moore, A., Gray, A. & Yang, K. (2004). An Investigation of Practical Approximate Nearest Neighbor Algorithm. In *Advances in Neural Information Processing Systems*. 93

[60] Liu, W., Wang, J., Ji, R., Jiang, Y.G. & Chang, S.F. (2012). Supervised Hashing with Kernels. In *Conference on Computer Vision and Pattern Recognition*. 24, 50

[61] Lowe, D. (1999). Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision*, 1150–1157. 13, 16

[62] Lowe, D. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, **20**, 91–110. 1, 2, 10, 11, 18, 20, 45, 65, 68, 93

[63] Lv, Q., Josephson, W., Wang, Z., Charikar, M. & Li, K. (2007). Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search. In *International Conference on Very Large Data Bases*. 108

[64] Malekesmaeili, M., Ward, R. & Fatourechi, M. (2012). A Fast Approximate Nearest Neighbor Search Algorithm in the Hamming Space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 44

[65] Marimon, D., Bonnin, A., Adamek, T. & Gimeno, R. (2010). Darts: Efficient Scale-Space Extraction of DAISY Keypoints. In *Conference on Computer Vision and Pattern Recognition*. 1

[66] Masci, J., Bronstein, M., Bronstein, A. & Schmidhuber, J. (2014). Multimodal Similarity-Preserving Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**. 43, 117

[67] Masci, J., Bronstein, M., Bronstein, A., Sprechmann, P. & Sapiro, G. (2014). Sparse Similarity-Preserving Hashing. In *International Conference on Learning Representations*. 43, 95

[68] Matas, J., Chum, O., Martin, U. & Pajdla, T. (2002). Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference*, 384–393. 13

[69] Mikolajczyk, K. & Schmid, C. (2002). An Affine Invariant Interest Point Detector. In *European Conference on Computer Vision*, 128–142. 14

[70] Mikolajczyk, K. & Schmid, C. (2004). Scale and Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, **60**, 63–86. 14, 16

[71] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T. & Van Gool, L. (2005). A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, **65**, 43–72. iv, 49, 65, 68, 71

[72] Miller, M., Rodriguez, M. & Cox, I. (2005). Audio Fingerprinting: Nearest Neighbor Search in High Dimensional Binary Spaces. *Journal of VLSI Signal Processing Systems*, **41**. 97

[73] Muja, M. & Lowe, D. (2009). Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision*. 2, 95, 99, 105

[74] Muja, M. & Lowe, D. (2012). Fast Matching of Binary Features. In *Conference on Computer and Robot Vision*. 2, 97

[75] Murase, H. & Nayar, S. (1995). Visual Learning and Recognition of 3D Objects from Appearance. *International Journal of Computer Vision*, **18**, 5–24. 9

[76] Neubeck, A. & Gool, L.V. (2006). Efficient Non-Maximum Suppression. In *International Conference on Pattern Recognition*. 15

[77] Neyshabur, B., Yadollahpour, P., Makarychev, Y., Salakhutdinov, R. & Srebro, N. (2013). The Power of Asymmetry in Binary Hashing. In *Advances in Neural Information Processing Systems*. 95

[78] Nister, D. & Stewenius, H. (2006). Scalable Recognition with a Vocabulary Tree. In *Conference on Computer Vision and Pattern Recognition*. vii, 2, 18, 65, 69, 82, 93, 95, 96, 102, 118

[79] Norouzi, M. & Fleet, D. (2011). Minimal Loss Hashing for Compact Binary Codes. In *International Conference on Machine Learning*. 24

[80] Norouzi, M., Punjani, A. & Fleet, D. (2012). Fast Search in Hamming Space with Multi-Index Hashing. In *Conference on Computer Vision and Pattern Recognition*. 44, 94, 95

[81] Nowak, E., Jurie, F. & Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision*. 10

[82] Oliva, A. & Torralba, A. (2001). Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision*, **42**, 145–175. 9

[83] Osendorfer, C., Bayer, J., Urban, S. & van der Smagt, P. (2013). Convolutional Neural Networks Learn Compact Local Image Descriptors. In *International Conference on Neural Information Processing*. 117

