Paul Joseph Hidalgo Flores

Compact features for mobile visual search

Descritores compactos para busca visual em dispositivos móveis

Campinas

2015

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Paul Joseph Hidalgo Flores

Compact features for mobile visual search
Descritores compactos para busca visual em dispositivos móveis

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de AE - Engenharia de Computação .

Supervisor: Prof. Dr. Eduardo Alves do Valle Junior.

Este exemplar corresponde à versão final da tese defendida pelo aluno Paul Joseph Hidalgo Flores, e orientada pelo Prof. Dr. Eduardo Alves do Valle Junior.

Campinas

2015

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

H53c

Hidalgo Flores, Paul Joseph, 1986-
Compact features for mobile visual search / Paul Joseph Hidalgo Flores. – Campinas, SP : [s.n.], 2015.

Orientador: Eduardo Alves do Valle Junior.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Descritor de imagem. 2. Dispositivo móvel. I. Valle, Eduardo,1978-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Descritores compactos para busca visual em dispositivos móveis
Palavras-chave em inglês:
Image descriptor
Mobile devices
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Eduardo Alves do Valle Junior [Orientador]
Otávio Augusto Bizetto Penatti
Christian Rodolfo Esteve Rothenberg
Data de defesa: 10-03-2015
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Paul Joseph Hidalgo Flores

Data da Defesa: 10 de março de 2015

Título da Tese: "Compact Features for Mobile Visual Search (Descritores Compactos para Busca Visual em Dispositivos Móveis)"

Prof. Dr. Eduardo Alves do Valle Junior (Presidente): Eduardo Valle

Dr. Otávio Augusto Bizetto Penatti: O. A. B. Penatti

Prof. Dr. Christian Rodolfo Esteve Rothenberg:

Abstract

Mobile Visual Search (MVS) applications became possible due to the computational power and multiple sensors on current mobile devices (smart-phones, tablets). In addition, the state-of-the-art in content based image retrieval (CBIR) has reached a maturity to perform these tasks efficiently. In this dissertation, we present a study of the major techniques in CBIR. An extensive study of the literature was performed, including the most common descriptors based on gradients and the recently proposed binary descriptors. As a result of comparative analysis between the main techniques in the context of MVS, we present the most appropriate alternatives to use in such applications.

Keywords: Visual Search, Descriptors, Compact descriptors, Mobile aplications.

Resumo

Aplicações de busca visual em aparelhos móveis (Mobile Visual Search – MVS) tornaram-se possíveis devido ao alto poder computacional e a multiplicidade de sensores dos novos dispositivos móveis (smart-phones, tablets). Além disso, o estado da arte em recuperação de informação multimídia baseada no conteúdo (Content Based Image Retrieval - CBIR) alcançou uma maturidade que permite realizar estas tarefas de forma eficiente. Nesta dissertação, apresentamos um estudo das principais técnicas em CBIR. Uma vasta investigação da literatura foi realizada, que inclui desde os descritores baseados em gradiente mais comuns aos mais recentes descritores binários e compactos. Como resultado da análise comparativa entre as principais técnicas no contexto de MVS, apresentamos as alternativas mais apropriadas para serem utilizadas em tais aplicações.

Palavras-chaves: Busca Visual, Descritores, Descritores compactos, Aplicações móveis.

Contents

Acknowledgements

First and foremost, I would like to thank Prof. Eduardo Valle for being a great mentor and advisor. He has been instrumental in guiding this work over the years. I have learnt a lot from Prof. Eduardo: how to pick fruitful research problems to work on, how to face new theoretical contents, and how to communicate research goals and results effectively, skills I hope to implement in my own research career. I am also grateful to him for his unwavering support during the highs and lows of my Msc and for encourage me to continue my research work. I would like to thank Sandra Avila, her priceless advice and insights have influenced this work at different points.

I would like to thank University of Campinas (UNICAMP) and Faculty of Electrical and Computing Engineering (FEEC) for giving me the opportunity to do my Msc studies. I am also grateful to Coordination for the Improvement of Higher Level -or Education- Personnel (CAPES) for their support during these 2 years of my Msc at UNICAMP. This work would not be possible without their funding. I would also like to thank the Reasoning for Complex Data Laboratory (RECOD Lab) and the Laboratory of Computer Engineering and Industrial Automation (LCA), groups with which I have collaborated closely.

I would like to thank my family for their encouragement and support. Finally, I would like to thank some of my friends at UNICAMP, who have made these years memorable: Roberto Medeiros, Augusto Cavalcante, Eliezer de Souza da Silva, Micael Carvalho, Jomara Bindá, Wallace Loos, Gaby Zemanate, Elvis Jara, José Hinostroza and Jackelyn Sedano.

List of Figures

List of Tables

1 Introduction

A few years ago, the thought of advanced mobile applications was preposterous, considering that mobile phones had low computational, and wireless networks were sparsely spread and had low bandwidth. Since then, thanks to technological advances on both devices and networks, mobile phones evolved into "smart" devices that are equipped with high-resolution cameras and displays, and powerful processing abilities. Nowadays, "smartphones" are a pervasive element of daily activities due to the large and constantly growing number of applications they support. According to comScore Report[1], in October 2013, 149.2 million people in the U.S. owned smartphones (62.5% mobile market penetration). A similar study, in June 2013, revealed that 84% of Brazilians between 16 and 64 years-old owned mobile phones, 36% of which corresponded to smartphones users[2]).

Mobile image-retrieval applications are a hotspot in academic research, at the junction of mobile devices and Content-Based Image Retrieval (CBIR). The idea is to enable a camera phone to obtain images and initiate search queries about objects in the 'clickable' visual proximity to the user. Those online applications are closely involved with augmented reality and can be used in different tasks e.g., product identification, game development, weather queries, location services, and queries about movies, compact disks (CDs), books or artworks. Some of the initial deployments like Google Goggles[3], Nokia Point and Find, Kooaba[4] and Amazon SnapTell (an early version of Flow[5]) are just a few examples of pioneering industrial products mentioned in a study on mobile image search [Nikolopoulos et al., 2011].

CBIR depends on the extraction of feature vectors from the images. The feature vectors are used to establish the similarity or dissimilarity between the images. Feature extraction schemes that can be roughly classified into "global descriptions" (one feature vector for the entire image), or "local descriptions" (several feature vectors per image, each describing a small patch, usually around an invariant location, like a point of interest). Local descriptors have been extensively used for CBIR systems and continue to be a very active area of com-

[1] `http://www.comscore.com/Insights/Press_Releases/2013/12/comScore_Reports_October_2013_US_Smartphone_Subscriber_Market_Share`

[2] Source, Nielsen: `http://www.nielsen.com/br/pt/insights/reports/2013/o-consumidor-movel.html`

[3] Google Goggles: `http://www.google.com/mobile/goggles/`

[4] Kooaba: `www.kooaba.com`

[5] Flow Amazon: `http://www.a9.com/whatwedo/mobile-technology/flow-powered-by-amazon/`

puter vision. Over the years, numerous local descriptors have been proposed in the literature and the highly discriminative SIFT [Lowe, 2004] proposed by D. Lowe remains the most commonly used. Other popular local descriptors include Gradient Location and Orientation Histogram (GLOH) proposed by Mikolajczyk and Schmid [2005] and Speeded Up Robust Features (SURF) proposed by Bay et al. [2008]. Winder and Brown [2007], Winder et al. [2009] and Mikolajczyk and Schmid [2005] provide a comprehensive analysis of several descriptors in a common framework. For mobile applications, however, there is a strong trade-off between the cost of computation in the (mobile) client and the bandwidth for transmitting either the image, either the descriptors extracted. Therefore, the use of compact local descriptors has been an active area of study. Those could be simply a compressed version of a traditional local descriptor, or a descriptor projected from scratch to have low bit-rate. The 60-b MPEG-7 trace-transform descriptor [Brasnett and Bober, 2007], the Compressed Histogram of Gradients (CHoG) [Chandrasekhar et al., 2012] and more recently binary descriptors like Binary Robust Indepent Elementary Features (BRIEFs) [Calonder et al., 2010], Oriented FAST and Rotated BRIEF (ORB) [Rublee et al., 2011], Binary Robust Invariant Scalable Keypoints (BRISK) [Leutenegger et al., 2011], Fast Retina Keypoint (FREAK) [Alahi et al., 2012] and Boosting Binary Keypoint Descriptors (BinBoost) [Trzcinski et al., 2013] are some examples of low bit-rate or compact descriptors. In this work, we are studying and comparing feature descriptors (with emphasis in compact representations), since the intended application will be in a mobile device.

To take advantage of the computational power of modern mobile phones the trend is to perform more computational processes on the client. Therefore, it might be reasonable to compute the local features on the mobile phone, provided that the time of computing those features on the client is not offset by the cost of transmitting them. This is far from obvious, because the local descriptors that are the most effective in terms of retrieval precision are often those that take the most space, sometimes more than the original image. The most common descriptor, SIFT, provides as an output a set of 128-dimensional feature vectors and conventionally each vector is stored as 1024 bits (8 bits/dimensions). Because often there are hundreds, or even thousands of vectors per image, typically the collective size of SIFT vectors from an image will be larger than the JPEG image itself. Therefore, for SIFT descriptor it is often better to transmit the image instead of transmitting the feature vectors. In contrast, if a compact descriptor is used, transmitting the feature vectors may become advantageous. Furthermore, using compact feature descriptors enable to represent an image using few amount of bits. Then, the computational load in matching becomes cheaper than using a traditional feature descriptor like SIFT.

1.1 Background

We present a briefly introduction of the basic concepts in Mobile Visual Search (MVS) and a discussion of the related literature. We should distinguish two application contexts: retrieval by image matching, and image classification. In the former, we are interested in a "target" image, object or scene. Retrieving a CD by its cover, a book by its cover, information about a shop by its façade, or about a monument in a city by its appearance, all those are examples of "target" retrieval. Image classification is a more complex problem because we are no longer interested only in an specific object or scene, but in a "class" of objects or scenes. This introduces an additional visual diversity, like all flowers of a given species, or all chairs. The scope of this dissertation is retrieval by image matching, but we present the classification problem for the sake of completeness.

1.1.1 Target Retrieval by Image Matching

A typical image matching and retrieval pipeline is shown in Figure 1, which is basically composed of an offline and an online stage. First, in the offline stage, the feature descriptors of data set images are computed and stored. Then, in the online stage, the local features are extracted from query image, this set of features is used to assess the similarity between query and database images. A short list of database images (candidates) is selected based on the number of features they have in common with the query image. Finally, geometric verification process rejects matches where the correspondence between local features is not consistent.

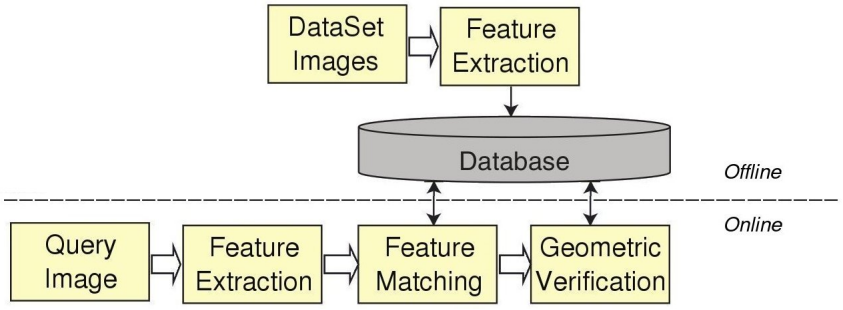


Figure 1 – A Pipeline for image retrieval (Reproduced from Girod et al. [2011]).

However, in mobile visual search context, the retrieval framework adopt a client–server approach (Figure 2) and divide the computational load between server and mobile client instead of perform all processes on server.

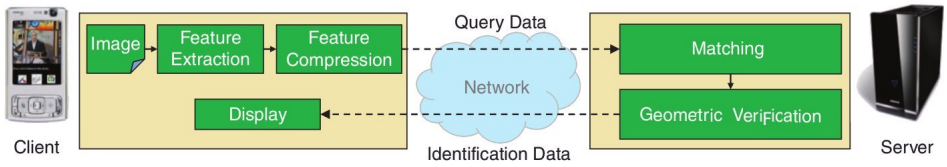


Figure 2 – Client/Server approach for mobile image search (Reproduced from Girod et al. [2011]).

The final step in retrieval pipeline, as shown in Figures 1 and 2, is geometric verification (GV). In this stage, the location information of features are used to establish geometric correspondence between query image and one database images. To estimate this geometric correspondence are used robust regression techniques such as Hough transform [Lowe, 2004] or RANSAC [Fischler and Bolles, 1981], the latter being the dominant approach. This process tend to be computationally expensive, therefore the list of candidates is limited to a small number of database images.

1.1.2 Image Classification

For classification, the "Bag of Features" (BoF) or "Bag of Visual Words" (BoVW) is one of the most successful approaches, mainly for its capability to perform a very fast retrieval operation. The BoF was presented by Sivic and Zisserman [2003] as an idea borrowed from text retrieval (where a document can be represented by a vector of word frequencies) by making an analogy between the local descriptors and the words in a document. Either in text or image retrieval that approach ignores the structure of the document (order of the textual words, position of the visual words).

The "bag" is a single feature vector built from the many local features. It can be used, during the training phase, to build a model from the training samples. This model can then be used for retrieval purposes, with the bag extracted from the query image. Support Vector Machines (SVM) are the most common classifier used with BoVW, since they are robust to the high feature vector dimensionalities that the BoVW model usually implies.

1.1.3 Feature Extraction

In CBIR a feature is a relevant piece of information of an image and can be classified into global and local. In global features, the whole image is processed and all spatial information about color or texture is destroyed. This lack of spatial information generate a poor discrim-

inating power in target retrieval tasks. On the other hand, local features aim to conserve spatial information and consequently are useful in target retrieval applications. Since the spatial information is essential for visual search applications, hereafter, we only discuss local feature extractors.

Local feature extraction typically involves two main steps: feature detection and feature description. Feature detection or interest point detection consists in identifying interest points in the image (*e.g.* edges, corners), these interest points must be repeatable under transformations (*e.g.* scale changes, viewpoint changes and rotation), illumination variations and blur. Feature description consists in computing a discriminative feature vector on each normalized patch, where image patches are regions centered in the interest points.

1.2 Contributions

The major contributions in this dissertation are as follows:

- Comprehensive study of feature extractors: We perform an extensive historical review of feature extractors, this allow us to visualize the evolution of such methods over the years and emphasize the increasing level of computational efficiency. We make this review as exhaustive as possible in the effort to put together the state of the art and the current developments, this was an special motivation due to the fact that existing surveys do not include the most recent feature extractors mainly because of the temporal gap between their date of publication.
- Experimental evaluation using a common framework: We perform an experimental comparison of feature detectors and feature descriptors, In contrast to previous evaluations, we perform this evaluation using the Stanford Mobile Visual Search Data Set: a data set composed of images obtained from actual camera phones. This allow us to evaluate the efficiency and effectiveness of such methods in a context of mobile applications.

1.3 Organization

The dissertation is organized as follows:

- In Chapter 2, we present a general review of the state of the art (SoA) in content-based image retrieval (CBIR). This includes feature detection, feature description and image matching. We emphasize on the topics relevant to mobile visual search.

- In Chapter 3, we describe the experimental procedure to evaluate the feature detectors and feature descriptors. We describe the resources employed (datasets, implementations, tests) and discuss the results. This experimental chapter aims to find the best feature extractor for mobile visual applications.

- Finally, in Chapter 4, we summarize the results and draw conclusions. We discuss open questions and suggest topics for future work.

2 State of the Art

In this chapter we present a review of the literature in local feature extractors which is one of the most important concepts in Content Based Image Retrieval (CBIR). A feature extractor creates from the image a set of feature vectors, which is a mathematical representation about the content of the image and is designed to be discriminant and invariant over several transformations, i.e., the notion of similarity or dissimilarity between images are preserved by their feature vectors and will be transferred to the feature space. The procedure of the feature extractor can be split into two steps clearly distinguishable: feature detection, which is studied in the section 2.1; and feature description, presented in the section 2.2. In both cases, detection and description, we present a general review of the most remarkable contributions with an emphasis in the computational efficiency, aiming at identifying the most suitable detector-descriptor for Mobile Visual Search applications.

2.1 Feature Detectors

Local *feature detection* goes back to 1954, when Attneave [1954] first observed that "information is concentrated along contours and is further concentrated at those points on a contour at which its direction changes most rapidly (*e.g.* at peaks of curvature)". Since then, numerous advances were made in the effort to interpret images as local features and the literature on feature detection became extensive. Our aim here is to review some of the most important feature detectors proposed over the years and not to provide an exhaustive survey. For the most comprehensive review available, the reader is referred to the work of Tuytelaars and Mikolajczyk [2008] or to the work of Gauglitz et al. [2011] which includes recent contributions in speed-up for feature detectors.

The most important property expected from a good local feature is *repeatability*, a notion introduced by Schmid et al. [2000]. Repeteatability (explained in detail in Section 3.1.1) measures the fraction of features detected in an object or scene under a specific viewing condition that are also detected on another different viewing condition.

Feature detectors can be classified using multiple criteria; here we adopt a classification based upon the type of feature extracted from the image (corner, blob or region) used in [Tuytelaars and Mikolajczyk, 2008]. The most important techniques from each family are presented below.

2.1.1 Corner Detectors

Corners are points in the image where edges change direction, i.e., points of high curvature in edges [Attneave, 1954]. Corner detectors may be derivative-based, like the Harris detector [Harris and Stephens, 1988], or morphological-based, like the SUSAN detector [Smith and Brady, 1997]. Basic detectors are invariant to translation and rotation. More advanced extensions will seek invariance to scale changes (e.g., Harris–Laplace, [Mikolajczyk and Schmid, 2004]), and most generally to affine transformations (Harris–Affine, [Mikolajczyk and Schmid, 2004]).

- Harris Detector

  Proposed by Harris and Stephens [1988], this detector is based on the second-moment matrix, or auto-correlation matrix M, defined below. Rotation invariance is provided by an eigenvalue analysis: a corner happens when both eigenvalues have large values.

  More formally, considering $I(x, y)$ as the grayscale image function at point (x, y), the Harris detector considers the variation of intensity C on x, y when shifted by $(\Delta x, \Delta y)$ (over an averaging window $w(x, y)$ centered on x, y):

  $$C(\Delta x, \Delta y) = \textstyle\sum_{x,y} w(x, y)[I(x + \Delta x, y + \Delta y) - I(x, y)]^2.$$

  Using first-order Taylor expansion in the shifted image function $I(x+\Delta x, y+\Delta y)$ this average intensity variation can be expressed in a matrix form as:

  $$C(\Delta x, \Delta y) = \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \underbrace{(\sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix})}_{M} \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}.$$

  Where I_x and I_y are respectively the partial derivatives of the image along the x and y variables.

  In order to save computation, the *eigenvalues* (λ_1, λ_2) are not computed directly, instead, a "cornerness" measure H compares the determinant and the trace of the matrix, the eigenvalues are big when the difference between those values is large.

  $$H = \det(M) - k\ (\text{trace}(M))^2.$$

  Where $\det(M) = \lambda_1\lambda_2$, $\text{trace}(M) = \lambda_1 + \lambda_2$ and $k = 0.04$ (commonly used value).

  The Harris detector is derivative, but is also integrative, due to the averaging window w (the sum in both equations correspond to integrative operations over that window).

Therefore, a hidden parameter of the technique is the choice and width of the window. For several theoretical and practical reasons, a Gaussian window is often employed (e.g., a Gaussian window avoids introducing artifacts of its own [Lindeberg, 1994]). The choice of the variance of the Gaussian window becomes a hidden parameter, related to the scale of the detected corner. Because of that, the Harris detector lacks scale invariance. This problem is addressed by more sophisticated detectors presented below.

- SUSAN detector

  The Smallest Univalue Segment Assimilating Nucleus (SUSAN) was proposed by Smith and Brady [1997]. This descriptor considers a circular area of preestablished radius around each pixel in the image, and sets the intensity value of center pixel (nucleus) as reference. Then, the pixel intensities inside the circular area are compared to the nucleus, and classified as "similar" or "different". Finally, those nucleus whose circular regions are $\sim 25\%$ similar to them are considered corners (Figure 3a).

  The idea of computing corners by comparing pixels instead of computing complex derivative/integrative filters inspired fast feature detectors that will be explained in Section 2.1.3.4.

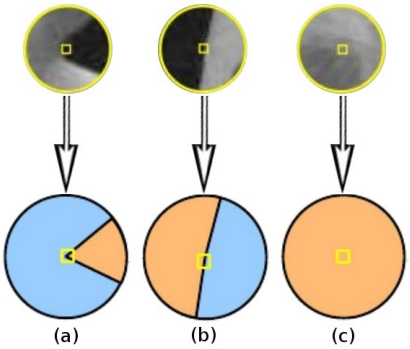


Figure 3 – SUSAN corner detector examples: "similar" (orange) and "dissimilar" (blue) regions. A ratio of $\sim 25\%$ dissimilarity represents corners (a); for near edges, this ratio is $\sim 50\%$ (b); in homogeneous regions almost the entire circular region has a similar intensity (c).

- MIC detector

  The Minimum Intensity Change (MIC) was proposed by Trajković and Hedley [1998]. For a candidate point, MIC makes a number of checks "k" in a circle around it, to determine whether that candidate is a corner or not. Each check takes two points in a diameter lying accross the circle and verify whether their values are similar of different. In a corner, all checks must reveal different values.

More formally. MIC defines the Corner Response Function (CRF):

$$CRF_N = \min_k ((f_p - f_N)^2 + (f_{p'} - f_N)^2).$$

Where N is the central point, f_N is the image intensity at point N, f_p and $f_{p'}$ are the image intensity values at either end of a diameter line across a pre-established discrete circle or Bresenham circle.

A corner point will present large CRF response, which means, for all diameter lines, the response of the intensity value comparison are large. A non-corner point will present a low CRF response, which imply that at least for one diameter of the Bresenham circle the response of the intensity value comparison is low. Then, this point belongs to an uniform region or an edge.

- Harris–Laplace / Affine

  These extensions of Harris detector were developed by Mikolajczyk and Schmid [2004]. In order to make feature detection scale-invariant, a sophisticated *scale–space* theory was developed [Lindeberg, 1994].

  Intuitively, the theory of scale–space establishes that the scale of structures in the image was intimately related to the resolution at which those structures first become apparent. This allows to detect the characteristic scale of different points in the image, by applying progressively stronger blurs and detecting when the point change the most (the reversal, in time, of the appearance of the structure of that point).

  Gaussian filter is often used for the blurring. There are strong theoretical arguments for this, e.g. the Gaussian is the maximum entropy distribution for a given variance (width), and thus tends to add the least extraneous information, in the form of artifacts. The theory of scale–space provides other compelling reasons [Lindeberg, 1994]. Because we will be usually looking for points of maximum change in appearance, derivatives (first and second) of the Gaussian are often employed.

  The Harris–Laplace detector is a scale-invariant version of Harris corner detector. In the Harris operator, the scale is determined by the variance of the Gaussian window used in the integration. This technique performs a multi-scale point detection using several (previously chosen) values for the variance of the Gaussian, and then uses the extrema of the Laplacian-of-Gaussian to pick the characteristic scale of the point.

  For a long time, an affine-invariant detector with correspondence to the scale–space theory was sought, that could be used as a basis for designing feature extractors invariant under arbitrary affine geometrical transformations (rotations, scales and translations obviously; but also local perspective changes). Harris–Affine is the affine-invariant

version of Harris corner detector. It takes an initial region detected using the Harris–Laplace detector, and, in an iterative process, uses the second moment matrix to estimate the affine shape of this region. The affine region is normalized into a circular region and in the normalized region the location and scale are re-detected. The iteration stops when the eigenvalues of the second moment matrix for the new point are equal (i.e., a circular region).

Affine-invariance, however, proved computationally very expensive, and ultimately the similarity-invariance (rotations, scales and translations) proved sufficient for most practical purposes, since, in practice, scale-invariant detectors tend to be also robust to small baseline (perspective) changes.

2.1.2 Blob Detectors

Blob detection typically provides complementary features to the corner detection, detecting stable "peaks" and "troughs" in the image. Often the "blob" is represented by an oriented circle (similarity-invariant extractors) or oriented ellipse (affine-invariant extractors).