[84] Philbin, J., Chum, O., Isard, M., Sivic, J. & Zisserman, A. (2007). Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *Conference on Computer Vision and Pattern Recognition*. 4, 18

[85] Philbin, J., Chum, O., Isard, M., Sivic, J. & Zisserman, A. (2008). Lost in Quantisation: Improving Particular Object Retrieval in Large Scale Image Databases. In *Conference on Computer Vision and Pattern Recognition*. 88, 118

[86] Philbin, J., Isard, M., Sivic, J. & Zisserman, A. (2010). Descriptor Learning for Efficient Retrieval. In *European Conference on Computer Vision*. 24

[87] Raginsky, M. & Lazebnik, S. (2009). Locality-Sensitive Binary Codes from Shift-Invariant Kernels. In *Advances in Neural Information Processing Systems*. 24, 93

[88] Robertson, D. & Cipolla, R. (2004). An Image-Based System for Urban Navigation. In *British Machine Vision Conference*. 4

[89] Rosset, S., Zhu, J. & Hastie, T. (2004). Boosting as a Regularized Path to a Maximum Margin Classifier. *Journal of Machine Learning Research*. 25

[90] Rosten, E. & Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision*. 13, 15, 118

[91] Rosten, E., Porter, R. & Drummond, T. (2010). Faster and Better: A Machine Learning Approach to Corner Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**, 105–119. 83, 118

[92] Rublee, E., Rabaud, V., Konolidge, K. & Bradski, G. (2011). ORB: An Efficient Alternative to SIFT or SURF. In *International Conference on Computer Vision*. 1, 2, 11, 18, 22, 23, 30, 44, 45, 53, 65

[93] Saha, S. & Demoulin, V. (2012). An Efficient Binary Descriptor Based on Haar Features. In *International Conference on Image Processing*. 11

[94] Salakhutdinov, R. & Hinton, G. (2009). Semantic Hashing. *International Journal of Approximate Reasoning*, **50**. 24, 93, 95

[95] Schaffalitzky, F. & Zisserman, A. (2002). Multi-View Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps?". In *European Conference on Computer Vision*, 414–431. 10

[96] Schapire, R.E. & Singer, Y. (1999). Improved Boosting Algorithms Using Confidence Rated Predictions. *Machine Learning*, **37**, 297–336. 50

[97] Schmid, C. & Mohr, R. (1997). Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**, 530–534. 10

[98] Shakhnarovich, G. (2006). *Learning Task-Specific Similarity*. Ph.D. thesis, MIT. 2, 12, 43, 44, 46, 56, 65, 93

[99] Shen, S., Shi, W. & Liu, Y. (2009). Monocular 3D Tracking of Inextensible Deformable Surfaces Under L2-Norm. In *Asian Conference on Computer Vision*. 43

[100] Silpa-anan, C. & Hartley, R. (2008). Optimised kd-Trees for Fast Image Descriptor Matching. In *Conference on Computer Vision and Pattern Recognition*. 93, 95

[101] Simonyan, K. (2013). *Large-Scale Learning of Discriminative Image Representations*. Ph.D. thesis, University of Oxford. 12

[102] Simonyan, K., Zisserman, A. & Criminisi, A. (2011). Immediate Structured Visual Search for Medical Images. In *Conference on Medical Image Computing and Computer Assisted Intervention*. 4

[103] Simonyan, K., Vedaldi, A. & Zisserman, A. (2012). Descriptor Learning Using Convex Optimisation. In *European Conference on Computer Vision*. 1, 2, 12, 25, 44, 70

[104] Simonyan, K., Vedaldi, A. & Zisserman, A. (2014). Learning Local Feature Descriptors Using Convex Optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 12, 25, 70, 72

[105] Sivic, J. & Zisserman, A. (2003). Video Google: A Text Retrieval Approach to Object Matching in Videos. In *International Conference on Computer Vision*. 4, 70, 85, 118

[106] Smith, S.M. & Brady, J.M. (1995). Susan – A New Approach to Low Level Image Processing. Tech. rep., Oxford University. 15

[107] Snavely, N., Seitz, S. & Szeliski, R. (2006). Photo Tourism: Exploring Photo Collections in 3D. In *ACM SIGGRAPH*, 835–846. 18