We also consider in this category feature detectors without shape restrictions also known as region detectors. Although region detectors provide features with no regular shape, the first and second moments of the shape can be used to provide an ellipse instead. Therefore, all region detectors can be used as oriented-ellipse detectors (but not vice-versa).

- Hessian Detector

  Hessian detector, as explained in Tuytelaars and Mikolajczyk [2008, Section 4.1], is a detector based on the matrix of second derivatives. This detector searches for image locations that exhibit strong derivatives in two orthogonal directions. If we consider $I(x)$ as the image function at point (x), the Hessian matrix will be:

$$H = \begin{bmatrix} I_{xx}(x, \sigma_D) & I_{xy}(x, \sigma_D) \\ I_{xy}(x, \sigma_D) & I_{yy}(x, \sigma_D) \end{bmatrix}.$$

  Where $I(x, \sigma_D)$ is the convolution of the image with the Gaussian $g(x, \sigma_D)$ defined as: $I(x, \sigma_D) = g(\sigma_D) * I(x)$. Then, $I_{xx}(x, \sigma_D), I_{xy}(x, \sigma_D)$ and $I_{yy}(x, \sigma_D)$ are the second-order partial derivatives of an image smoothed by the Gaussian kernel.

  The elements of a Hessian matrix encode the shape information of a region and capture important properties of local image structures. The local scale–space extremas of the determinant of the Hessian matrix (DoH) and the trace of the Hessian matrix

(*Laplacian*) provide a differential blob detector. Similar to the Harris detector, the Hessian detector has an integrative component given by the Gaussian kernel, and a hidden scale parameter given by the variance σ_D^2 of that kernel. This detector can also be made scale-invariant using the same scale–space apparatus used for the Harris detector.

- Hessian–Laplace / Affine

  These extensions of Hessian detector were also developed by Mikolajczyk and Schmid [2004] and are quite analogous to their Harris-based counterparts, explained in Section 2.1.1. That turns Hessian–Laplace/Affine detector into viewpoint invariant blob detectors.

- Salient Regions

  Proposed by Kadir and Brady [2001], this method focuses on visual saliency, the idea that certain parts of a scene are pre-attentively distinctive in the Human Visual System (HMS). Saliency is defined as local complexity or unpredictability, and implies rarity. Then, saliency becomes discriminative and can be measured by the entropy of the probability distribution H within a local image region. The local maxima of entropy are recorded as candidates and in order to localize the features over scales these maxima are weighted by some measure of the self-dissimilarity in scale–space W. Then, the saliency is computed as $Y = WH$. Finally, candidates are sorted by their saliency Y and the top k-ranked are selected. This method is considered as a blob detector because the weighting step tends to give preference to blob-like structures in the image, but could also be considered a region detector.

- Maximally Stable Extremal Regions (MSER)

  Proposed by Matas et al. [2004], MSER regions are defined by an extremal property of the intensity function within the region and beyond its border. Formally, a region of the image is called extremal if setting an appropriate threshold all pixels inside the region have either higher (maximal extremal regions) or lower (minimal extremal regions) intensity than pixels outside the boundary.

  More intuitively, looking at the image as a 3D profile (the coordinates x, y as the plane, and the intensity as the elevation), imagine a flooding procedure similar to a watershed transform [Roerdink and Meijster, 2000] in which we progressively plunge the image profile in deeper and deeper water. The low-intensity areas of the image will be flooded first, forming "lakes", and the high-intensity areas will form "islands". At different steps of the flooding, some lakes and islands will suffer strong changes (lakes growing, islands shrinking) while other will change very little. The MSER regions are

those going through the *least* change during the flooding, i.e., for each lake or island, those moments where their area is very stable.

MSER provides an arbitrary region as a feature, that can be replaced by a ellipse using the first and second moments of the shape, if desired.

- Intensity-based regions (IBR)

  Proposed by Tuytelaars and Van Gool [2000] this method is directly based on the analysis of the image intensity. The procedure of this detector starts detecting local intensity extrema at multiple scales and then explores his neighborhood in a radial way as showed in Figure 4. Given a local extrema, the function presented below $f(t)$ is evaluated along each ray. For each ray, the function $f(t)$ has a point (along the ray) which maximizes its value. These points are linked to enclose an irregularly-shaped region. Analogous to MSER, this region is replaced by an ellipse that has the same first and second shape moments. Finally, the area of the ellipse is doubled to provide a higher distinctive power. This final region (ellipse) determines the region of interest and is invariant to affine deformations.

$$f(t) = \frac{|I(t) - I_0|}{max(\frac{\int_0^t |I(t) - I_0| dt}{t}, d)}.$$

  Where t is an arbitrary parameter along the ray, $I(t)$ is the intensity at position t, I_0 is the intensity value at the extrema and d is a small number added to prevent a division by zero.

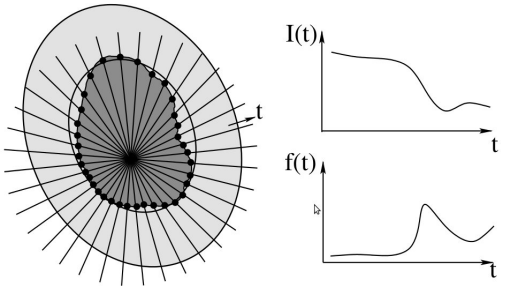


Figure 4 – Construction of intensity-based regions. The image presents a local extrema, its rays and the irregularly-shaped region which is obtained by joining the values that maximize $f(t)$ for each ray. (Reproduced from Tuytelaars and Van Gool [2000]).

The feature detectors studied until this point (2004) have constructed a theoretical apparatus that in conjunction with some intelligent simplifications/approximations will lead to a new set of computationally efficient feature detectors. In the next section, called optimized implementations, we will study the most successful computationally efficient feature detectors.

2.1.3 Optimized Implementations

In this section we will describe feature detectors designed to avoid computationally expensive processes (*e.g* computation of derivatives, second moment matrices, or the entropy of salient regions). The Scale-Invariant Feature Transform (SIFT) [Lowe, 2004] explores the Difference-of-Gaussians (DoG) to approximate the Laplacian-of-Gaussian (LoG), in Speeded Up Robust Features (SURF) [Bay et al., 2008] a rough approximation of Hessian matrix (Fast-Hessian) is obtained using integral images. CenSure detector [Agrawal et al., 2008] approximates the Laplacian also using integral image and square filters.

Finally, we describe another class of feature detectors, that further emphasize computational efficiency, sacrificing repeatability and invariance in the name of extremely streamlined implementation. The FAST detector [Rosten and Drummond, 2006], and its several proposed improvements.

2.1.3.1 Difference-of-Gaussians (DoG)

SIFT [Lowe, 2004] proposes a detector that is a streamlined feature detector based on scale–spaces. Instead of using the expensive Laplacian-of-Gaussian filters, SIFT employs Difference-of-Gaussians (DoG) in a scheme that *halves the image resolution whenever the standard deviation of the Gaussian doubles*. That ingenious scheme saves a lot of computation by avoiding the large convolution masks associated to large-variance Gaussian kernels. The whole process is illustrated in Figure 5: the image is smoothed using progressively stronger (larger variance) Gaussian filters; then, the difference between each pair of smoothed neighbors is computed (DoG); finally, the local maxima or minima is obtained in the scale–space comparing each sample point with its 26 neighbors in the 3×3 regions at the current and adjacent scales.

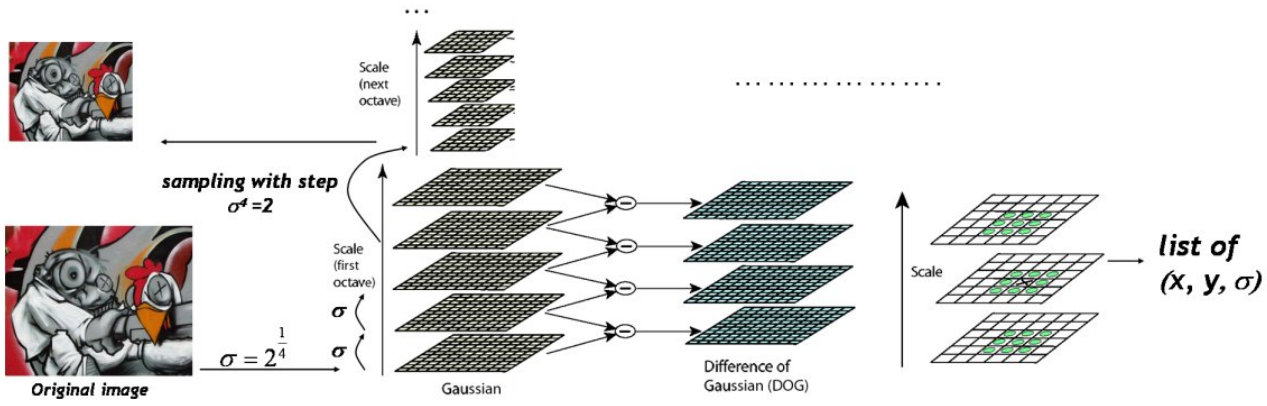


Figure 5 – Overview of DoG detection scheme. From left to right: image sampling using a Gaussian convolution mask, the Difference-of-Gaussians and the comparison of local extrema in scale–space (Reproduced from Tuytelaars and Van Gool [2000]).

2.1.3.2 Fast-Hessian

The scale-invariant feature detector Fast-Hessian, also known as SURF detector, was proposed by Bay et al. [2008]. The Fast-Hessian approach for interest point detection approximates the Hessian matrix using a set of box-type filters. This lends to the use of integral images, which reduces the computation time drastically. The calculation time of an integral image is independent of the size of the rectangular area, it allows to apply box filters of any size at exactly the same speed directly on the original image and even in parallel. Therefore, the scale-space is analyzed by up-scaling the filter size rather than iteratively reducing the image size.

- Integral images

  Integral images allow fast computation of box-type convolution filters. An integral image at a location $X = (x, y)$ is expressed as $I_{\sum}(x)$ and represents the sum of all pixels in the input image I within a rectangular region formed by the origin (O) and X.

$$I_{\sum}(x) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j).$$

  Using this expression, we can calculate the sum of the intensities inside any rectangular region in the image only with three additions as shown in Figure 6.

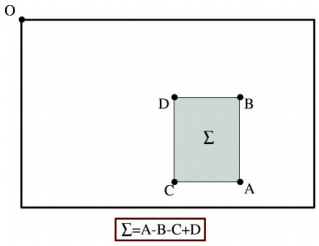
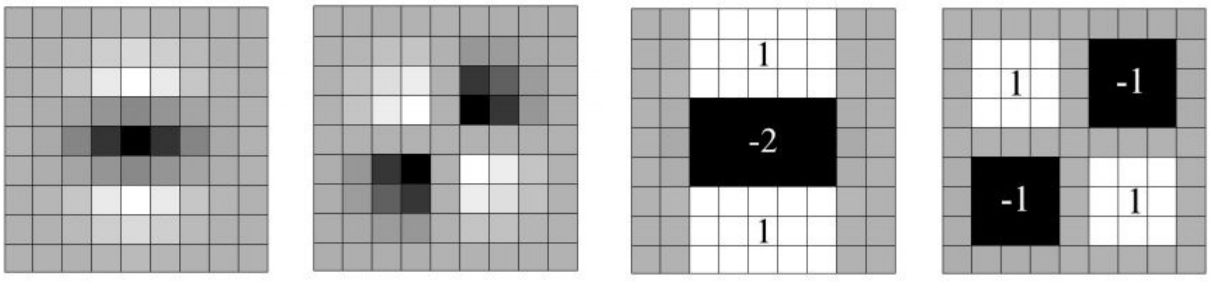


Figure 6 – Using integral images, it takes only three operations to calculate the area of any rectangular region inside the image (Reproduced from Bay et al. [2008]).

- Hessian matrix-based interest points

  The Fast-Hessian detector is a blob detector based on the determinant of the Hessian and uses box filters to approximate the Hessian matrix. These boxes approximate the second order Gaussian derivatives and can be easily evaluated using integral images (three operations). Then, the computation time becomes independent of the filter size. The SURF's box-filters approximation are shown in Figure 7.



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

Figure 7 – The Gaussian second-order partial derivative in y-direction (a) and xy-direction (b). SURF's box-filters approximation for the second-order Gaussian partial derivative in y-direction (c) and xy-direction (d). The gray regions are equal to zero (Reproduced from Bay et al. [2008]).

2.1.3.3 Center Surround Extremas (CenSurE)

The scale-invariant center-surround detector was proposed by Agrawal et al. [2008]. While SIFT [Lowe, 2004] uses DoG to approximate the Laplacian, CenSurE proposes a simpler approximation using center-surround filters that are bi-level (inner box and outer box). These boxes are used to multiply the image value by either 1 or -1. Figure 8 shows a progression of bi-level filters.

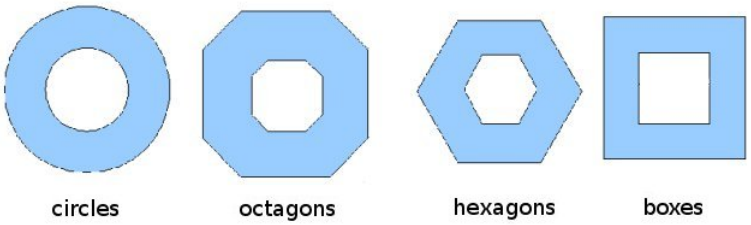


Figure 8 – Progression of Center-Surround bi-level filters. Left to right: circular symmetric BLoG (Bi-level LoG) filter. Successive filters (octagon, hexagon, box) have less symmetry. (Modified from Agrawal et al. [2008]).

The circular filter is the most faithful bi-level approximation for the Laplacian, but hardest to compute. The other filters can be computed rapidly using a slightly modification of integral images presented in Fast-Hessian detector. This modified version of integral images can be exploited to compute polygonal filters, where the degree of slant is controlled by a parameter α.

$$\sum_{\alpha}(x,y) = \sum_{j=0}^{y} \sum_{i=0}^{x+\alpha(y-j)} I(i,j).$$

2.1.3.4 Features from Accelerated Segment Test (FAST)

The FAST detector was proposed by Rosten and Drummond [2006] and takes further the idea of comparing pixels used in SUSAN detector [Smith and Brady, 1997] and MIC detector [Trajković and Hedley, 1998]. FAST compares pixels values on a discretized circle of 16 pixels around the central pixel under test p (corner candidate) as illustrated in Figure 9 . The candidate p is considered as corner if n contiguous pixels are darker than $(I_p - t)$ or brighter than $(I_p + t)$. Where I_p is the intensity value of the candidate pixel p and t is a threshold parameter.

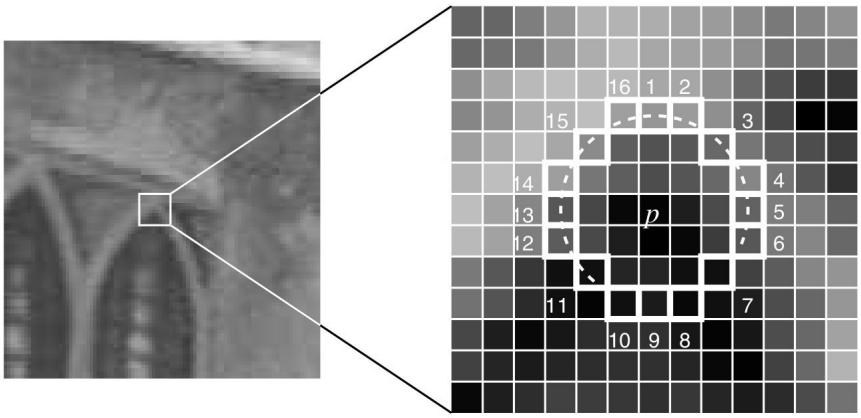


Figure 9 – Segment test corner detection. The highlighted squares (right side) are considered as the circular region around the pixel under test (Reproduced from Rosten and Drummond [2006]).

To accelerate the process, a high-speed test was explored, using $n = 12$. This test consists in making only four comparisons, between the pixel under test and pixels 1, 5, 9, and 13 in the circle, if three of these pixels are darker than $(I_p - t)$ or brighter than $(I_p + t)$ the candidate is retained, otherwise the candidate is rejected. Then, the full segment test is performed only on the remaining candidates. Nevertheless, the high-speed test does not reject as many candidates for n < 12 and also the choice of pixels is not optimal because it assumes an order of pixels and the distribution of corner appearances. In order to overcome those difficulties the authors use a machine learning approach based on decision tree classifiers and then this decision tree is used for fast detection in other images. In practice the decision tree detects multiple interest points in adjacent locations, which is undesirable. To avoid adjacent corners, a score function must be computed for each corner and remove corners which have an adjacent corner with higher score, this procedure is called non-maximal suppression and is applied since the segment test does not compute a corner response function.

FAST detector became widely used, particularly in systems where low cost and fast computation are crucial constraints, mainly due to its computational efficiency. On the other hand, FAST detector has several weaknesses when used for image retrieval, mainly because the detector itself does not produce multi-scale features, does not compute keypoint orientation and also does not produce a response allowing to select the best keypoints. To overcome these limitations, variations and improvements of FAST were proposed recently, and we will describe briefly the most important ones.

The authors themselves, propose an enhanced repeatability version of FAST called FAST–ER [Rosten et al., 2010]. This FAST derivation, as well as FAST, uses the ID3 algorithm to build a decision tree, but using a 3 pixels discretized circle instead of the 1 pixel discretized circle used in FAST (Bresenham's circle-Figure 9). Both, FAST and FAST–ER, perform the learning step over corner candidates detected from a training set of images using the segment test criteria. Since those candidates do not represent all possible corner configurations, the decision tree is optimized for the specific set of training images or environment, and by consequence the decision tree leads to false positive and false negative responses of the corner detector.

The Adaptive and Generic Accelerated Segment Test — AGAST [Mair et al., 2010] makes the decision trees more generic and provides high performance for an arbitrary environment, this is achieved by combining two binary trees which are optimized, one for homogeneous and one for structured regions. AGAST also performs non-maximum suppression as FAST. An additional scheme providing invariance to scale can be implemented over AGAST with a scale–space pyramid, as described in BRISK [Leutenegger et al., 2011] (further discussed in Section 2.2.2).

Another important FAST derivation was introduced as Oriented FAST — oFAST in ORB [Rublee et al., 2011] (further discussed in Section 2.2.2), this detector is essentially a multi-scale FAST with orientation. Features at multiple scales are obtained using a scale–space pyramid and the corner orientation correspond to the orientation of the vector from the patch's center to the intensity centroid or gravity center of the patch.

Efficient implementations are crucial in the context of mobile visual applications, where the response time has to be as fast as possible to provide a good a user experience, without draining the limited computational and power resources. All of those efficient implementations described before will be considered in the experimental procedure to determine the most suitable ones for mobile applications. Finally, we summarize the feature detectors presented in this section into the Table 1. This overview highlights the type of the feature detected and its major achievement (invariance to certain transformations).

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Reference
Harris	✓			✓			Harris and Stephens [1988]
Hessian		✓		✓			
SUSAN	✓			✓			Smith and Brady [1997]
MIC	✓			✓			Trajković and Hedley [1998]
Harris–Laplace	✓			✓	✓		Mikolajczyk and Schmid [2004]
Hessian-Laplace		✓		✓	✓		Mikolajczyk and Schmid [2004]
Harris–Affine	✓			✓	✓	✓	Mikolajczyk and Schmid [2004]
Hessian–Affine		✓		✓	✓	✓	Mikolajczyk and Schmid [2004]
Salient Regions			✓	✓	✓	✓	Kadir and Brady [2001]
MSER			✓	✓	✓	✓	Matas et al. [2004]
Difference-of-Gaussians (DoG)		✓		✓	✓		Lowe [2004]
Fast-Hessian		✓		✓	✓		Bay et al. [2008]
CenSurE		✓		✓	✓		Agrawal et al. [2008]
FAST	✓						Rosten and Drummond [2006]
AGAST	✓						Mair et al. [2010]
oFAST	✓			✓			Rublee et al. [2011]
Multi-scale AGAST	✓				✓		Leutenegger et al. [2011]

Table 1 – Overview of the feature detectors presented in this chapter. FAST and AGAST are not invariant to changes in scale and rotation by themselves, but they can obtain robustness against changes in scale by using a scale-space pyramid (e.g. Multi-scale AGAST) and against rotations by estimating the orientation of the point of interest (e.g. oFAST).

2.2 Feature Descriptors

A *feature descriptor* translates each local image information into a *feature vector*. Local information can be sampled using local features detectors (around regions of interest or patches), but, sometimes can be densely sampled on a regular grid over the image. Sampling sparsely around repeatable features is more commonly employed in target recognition tasks (targetting, recognizing or tracking an specific object, scene or image), while dense sampling is more common for classification tasks (recognizing broad classes, like "cats" or "people").

Usually, description is performed on *normalized* patches, i.e., regions already scaled and rotated according to information obtained by the feature detectors. Therefore, some of the invariances provided by local description are responsibility of the detector (e.g., translation, scale and rotation), while others are responsibility of the descriptor (e.g., illumination changes). Feature description and feature detection have a close relationship, since they are the main steps in feature extraction. Sometimes, a single technique (e.g. SIFT) is associated both to detection and description. Then, to avoid misunderstandings, keep in mind that detection returns a sequence of interest points, oriented circles, oriented ellipses, or region coordinates; while the descriptor processes that information to obtain feature vectors.

In general, description is more expensive to compute than detection (e.g. SIFT, SURF, BRISK [Leutenegger et al., 2011, Section 4.3]). Therefore, efficient implementations are even more crucial than in detection stage. Although there are several methods to measure the performance of a descriptor, all of them are based on establishing a degree of discriminative power and robustness to minor variations. Over the years, many feature descriptors were proposed and is out of the scope of this dissertation to perform a detailed study of each one. In this section we will explore the most important feature descriptors grouped according to two major approaches: those descriptors based on histograms of gradients and those descriptors based on binary intensity tests. For each family of descriptors, we will study their evolution over the years, pursuing high performance and computational efficiency.

2.2.1 Descriptors Based on Histograms of Gradients

Many of the most important descriptors in the literature are variations upon histograms of gradients (HoG). These descriptors, in general, share a common pipeline of processes: first, a smoothing filter is applied to the patch in order to avoid abrupt changes and to give less emphasis to gradients that are far from the center of the patch. Then, the gradients are computed using derivative masks. Next, the feature vector is obtained by concatenating the accumulative result of the *gradient histogram binning* over the *spatial binning*. These two terms are extensively used in [Winder et al., 2009; Chandrasekhar et al., 2012].

Gradient histogram binning: A histogram itself implies to "bin" the range of values (quantization), i.e., slicing the entire range of values into intervals. For gradients, the range of values to be divided correspond to all possible angles of the gradient (0°–360°). Then, every gradient (d_x, d_y) is assigned to one bin (hard assignment) or more than one bin (soft assignment).

Spatial binning: In this case the binning is done over the patch. Then, we found sampled regions or subregions distributed in two main configurations, those with a square grid or SIFT-like distribution and those with log-polar configurations or DAISY-like as presented in Figure 10. In both cases, the sampled regions may or may not allow overlapping, called as soft-assignment and hard-assignment, respectively.

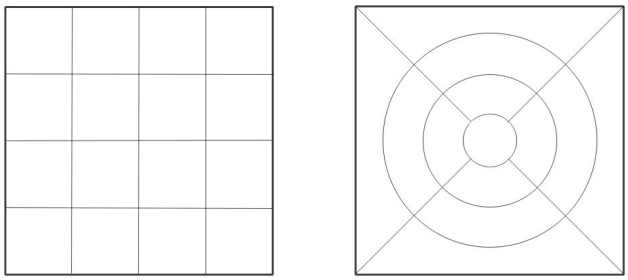


Figure 10 – Spatial binning configuration examples in a square patch. A square grid 4×4 defines 16 square spatial bins (left). A log-polar configuration with 4 angular bins and 2 radial bins defines 9 spatial bins (right).

Gradient histogram binning and spatial binning are performed in the space of derivates (d_x, d_y) and both have a set of parameters to determine the number of bins or granularity and their distribution. As we will see, these parameters can be obtained empirically or by using learning processes.