[108] Stirling, J. (1764). *Methodus Differentialis [electronic Resource] : Sive Tractatus De Summatione Et Interpolatione Serierum Infinitarum.*. J. Whiston and B. White. 101

[109] Strasdat, H., Davison, A., Montiel, J. & Konolige, K. (2011). Double Window Optimisation for Constant Time Visual SLAM. In *International Conference on Computer Vision*. 11, 23

[110] Strecha, C., Lindner, A., Ali, K. & Fua, P. (2009). Training for Task Specific Keypoint Detection. In *DAGM Symposium on Pattern Recognition*. 118

[111] Strecha, C., Bronstein, A., Bronstein, M. & Fua, P. (2012). LDAHash: Improved Matching with Smaller Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**. 12, 24, 29, 30, 32, 44, 45, 59, 65, 72, 93, 95, 98

[112] Swain, M. & Ballard, D. (1991). Color Indexing. *International Journal of Computer Vision*, **7**, 11–32. 9

[113] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society*, **58**, 267–288. 33

[114] Tola, E., Lepetit, V. & Fua, P. (2010). Daisy: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**, 815–830. 1, 10, 23

[115] Torralba, A., Murphy, K., Freeman, W. & Rubin, M. (2003). Context-Based Vision System for Place and Object Recognition. In *International Conference on Computer Vision*. 12

[116] Torralba, A., Fergus, R. & Weiss, Y. (2008). Small Codes and Large Databases for Recognition. In *Conference on Computer Vision and Pattern Recognition*. 12, 29, 44

[117] Trulls, E., Kokkinos, I., Sanfeliu, A. & Moreno-Noguer, F. (2013). Dense Segmentation-Aware Descriptors. In *Conference on Computer Vision and Pattern Recognition*. 10

[118] Trzcinski, T. & Lepetit, V. (2012). Efficient Discriminative Projections for Compact Binary Descriptors. In *European Conference on Computer Vision*. 7

[119] Trzcinski, T., Christoudias, M., Lepetit, V. & Fua, P. (2012). Learning Image Descriptors with the Boosting-Trick. In *Advances in Neural Information Processing Systems*. 7

[120] Trzcinski, T., Lepetit, V. & Fua, P. (2012). Thick Boundaries in Binary Space and Their Influence on Nearest-Neighbor Search. *Pattern Recognition Letters*, **33**, 2173–2180. 7

[121] Trzcinski, T., Christoudias, M., Fua, P. & Lepetit, V. (2013). Boosting Binary Keypoint Descriptors. In *Conference on Computer Vision and Pattern Recognition*. 7

[122] Turk, M. & Pentland, A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, **3**, 71–86. 9

[123] Tuytelaars, T. & Mikolajczyk, K. (2008). Local Invariant Feature Detectors: A Survey. *Found. Trends. Comput. Graph. Vis.*, **3**, 177–280. iii, 12, 16

[124] Tuytelaars, T. & Van Gool, L. (2000). Wide Baseline Stereo Matching Based on Local, Affinely Invariant Regions. In *British Machine Vision Conference*, 412–422. 10

[125] Vedaldi, A. (2005). `http://www.vlfeat.org/~vedaldi/code/siftpp.html`. 65, 72

[126] Viola, P. & Jones, M. (2001). Rapid Object Detection Using a Boosted Cascade of Simple Features. In *Conference on Computer Vision and Pattern Recognition*. 17, 25

[127] Wang, J., Kumar, S. & Chang, S.F. (2010). Sequential Projection Learning for Hashing with Compact Codes. In *International Conference on Machine Learning*. 24, 50, 59

[128] Wang, J., Kumar, S. & S.-F.Chang (2010). Semi-Supervised Hashing for Scalable Image Retrieval. In *Conference on Computer Vision and Pattern Recognition*, 3424–3431. 12

[129] Wang, X., Doretto, G., Sebastian, T., Rittscher, J. & Tu, P. (2007). Shape and Appearance Context Modeling. In *International Conference on Computer Vision*. 10