2.2.1.1 Scale-Invariant Feature Transform (SIFT)

Proposed by D. Lowe Lowe [2004], this is undoubtly the most well-known and influential local descriptor in the literature. SIFT is both a feature detector (based upon Differences-of-Gaussians, or DoG, which approximate Laplacians-of-Gaussians), and a feature descriptor. As a descriptor, it computes the gradient magnitude and orientation at each point in the patch. Gradient magnitudes are weighted with a Gaussian centered at the interest point to give less emphasis to those in the border, and gradient orientations are rotated relative to the keypoint orientation in order to achieve rotational invariance. In the common SIFT configuration, those gradients are accumulated into 8 orientation bins over 4×4 spatial cells.

Then, each one of these 4×4 spatial bin has a 'star' of 8 orientation bin D_{SIFT}.

$$D_{SIFT}(i) = \sum_{d_x, d_y \in \Omega_i} \sqrt{d_x^2 + d_y^2}.$$

Where: $\Omega_i = \left\{(d_x, d_y) \mid \frac{\pi(i-1)}{4} \leq tan^{-1}\frac{d_y}{d_x} < \frac{\pi i}{4}\right\}$ and $i = \{1, \cdots, 8\}$.

The resulting descriptor is $4 \times 4 \times 8 = 128$-dimensional descriptor. This basic SIFT scheme of description inspired several new feature descriptors with various refinements, some of them are described in the next pages.

2.2.1.2 Gradient Location and Orientation Histogram (GLOH)

Proposed by K. Mikolajczyk and C. Schmid Mikolajczyk and Schmid [2005]. This descriptor quantizes the gradient into 16 orientation bins and uses a log-polar configuration as spatial bin. The optimized configuration presented by the authors correspond to 17 spatial bins (8 angular bins, 2 radial bins and a central bin). Furthermore, they use PCA as a dimensionality reduction technique to obtain a 128-dimensional vector descriptor.

2.2.1.3 Histograms of Oriented Gradients (HOG)

Introduced by N. Dalal and B. Triggs Dalal and Triggs [2005] in a context of object detection. HOG works in a portion of image or window setted a priori (64×128 pixels in a detection window for original article). A description of HOG with the best configuration attained by the authors is as follows: gradients are computed using the simplest scheme ($1 - D$ mask $[-1, 0, 1]$ without previous smoothing) and then use 9 orientation bins equally spaced over $0° - 180°$ ("unsigned" gradient). They use two spatial configurations: square spatial cells, similar to SIFT, to obtain the R-HOG descriptor and circular blocks partitioned into cells in log-polar configuration, similar to GLOH, to obtain the C-HOG descriptor. Then, each block (group of cells) is normalized separately and the final descriptor will be the concatenation of the vector of all components of the normalized cell responses from all of the blocks in the detection window. The optimal configuration values are:
R-HOG: use cells of 8×8 pixels and blocks of 2×2 cells.
C-HOG: number of angular bins = 4, number of radial bins = 2, radius of the central bin = 4 pixels. Then, this configuration has 9 spatial bins.
The final feature for R-HOG is a 3780-dimensional vector: $3780 = 7 \times 15 \times 2 \times 2 \times 9$. Since the window of 64×128 pixels has 7×15 blocks and every block has 2×2 cells. Finally, there are 9 orientation bins.

2.2.1.4 Speeded Up Robust Features (SURF)

Arguably the second most important local descriptor in the literature, it was proposed by Bay et al. [2008] as an accelerated version of SIFT. SURF is also both a detector and descriptor. As a descriptor, it splits the patches into 4×4 square cells. For each cell, the Haar wavelet responses are computed. The horizontal and vertical Haar wavelet responses can be interpreted as d_x and d_y respectively. To bring information about polarity of intensity changes, SURF employs the sum of absolute values of the horizontal and vertical responses $|d_x|$ and $|d_y|$. Then, each spatial bin has a 4-dimensional vector v that describes the intensity structure.

$$D_{SURF} = v = (\sum_{d_x}\sum_{d_y} d_x, \sum_{d_x}\sum_{d_y} d_y, \sum_{d_x}\sum_{d_y} |d_x|, \sum_{d_x}\sum_{d_y} |d_y|).$$

Finally, these v vectors are concatenated for all spatial bins, resulting in a $4 \times 4 \times 4 = 64$-dimensional feature vector. SURF descriptor is very similar to SIFT descriptor because they both use a square grid 4×4 spatial bin in their configuration and are focused on the spatial distribution of gradients.

2.2.1.5 DAISY

Introduced by E Tola, V Lepetit and P. Fua Tola et al. [2008]. DAISY is inspired by earlier descriptors SIFT and GLOH. First, the gradients are computed and quantized into $N = 8$ directional bins. From interest point $P(x, y)$ use these directions and choose N points at distances R_1, R_2, R_3 generating a sort of log-polar configuration and use these points $P_{1\leq i\leq N}(x, y, R_{1\leq j\leq 3})$ as centers for spatial sub-regions. Then, a convolution of directional bins with Gaussian kernels are performed and the interest values of this process are those related to the points $P(x, y), P_1(x, y, R_1), \cdots, P_8(x, y, R_1)$ convolved with Gaussian of σ_1, $P_1(x, y, R_2), \cdots, P_8(x, y, R_2)$ convolved with Gaussian of σ_2, $P_1(x, y, R_3), \cdots, P_8(x, y, R_3)$ convolved with Gaussian of σ_3 and finally these values are normalized. These convolutions are an efficient way to compute the weighted sums in the accumulative process of orientation bins over spatial sub-regions. For optimal values of σ_1, σ_2 and σ_3 this spatial sub-regions are overlapping achieving smooth transitions between the regions. The optimized configuration has: $8 + 8 \times 3 \times 8 = 200$-dimensional feature vector.

Winder and Brown [2007] shows that DAISY configurations are better than square grid configurations in the designing of a discriminative local image descriptor. This is based mainly on his polar Gaussian accumulative process (polar Gaussian pooling) and the fact that this kind of configuration can be very efficiently computed. As a consequence of these results, S. Winder performs a learning study of the optimal descriptors with DAISY configurations Winder et al. [2009]. In this study, the pipeline used attempt to optimize the

parameters of DAISY in order to obtain the best matching performance using a training set of matching and non-matching image patches.

The feature descriptors described before were inspired by SIFT and consequently they also share some of its properties. SIFT descriptor is highly discriminant and has become a standard, but its feature vectors have 128 components (high dimensionality) and also is slow to compute for real-time tasks. Conventionally, each one of the 128-dimensional SIFT feature vector is stored as 1024 bits and store millions of descriptors becomes impractical. SURF and DAISY address the issue of speed, but their feature vectors still have high dimensionality, 64-D for SURF and 200-D for DAISY, this high dimensionality is an issue not only for storage but also for further matching process. To reduce the amount of data to be stored we can identify two major approaches: *dimensionality reduction* and *binarization*. For more information about compression schemes, the reader is referred to the comprehensive survey of Scale Invariant Feature Transform (SIFT) compression schemes performed by V. Chandrasekhar Chandrasekhar et al. [2010a]. In general, compression techniques are applied to feature vectors previously obtained *e. g.* using Principal Component Analisis: PCA-SIFT/PCA-DAISY to reduce the dimensionality or using hashing techniques like Locality Sensitive Hashing (LSH) to build a binary string for high dimensional descriptors. These compression schemes provide satisfactory results, this means, they reduce the amount of data to be stored with only a small loss in performance. Nevertheless, they are even more expensive to compute than SIFT or DAISY, since they also perform a compression, which involves an additional computation time.

2.2.1.6 Compressed Histogram of Gradients (CHoG)

The Compressed Histogram of Gradient was introduced in Chandrasekhar et al. [2012] as an alternative to compute a low bit-rate descriptor. This descriptor achieves similar performance to 1024-bits SIFT at aproximately 60 bits/descriptor and also performs better than schemes that simply take SIFT-like descriptors in tandem with a compression algorithm. One of the main vantages of CHoG descriptor is the capability to match and compare the compressed feature vectors (without decompressing first), thus saving time and computational power Girod et al. [2011].

CHoG descriptor, as well as all the descriptors based on histograms of gradients described so far, performs a spatial binning and gradient histogram binning. Since DAISY configurations perform better than square-grid configurations, the patch is divided into subregions using a log-polar configuration as proposed by Mikolajzyc and Schmid Mikolajczyk and Schmid [2005]. The spatial bin in CHoG descriptor uses overlapping regions (soft-binning) instead of disjointed cells (hard-binning) as used in "Picking the best DAISY" Winder et al.

[2009]. The histogram bin[1] configuration of CHoG, aims at describing the distribution of 2D-gradients. But, unlike other descriptors described before, CHoG's histogram bin is obtained performing a statistical study over 10000 cells from a training data set. The gradients of these cells are computed and plotted. As a result, is observed that the distribution of gradients is strongly peaked around $(0, 0)$ as shown in Figure 11a, and slightly more oriented over the Y axis (Figure 11b). Finally, the histogram bin configuration is chosen considering this statistical behaviour. Some examples of configurations and their associated quantized cells are shown in Figure 11c.

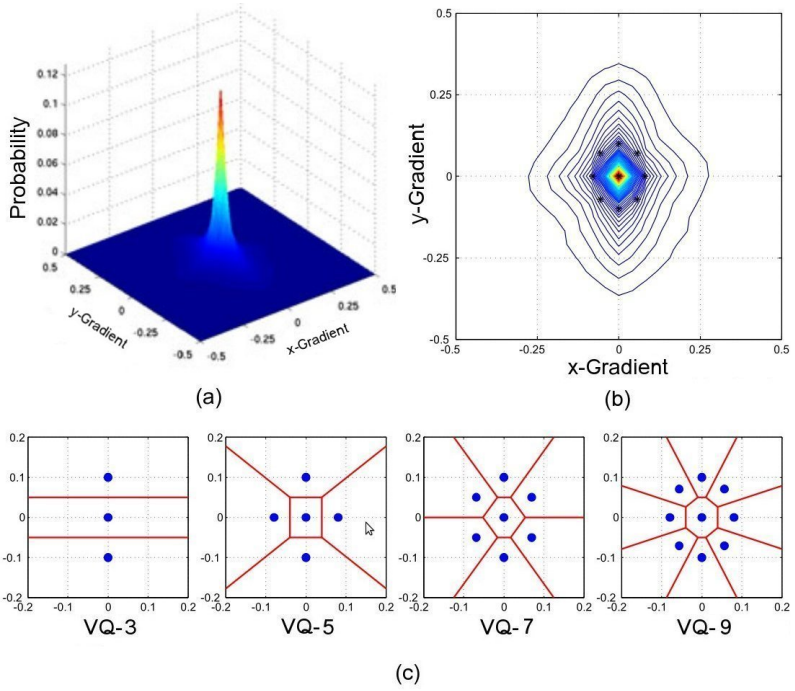


Figure 11 – Histrogram bin configuration. (a) distribution of gradients. (b) Contour curves of distribution. (c) examples of histogram bin configurations referred as Vector Quantization (VQ). Although figure (c) shows well-delimited regions, it is important to remember that soft binning is employed. The red lines mark the regions where a particular bin has dominant, but not necessarily exclusive, influence. (Reproduced from V. Chandrasekhar *et al.* Chandrasekhar et al. [2012]).

The result of concatenating the normalized accumulated histograms of gradients over each spatial bin is called as UHoG (Uncompressed Histogram of Gradients) and is a high dimensional vector. Probably, the most innovative process of CHoG descriptor lies in its compression approach, and as we explain before, the compression is typically applied on the final feature vector. But, since UHoG has a fixed-length normalized histogram binning for each

[1] Although the term histogram binning is somewhat ambiguous it is extensively used in the Chandrasekhar et al. [2012]; Winder and Brown [2007] papers, meaning the **gradient** histogram binning.

spatial bin, it is feasible to apply the compression directly on these normalized histograms. Three compression schemes were tested: Huffman Tree Coding, Entropy Constrained Vector Quantization (ECVQ) and Type Quantization Chandrasekhar et al. [2010b] Reznik et al. [2010]. Among them, ECVQ is the more expensive to compute (its cost being comparable to a k-means algorithm) and the lattice-based Type Quantization presents the better results.

The CHoG configuration that has a slightly better performance than SIFT corresponds to a 9-dimensional spatial binning and 7-dimensional histogram binning. Then, the UHoG is a $9 \times 7 = 63$-dimensional descriptor. Then, each one of the normalized accumulated 7-bin histogram distribution is compressed independently and represented with few bits as possible. In Huffman Tree Coding each normalized accumulated histogram bin is encoded into a binary tree and as the number of all possible binary trees is limited, we can enumerate them and store only the index of the tree that represent the normalized histogram distribution. In Type Quantization the basic idea is to quantize the normalized histogram distributions using a set of reconstruction points or codebook. This codebook, is obtained by constructing a Lattice of distributions or Types. Then, each normalized histogram distribution is quantized into the closest Type. The number of types is limited and can be enumerated, again only the index will be stored. Finally, the descriptor is obtained concatenating these indexes.

CHoG is the most sophisticated descriptor based on the analysis of the histogram of gradients and is considered as a pioneer in low-bitrate representation for local descriptors. Therefore, due to its relevance, we have studied CHoG in detail and reimplemented it from scratch. This effort was very rewarding in terms of the high amount of knowledge gained about the deepest details in the processes and implementation of a descriptor based on histogram of gradients, but no so much in terms of replicability of its results. The Content about our implementation can be found in Appendix B. A more detailed numerical study of the bit-rate of the descriptor and additional information about the compression algorithms can be founded in Appendix A.

2.2.1.7 Boosting Binary Keypoint Descriptor (BinBoost)

The Boosting Binary Keypoint Descriptors was proposed by Trzcinski et al. [2013] as a fast and compact descriptor.

Considering: $C(x) = [C_1(x), \cdots, C_D(x)]$ the binary BinBoost descriptor that maps a patch x into a D-dimensional binary string (D = 64 bits, in the optimal case). Each bit is represented as $C_d(x)$, where: $d \in \{1, \cdots D\}$ and is computed as:

$$C_d(x) = sgn(b_d^T h_d(x)).$$

Where: $h_d(x) = [h_{d,1}(x), \cdots, h_{d,K}(x)]^T$ are K weak learners ($K = 128$, in the optimal case) weighted by a vector $b_d = [b_{d,1}, \cdots, b_{d,K}]$. These weak learners are gradient-based image features obtained by function that considers the orientation of the gradients over the patch.

The problem formulation (optimization problem) for a labeled training set is closely related to the AdaBoost standard, i.e. the problem is based on weak learners and the formulation will produce a strong learner or classifier. After solving the optimization problem, the values of the weights b_d are known. Then, since h_d is obtained for each patch and now we know b_d, we can compute the BinBoost descriptor of new patches.

Note: We choose to not go deep into this formulation because at this point it is not relevant for us to know the solution in detail. Also, we skip the definition of weaker learners since it is not crucial to know the formalism to obtain them.

2.2.1.8 Binary Reslzable Gradient HisTogram (BRIGHT)

Proposed by K. Iwamoto Iwamoto et al. [2013]. BRIGHT is a binary descriptor based on hierarchical histogram of oriented gradients and a progressive bit selection. The hierarchical HOG provides an extra robustness to scale changes and consists in three layers as showed in Figure 12.

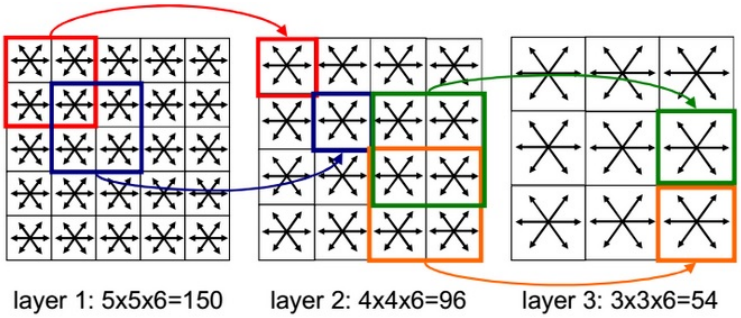


Figure 12 – Hierarchical HOG layers. A layer is constructed agregating the 2×2 blocks of the previous layer (Reproduced from K. Iwamoto Iwamoto et al. [2013]).

Then, the resultant $150 + 96 + 54 = 300$-dimensional feature vector is binarized by comparing to a threshold 't'. The HOG has 6 orientation bins $\{h_0, h_1, \cdots, h_5\}$ and their binarization is computed as follows:

$$b_i = \begin{cases} 1 & if : h_i > t \\ 0 & if : h_i \leq t \end{cases}$$

Where t is: $t_{layer1} = 0.063 \sum_{i=0}^{5} h_i,\ t_{layer2} = 0.1 \sum_{i=0}^{5} h_i,\ t_{layer3} = 0.12 \sum_{i=0}^{5} h_i$

Since the 300-dimensional vector is binarized, a progressive bit selection is performed following a pre-established pattern that selects a half of the bits of each block as maximum. Then, the descriptor is a string bits: $300/2 = 150$ bits in full size and 32 bits as minimum (progressive scalable from 32 bits to 150 bits).

2.2.2 Descriptors Based on Binary Intensity Test

In 2010 M. Calonder Calonder et al. [2010] introduces his BRIEF descriptor with a novel approach to compute a highly efficient and discriminative descriptor. The algorithm proposed computes a set of intensity value comparisons and it avoids any computation of gradients, quantization schemes, compression processes or even an accumulative histogram over spatial configurations. This descriptor represents the comparison responses into binary values $(0, 1)$, and it generates a bit string as output. Feature descriptors based in this binary intensity test are called as *binary descriptors* and due to their nature, these descriptors allow to compute a highly efficient distance between two feature vectors (Hamming distance). After BRIEF was released, several extensions come to light, improving mainly the spatial arrangement of pairs to be tested. In this section we will present a review of these recent binary descriptors.

2.2.2.1 Binary Robust Independent Elementary Features (BRIEF)

BRIEF is the first descriptor based on binary intesity test and was proposed by M. Calonder Calonder et al. [2010]. This descriptor shows to be a highly discriminant descriptor and more efficient than descriptors previously proposed. The BRIEF efficiency is achieved thanks to the novel approach applied, which consists mainly in generating binary strings from simple intensity value comparisons over a smoothed image patch. The intensity value test (τ) represents the comparison responses whether by "0" or "1".

$$\tau(P; x, y) = \begin{cases} 1 & if : I(x) < I(y) \\ 0 & otherwise \end{cases}$$

Where, P is the patch smoothed using a Gaussian kernel and $I(x)$ represents the intensity value at pixel x. The patches are obtained with the Fast-Hessian detector, but it is not limited only to the use of this feature detector. In fact, we can use any feature detector and describe using the intensity value test.

Then, each test (τ) generates a binary value. These binary values are concatenated into a binary string that will be the final feature vector. Commonly, the bit-length of BRIEF descriptor are 128, 256 or 512 bits and due to their correspondence in bytes they can also be referred as BRIEF-16, BRIEF-32 and BRIEF-64, respectively. An empirical comparison

between several spatial arrangements of binary tests (Figure 13) shows the best results on tests sampled from an isotropic Gaussian distribution (Figure 13b).

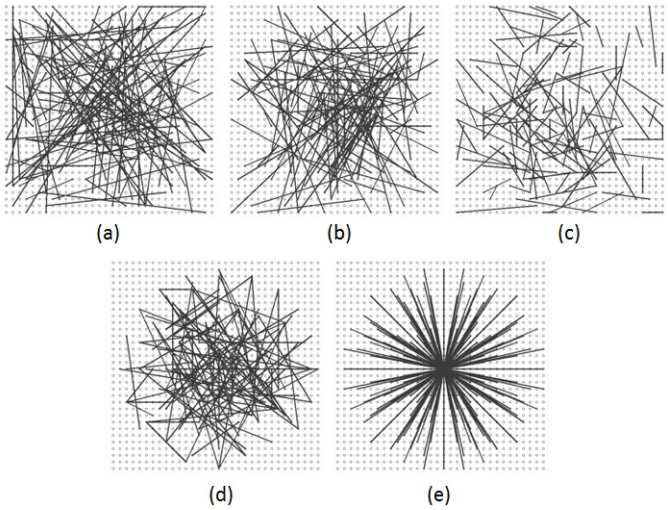


Figure 13 – Different approaches to choosing the 128 test locations over a patch. (a) Uniform distribution, (b) isotropic Gaussian distribution, (c) Gaussian distribution oriented in the horizontal axis "X" (d) random sample from polar grid.(e) random sample from polar grid and $(0, 0)$. (Reproduced from M. Calonder Calonder et al. [2010]).

2.2.2.2 Oriented FAST and Rotated BRIEF (ORB)

Proposed by E. Rublee Rublee et al. [2011]. ORB descriptor is, in different aspects, an improved BRIEF.

First, as the name itself allude, ORB uses the oriented FAST detector (oFAST) described in 2.1.3, which is more efficient to compute than the Fast-Hessian detector used in BRIEF. Second, ORB descriptor uses a learning process to determine the spatial arrangement of binary tests. First, the training set of patches (31×31) are smoothed using an integral image, where each test point is a 5×5 sub-window. Then, the number of possible tests result in $\binom{26}{2} = 205590$. The learning algorithm search for a set of 256 uncorrelated tests, which will produce a 256 bit string. Finally, once the spatial arrangement is known, the BRIEF binary string is computed. For cases with in-plane rotations, the spatial arrangement of tests is rotated (steered BRIEF) instead of rotate the patch itself (more efficient).

Since ORB include a training process, the spatial arrangement obtained will be better for an specific dataset (from which the training set was selected) than the spatial arrangement of test for the BRIEF descriptor.

2.2.2.3 Binary Robust Invariant Scalable Keypoints (BRISK)

Proposed by S. Leutenegger Leutenegger et al. [2011]. BRISK is a super-fast descriptor which uses a DAISY inspired deterministic sampling pattern for intensity tests. The BRISK efficiency is obtained by using the accelerated extension of FAST, the AGAST detector Mair et al. [2010].

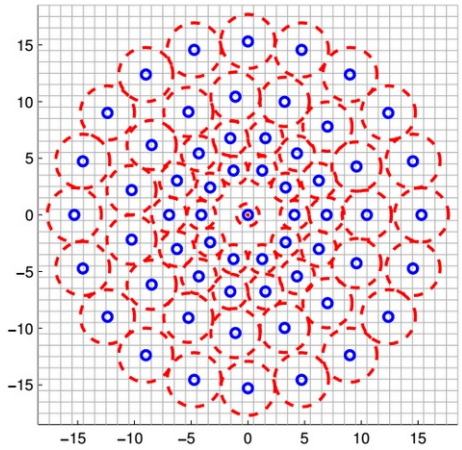


Figure 14 – The BRISK sampling pattern with 60 sampling points: blue circles denote the sampling locations and red dashed circles are drawn at a radius corresponding to the standard deviation σ of the Gaussian kernel used to smooth the intensity values. (Reproduced from S. Leutenegger Leutenegger et al. [2011].

BRISK descriptor uses sampling locations in a DAISY pattern (Figure 14). The intensity value at each sampling location P_i is obtained by using a Gaussian smoothing with a standard deviation (σ) proportional to the distance between sampling locations on the same concentric circle. Then, the smoothed intensity values at sampling point pair (P_i, P_j) are $I(P_i, \sigma_i)$ and $I(P_j, \sigma_j)$. Those pairs where the distance $\|P_j - P_i\|$ is bigger than $\delta_{min} = 13.67 \times (keypoint\_scale)$ are grouped into the long-distance pairs L and those pairs where the distance is less than $\delta_{max} = 9.75 \times (keypoint\_scale)$ are grouped into the short-distance pairs S. The L subset of sampling pairs is used to estimate the orientation of the keypoint. The local gradient $g(P_i, P_j)$ can be estimated as:

$$g(P_i, P_j) = (P_j - P_i).\frac{I(P_j,\sigma_j) - I(P_i,\sigma_i)}{\|P_j - P_i\|^2}.$$

Then, the direction of the keypoint (orientation) can be estimated by the aggregation of local gradients of the L pairs.

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \sum_{(P_i,P_j)\in L} g(P_i, P_j).$$

To obtain invariance to in-plane rotations the sampling pattern is rotated by $\alpha = arctan2(g_y, g_x)$ around the keypoint. Finally, the intensity value tests are performed over the S subset of sampling pairs to obtain the 512 bits BRISK descriptor.