[130] Wang, X., Zhang, C. & Zhang, Z. (2009). Boosted Multi-Task Learning for Face Verification with Applications to Web Image and Video Search. In *Conference on Computer Vision and Pattern Recognition*. 63

[131] Weiss, Y., Torralba, A. & Fergus, R. (2009). Spectral Hashing. *Advances in Neural Information Processing Systems*, **21**, 1753–1760. 11, 12, 24, 93, 95, 99

[132] Weiss, Y., Fergus, R. & Torralba, A. (2012). Multidimensional Spectral Hashing. In *European Conference on Computer Vision*. 24

[133] Yeo, C., Ahammad, P. & Ramchandran, K. (2011). Coding of Image Feature Descriptors for Distributed Rate-Efficient Visual Correspondences. *International Journal of Computer Vision*, **94**, 267–281. 11

[134] Yianilos, P. (1993). Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In *ACM-SIAM Symposium on Discrete Algorithms*. 93, 97

[135] Zervos, M. (2013). *Multi-Camera Face Detection and Recognition Applied to People Tracking*. Master's thesis, Ecole Polytechnique Fédérale de Lausanne. 61

[136] Zitnick, C. (2010). Binary Coherent Edge Descriptors. In *European Conference on Computer Vision*. 11

Tomasz Trzciński

Contact Information

EPFL / IC / ISIM / CVLab
Station 14
CH-1015 Lausanne
SWITZERLAND

E-mail: tomasz.piotr.trzcinski@gmail.com
WWW: `http://people.epfl.ch/tomasz.trzcinski`

Research Interests

Computer Vision: local feature descriptors, visual content search, augmented reality, large-scale vision-based localization, real-time image processing.
Machine Learning: boosting, neural networks, supervised dimensionality reduction.

Experience

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
Research Assistant in Computer Vision Lab **September 2010 – September 2014**
Developed and implemented various algorithms for computer vision and machine learning. Participated in teaching activities and international projects (*e.g.* with CERN), supervised graduate students, gave several talks internally and externally (Google Zürich, ETHZ), reviewed papers for major conferences (CVPR, ICCV, ECCV) and journals (PAMI, IJCV, TIP).

Google, Zürich, Switzerland
Software Engineer Intern in Knowledge Search Infrastructure Team **June – August 2013**
Researched and implemented efficient computer vision algorithms for a large-scale visual search engine: `http://images.google.com`.

Qualcomm, San Diego, CA, USA
System Engineering Trainee in Corporate Research & Development **April – July 2012**
Designed robust algorithms for machine vision and augmented reality applications. Organized and led several presentations on the topics of computer vision and machine learning.

Telefónica R&D, Barcelona, Spain
Research Assistant in Augmented Reality Group **January – July 2010**
Developed algorithms for precise geo-localisation using a visual search engine. Improved the quality of visual search by over 20%.
The commercial version of the search engine is available at: `http://catchoom.com`.

Opero, Gorzów Wielkopolski, Poland
IT Manager of Opero Online Backup Product, `www.opero.eu` **July 2009 – August 2010**
Responsible for project management in a start-up company: software development supervision, hardware configuration, market analysis and business strategy development.

Poznan Supercomputing and Networking Centre (PSNC), Poznań, Poland
PHP Programmer in Network Department **June – July 2008**
Developed a PHP Web application based on a XML database as a part of Geant 2 Educonf - European Commission project (`www.geant.net/service/educonf`).

Alopak, Ostrzeszów, Poland
Intern in Office Department **June – July 2005**
Responsible for research on development opportunities, contact with companies and customers.

Education

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
Ph.D. Computer Vision (CGPA 5.8/6) **2014**
- *Dissertation Topic*: Learning and matching binary local feature descriptors.

Universitat Politècnica de Catalunya, Barcelona, Spain
M.Sc. Research on Information and Communication Technologies (CGPA: 8.96/10) **2010**
- *Master Thesis*: Towards precise outdoor localisation based on image recognition.