2.2.2.4 Fast Retina Keypoint (FREAK)

Proposed by A. Alahi Alahi et al. [2012]. The two major contributions of FREAK descriptor are inspired by the human visual system, its retinal sampled pattern for intensity tests and its capability to perform a saccadic search. FREAK descriptor uses the same multi-scale AGAST feature detector used in BRISK and also uses a similar approach to estimate the keypoint orientation.

The FREAK sampling pattern is an analogy to the topology of the retina (Figure 15). The sampling points are distributed along concentric circles but unlike BRISK, the density of points decrease exponentially with the distance to the keypoint (center of the patch). Finally, a learning algorithm similar to ORB is applied to find the 512 most relevant pairs and build the FREAK bit string.

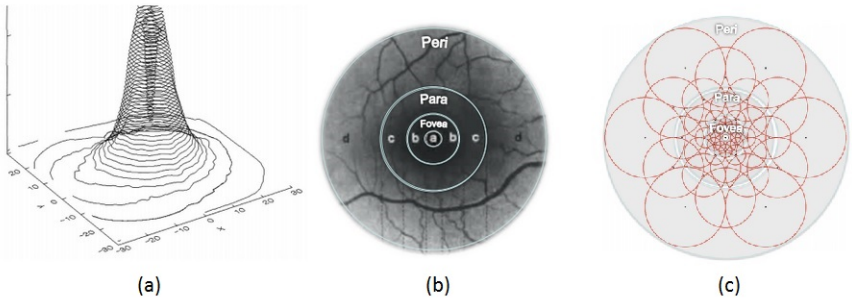


Figure 15 – From retinal photoreceptors to pixels. (a) Density of ganglion cells over the retina, (b) Retina areas, (c) FREAK sampling pattern, each circle represents the region of influence to be considered to compute the intensity value. Each one of this regions will consider a Gaussian kernel to smooth this region before to compute the intensity value (Reproduced from A. Alahi Alahi et al. [2012]).

The 512 FREAK bit string is sorted by its relevance and this allows to perform a mimic of the human phenomena of saccadic search by performing a cascade of comparisons. Then, the first cascade corresponds to the first 128 bits (16 bytes) of FREAK descriptor, the second cascade correspond to the next 128 bits, and so on, until the fourth cascade (last 128 bits). Then, each cascade has 16 bytes and the cost to compare 1 byte or 16 bytes is almost the same with the use of Single Instruction and Multiple Data (SMID) instructions. Finally, the first cascade of comparisons discards more than 90% of the candidates and can be executed with the computational cost of comparing two bytes.

2.2.2.5 Fast Robust Invariant Feature (FRIF)

Proposed by Z. Wang Wang et al. [2013]. FRIF descriptor is a binary descriptor which adds local pattern information to the bit string obtained from intensity value tests. The FRIF sampling pattern is a slightly modified version of the BRISK spatial arrangement (more overlapping areas) and the keypoint orientation is obtained using the same procedure described in BRISK.

FRIF is called as mixed descriptor because its bit string is a concatenation of local pattern information and responses to intensity value tests. For each one of the N pattern locations (pattern location is a point in the spatial arrangement of binary tests), the local pattern information is encoded by computing a pair-wise comparison between four points $S(P_i) = \{s_{i,k}, k = 1, 2, 3, 4\}$ sampled on a circle centered at P_i ($i = \{1, 2, 3, \cdots, N\}$). Then, the local information is a $\binom{4}{2} \times N = 6 \times N$ bit string. On the other hand, the intensity value test is performed over $512 - 6 \times N$ short-distance pairs using the same algorithm as BRISK descriptor. Finally, the local pattern information and the intensity value tests are concatenated into a $6 \times N + 512 - 6\times = 512$ bit string.

2.2.3 Summary

In this chapter we have studied the state of the art of feature descriptors and their evolution over the years. We have established a major distinction between them based on their approaches: whether the method is focused in the distribution of the gradients (histogram of gradients) or focused in the distribution of the intensity (binary test). At the first glance, these two families of descriptors have a big difference in terms of their computational cost. Binary descriptors only compute a intensity test of pixels over the smoothed patch while gradient-based descriptors have to compute gradients, histograms and accumulative processes over each spatial bin. We can summarize all methods just by highlighting two feature descriptors: SIFT as the most successful and representative feature descriptor based on histogram of gradients and BRIEF as a pioneer binary descriptor.

Overall, descriptors based on histogram of gradients have experienced a notorious evolution in terms of bit-rate (compactness), configurations and efficiency. SIFT descriptor is one of the most cited, used and successful algorithm in multimedia retrieval and computer vision. We also, have to emphasize the importance of some other feature descriptors studied along this chapter. SURF descriptor presents an accelerated algorithm maintaining good performance and is probably the most successful algorithm after SIFT. DAISY descriptor has explored intensively the spatial configurations (spatial binning), even using learning methods. CHoG descriptor goes further in the effort to accomplish compactness as a gradient based

descriptor, this descriptor explores intensively compressed algorithms. On the other hand, binary descriptors are drastically more compact and more efficient to compute, but in general they experience losses in terms of quality and invariance against transformations.

Feature vectors represent an image and can be used to establish whether two images are similar or not. To compare feature vectors we have to look into the characteristics of such feature vectors. Then, each one of them have an specific measure of distance that fits better. Descriptors like SIFT, SURF and DAISY generate a feature vector with high dimensionality and generally use the euclidean norm (L2) to measure the distance between two feature vectors. CHoG descriptor pursues compression and compactness. Then, their vectors are better represented as a sequence of normalized vectors and use the kullback Leibler divergence (KL) as a distance. On the other hand, those descriptors that produce a binary bit string are drastically more efficient to compare because this comparison can be performed using the Hamming distance, which is a simple XOR operation and a count of "1"s or count of population (popcount).

Finally, we summarize the feature descriptors studied in Table 2 and highlight their approach and characteristics of their feature vectors.

Feature Descriptor	Based on Histograms of Gradients	Based on Binary Intensity Test	dimension	distance	Reference
SIFT	✓		128	L2	Lowe [2004]
GLOH	✓		128	L2	Mikolajczyk and Schmid [2005]
HOG	✓		3780	L2	Dalal and Triggs [2005]
SURF	✓		64	L2	Bay et al. [2008]
DAISY	✓		200	L2	Tola et al. [2008]
CHoG	✓		45	KL	Chandrasekhar et al. [2012]
BinBoost	✓		64 b	Hamming	Trzcinski et al. [2013]
BRIGHT	✓		32 to 150 b	Hamming	Iwamoto et al. [2013]
BRIEF		✓	128, 256, 512 b	Hamming	Calonder et al. [2010]
ORB		✓	256 b	Hamming	Rublee et al. [2011]
BRISK		✓	512 b	Hamming	Leutenegger et al. [2011]
FREAK		✓	512 b	Hamming	Alahi et al. [2012]
FRIF		✓	512 b	Hamming	Wang et al. [2013]

Table 2 – Overview of the feature descriptors presented in this chapter. In the column "dimension": a number alone represents a vector of that dimensionality and a number followed by "b" represents the length of the bit string. In column "distance": L2 is the Euclidean distance and KL is the Kullback Leibler divergence.

3 Experiments and Results

In Chapter 2 we presented a review of the most important feature detectors and feature descriptors proposed in the literature. We described their procedures and established several key differences between them, mainly based on their approaches and outputs. In this chapter we will perform an experimental comparison to study not only their computational efficiency, but also their performance in a common workflow. The organization of this chapter is as follows: first, we will introduce the experimental setup and the resources used throughout the experiments. Then, we will compare the most important feature detectors reviewed in the literature. This comparison consists in an execution time comparison and an evaluation of their quality using the well-known repeatability test. Finally, we will briefly present an evaluation of the effectiveness of feature descriptor in a target retrieval application.

3.1 Experimental Setup

3.1.1 Definitions

Repeatability Test: this procedure measures the quality of feature detectors and was proposed by Schmid et al. [2000]. They define: "The repeatability explicitly compares the geometrical stability of the detected interest points between different images of a given scene taken under varying viewing conditions". The repeatability can be measured by the repeatability rate, i.e. the percentage of points simultaneously present in two images (repeated). The regions that are repeated are commonly known as *covariant regions*. An example of covariant regions is showed in Figure 16.

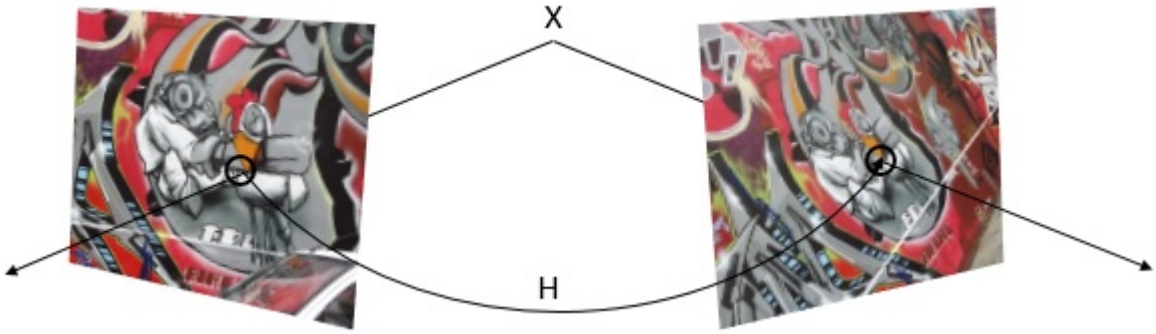


Figure 16 – Repeatability test: example of covariant regions.

Target Retrieval: as defined in 1.1.1, the target retrieval by image matching compares a query image against a set of candidates (image data set). The pipeline consists in several steps such as database feature extraction, query feature extraction, computation of distances and feature matching, voting&ranking and geometric verification.

The feature extraction (detection and description) of the images is performed once using the same algorithm for all images. The feature matching consists in a comparison of the features vectors of the query image against the feature vectors of the data set. This comparison is performed using some measure of distance and will be used to determine which feature vector of the data set is the nearest one to each feature vector of the query image. Then, each query feature vector will grant a vote (e.g. "+1") to their nearest data set feature vector. Finally, the images of the data set are sorted (ranked) according to their votes obtained. This process will return a list of the best ranked candidates among the images in the data set. Additionally, a geometric verification can be performed to eliminate some false positives (good ranked candidates, but without geometric consistency).

3.1.2 Resources

OpenCV (Open Source Computer Vision): among the implementations of feature detectors and feature descriptors available, we choose to use the open source computer vision and machine learning software library OpenCV. This decision was encourage by three major considerations: first, OpenCV give us the confidence to be working with reliable implementations (in various cases the code was contributed to OpenCV by the people who proposed the feature detector-descriptor). Second, OpenCV offers a variety of feature detectors-descriptors that are consistent with our experimental comparison needs. Third, OpenCV was designed to be cross-platform. Specifically, in 2010 OpenCV was ported to the Android environment and it allows to use the full power of the library in mobile applications development. Also, since 2012 it includes a full integration to iOS (since version 2.4.2).

In our experiments, we have used the version 2.4.6.1 (2013) which includes all the cross-platform capabilities. Table 3 shows the implementations used in the experimental comparison of feature detectors and feature descriptors.

Repeatability test: this test measures the performance of a detector by the repeatability criterion explained in subsection 3.1.1. To carry out the test, we have used the Matlab code[1]. This toolbox was developed and released in conjunction with the Oxford Affine Covariant Regions DataSet (presented in the next subsection). Then, both resources attain the condition to perform the repeatability test.

[1] Detectors Evaluation: `http://www.robots.ox.ac.uk/~vgg/research/affine/evaluation.html`

Detector	Descriptor
DoG (SIFT)	SURF
Fast-Hessian (SURF)	BRIEF
FAST	ORB
STAR (Censure)	BRISK
MSER	FREAK
AGAST (BRISK)	
oFAST (ORB)	

Table 3 – OpenCV implementations used in the experimental comparison.

Time Measure: to measure the execution time for each feature detector we choose to use the line_profiler[2] which is a module for doing line-by-line profiling of functions for Python.

Computational Resources: all experimental procedures were performed in a computer with Intel Core i7-2600 3.40GHz processor running Ubuntu 12.04 (64-bits).