Politecnico di Torino, Turin, Italy
M.Sc. Electronics Engineering – MERIT Master double degree (CGPA: 29.83/30) **2009**
B.Eng. Telecommunication (CGPA: 4.79/5) **2008**
- Studies done at *Poznan University of Technology*, Poznan, Poland

Poznan University of Economics, Poznan, Poland
B.A. Economics, completed 1st year before moving abroad (CGPA: 4.73/5) **2008**

Computer Skills

- **Languages**: C/C++, Java, C#, PHP, Bash scripting, LaTeX.
- **Software**: Matlab, Eclipse, LabView, database, spreadsheet and presentation software.
- **Operating Systems**: Unix/Linux, Mac OS X, Windows.
- **Others**: Worked using Agile methodology (Scrums) with GreenHopper.

Language Skills

- **English:** fluent. Certified by Cambridge CPE (2006), Cambridge CAE (2004).
- **Spanish:** fluent. Worked in a Spanish-speaking corporation.
- **French:** fluent. Worked and studied at a Swiss-French university.
- **German:** communicative. Took several extracurricular courses (1999-2005)
- **Italian:** communicative. Took a course at Politecnico di Torino (2008-2009).
- **Polish:** mother tongue.

Leadership Skills

Committee member of the *EPFL Consulting Society* 2012 – 2014
Organized various consulting events with companies (Accenture, Deloitte, McKinsey&Company).

PhD Student Mentor in the *EPFL Graduate Student Association* 2011 – 2014
Volunteered to help the incoming PhD students in their integration with the university.

Training

VentureLab – Venture challenge, Lausanne, Switzerland 2014

- Participated in a start-up oriented training program for young entrepreneurs organised by Swiss Commission for Technology and Innovation.

BCG Unlim-IT-ed – European IT Business Course, Milan, Italy 2013

- Invited by the Boston Consulting Group as one of 60 students from the top technical universities in Europe to take part in a business-oriented case study workshop.

Global Young Scientists Summit, Singapore 2013

- Invited as one of 280 young researchers worldwide to participate in discussion panels, lectures and brainstorming sessions with globally renowned scientific leaders - Nobel Prize, Fields Medal and Turing Award laureates.

Winter Augmented Reality Meeting, Graz, Austria 2013

- Invited to give a talk and participate in the interdisciplinary meeting of experts in augmented reality and related domains.

DTU Summer University in Telecommunication, Copenhagen, Denmark 2008

- Chosen as one of 20 students internationally to participate in several projects on cutting-edge technologies (optical fibres, mobile phone applications) at Technical University of Denmark.

Honors

Departmental fellowship of the School of Computer and Communication Sciences at EPFL.

Nokia scholarship for an exceptional performance during the DTU Summer University.

Academic scholarships of Poznan University of Technology (awarded repeatedly).

Publications

T. Trzcinski, M. Christoudias, V. Lepetit. *Learning Image Descriptors with Boosting*. Accepted to IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2014.

B. Fan, Q. Kong, **T. Trzcinski**, Z. Wang, C. Pan, P. Fua, Receptive Fields Selection for Binary Feature Description. Accepted to IEEE Transactions on Image Processing (TIP), 2014.

T. Trzcinski, M. Christoudias, P. Fua, V. Lepetit. *Boosting Binary Keypoint Descriptors*. Computer Vision and Pattern Recognition (CVPR), 2013.

T. Trzcinski, M. Christoudias, V. Lepetit, P. Fua. *Learning Image Descriptors with the Boosting-Trick*. Neural Information Processing Systems (NIPS), 2012.

T. Trzcinski, V. Lepetit. *Efficient Discriminative Projections for Compact Binary Descriptors*. European Conference on Computer Vision (ECCV), 2012.

T. Trzcinski, V. Lepetit, P. Fua. *Thick Boundaries in Binary Space and their Influence on Nearest-Neighbor Search*. Pattern Recognition Letters (PRL). Vol. 33, pp. 2173-2180, 2012.

M. Calonder, V. Lepetit, M. Özuysal, **T. Trzcinski**, C. Strecha, P. Fua. *BRIEF: Computing a local binary descriptor very fast*. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI). Vol. 34, Nr. 7, pp. 1281 - 1298, 2012.

D. Marimon, T. Adamek, A. Bonnin, **T. Trzcinski**. *Enhancing global positioning by image recognition*. International Symposium on Mixed and Augmented Reality (ISMAR), 2011.