3.1.3 Data Sets

Oxford Affine Covariant Regions DataSets [Mikolajczyk and Schmid, 2005]: is an image data set with an accurate homography (plane projective transformations) between images. Then, *ad hoc* data set for repeatability test. It includes 8 image sequences, each one containing a raw scene (reference) and 5 images with a gradual geometric or photometric transformation as showed in Figure 17. All images are of medium resolution, approximately 800× 640 pixels, and the increasing level of transformation can be obtained by one or more transformations as showed in Table 4.

Category	Transformations	Images
Bark	Scale and Rotation	6
Bikes	Blur	6
Boat	Scale and Rotation	6
Graffiti	Viewpoint Change	6
Leuven	Illumination	6
Trees	Blur	6
UBC	JPEG compression	6
Wall	Viewpoint Change	6

Table 4 – Sequences of images and their transformations in the Oxford Affine Covariant Regions DataSets.

Transformations as rotation, scale change, viewpoint change, and blur, are present in two different scenes. One structured scene (homogeneous regions with distinctive edge

[2] Line_profiler: `https://github.com/rkern/line_profiler`

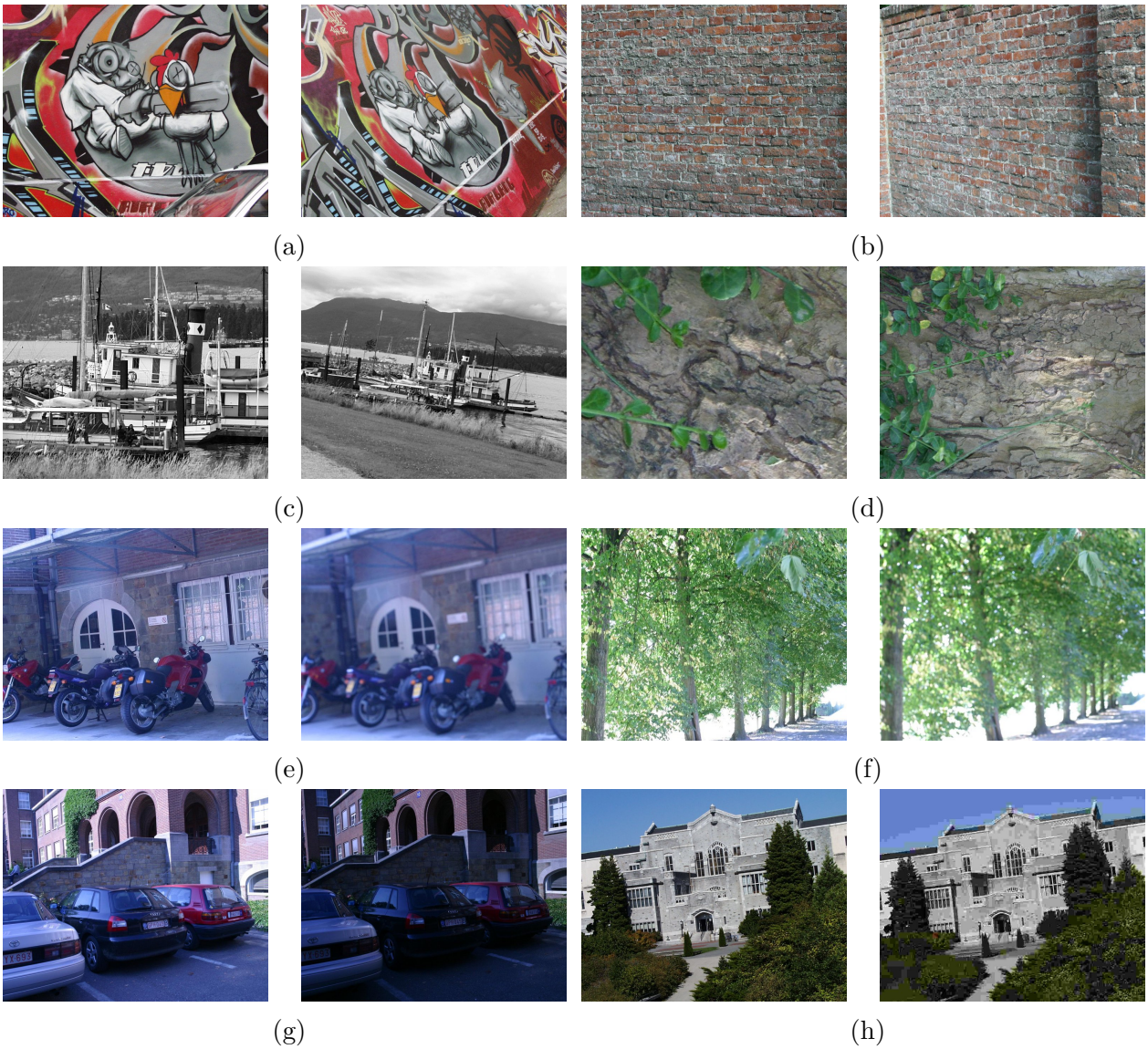


Figure 17 – Oxford Affine Covariant Regions Data set. Examples of reference images (left) and their transformations (right) for each image sequence: (a) and (b) viewpoint changes in structured scene and textured scene, respectively. (c) and (d) zoom+rotation in structured scene and textured scene, respectively. (e) Blur in structured scene and (f) Blur in texture scene. (i) Light changes, (j) JPEG compression.

boundaries), and the other is a texture scene. This is helpful to distinguish the influence of such transformations in each scene type separately.

Geometric transformations: image rotations are in the range of 30 and 45 degrees. The maximum scale changes are about a factor of four and are obtained by varying the camera zoom and viewpoint changes are obtained by varying the camera position. Photometric transformations: blur sequences are acquired by varying the camera focus. The light changes

are introduced by varying the camera aperture and the JPEG sequence is generated with a standard xv image browser with the image quality parameter varying from 40% to 2% percent.

The homographies between the reference image and the other images in a particular dataset are accurate computed. This means, the mapping function relating images is known and can be used to determine a ground truth matches for specific regions of interest obtained in the detection process. Then, this ground truth knowledge allow us to perform a repeatability test in this homography annotated data set.

SMVS—Stanford Mobile Visual Search Data Set [Chandrasekhar et al., 2011]: although at the time there are many image data set available, none of them was oriented to be used in mobile applications environment. The SMVS data set was released in 2011, this data set not only has the advantage to be obtained from low and high-end camera phones but also has several key properties that turn it into a good data set for mobile applications: the data set contains rigid objects, i.e. it is possible to estimate a transformation between reference image and query image. The images are captured within a wide range of lighting conditions and presents a foreground-background clutter. Finally, the data set has common perspective distortions (rotation, scale changes, viewpoint changes).

The SMVS data set has 3300 query images for 1200 distinct classes (objects) grouped into 8 categories as showed in Table 5. Due to the variety of categories, this data set can be used in a wide range of visual search applications like product recognition (CD, DVD, Books, etc), landmark recognition, augmented reality, text recognition or even video recognition. Examples of query and database images are shown in Figure 18.

Category	Database	Query
CD	100	400
DVD	100	400
Books	100	400
Video Clips	100	400
Landmarks	500	500
Business Cards	100	400
Text documents	100	400
Paintings	100	400

Table 5 – Number of query and database images in the SMVS data set for different categories.

Reference images are clean versions of images obtained from different sources: product websites (CDs, DVDs, books), data collected by Navteq's vehicle-mounted cameras (landmarks), website that mines the front pages of newspapers and research papers (text documents), high quality scan (business card) or the Cantor Arts Center at Stanford University

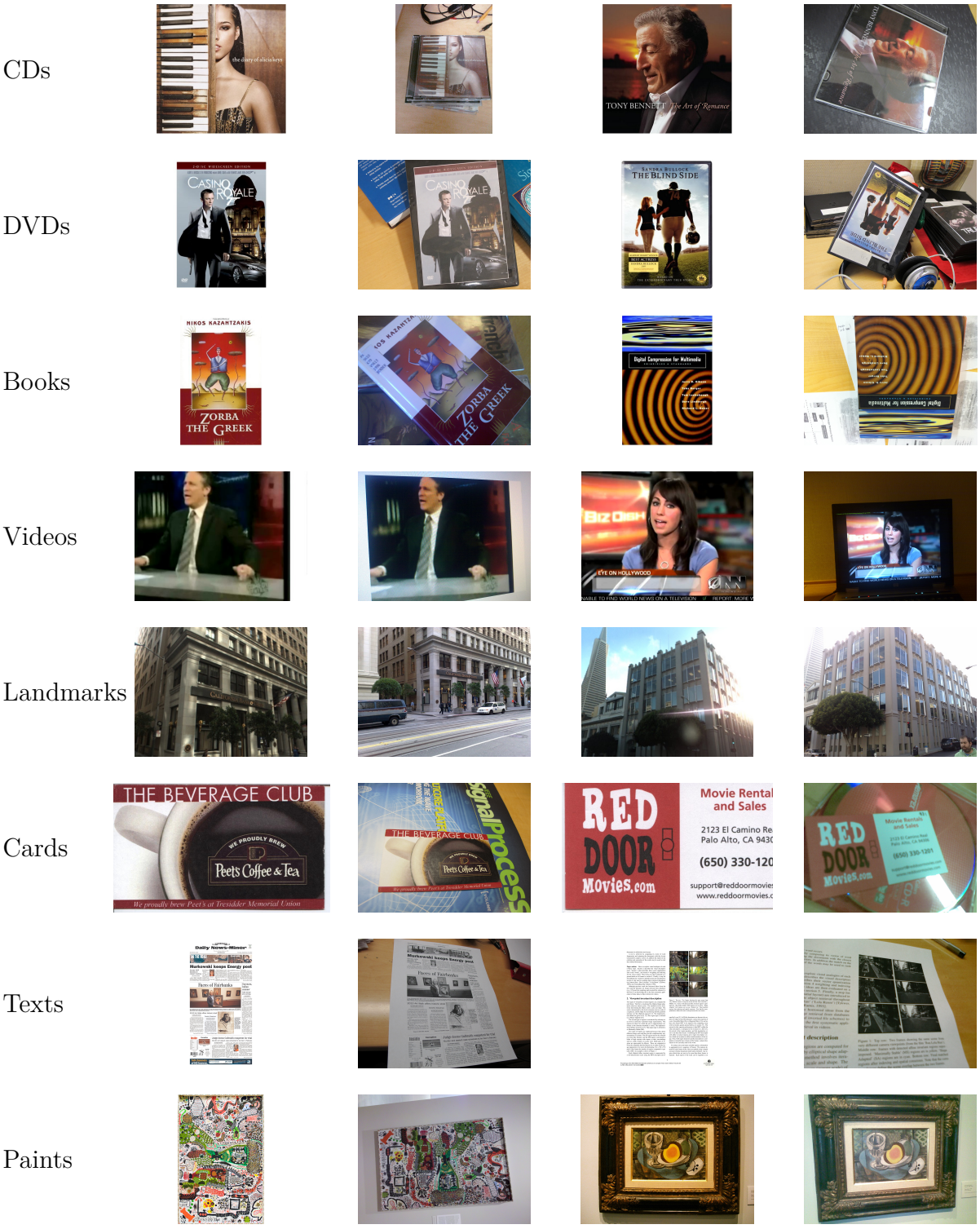


Figure 18 – Stanford Mobile Visual Search (SMVS) data set: consists of images for different categories captured with a variety of camera-phones and under widely varying lighting conditions. Each line in the figure shows 2 examples of reference image and 2 examples of a query images for an specific category.

for museum paintings. All reference images are high quality JPEG compressed color images and their resolution varies for each category. Query images were captured with several different camera phones, including some digital cameras. The list of companies and models used is as follows: Apple (iPhone4), Palm (Pre), Nokia (N95, N97, N900, E63, N5800, N86), Motorola (Droid), Canon (G11) and LG (LG300). Query images present wide variations in lighting conditions and foreground and background clutter. Also, for video clips, the query images were taken from laptop, computer and TV screens to include specular distortions. The resolution of the query images varies for each camera phone. The table 6 shows the query images available for each category. The Landmarks category only have one query image for each object and these query images do not correspond to a unique camera model.

Category	N5800	Canon	Droid	Iphone	E63	Palm	N97	N900
CD	–	✓	✓	–	✓	✓	–	–
DVD	–	✓	✓	–	✓	✓	–	–
Books	✓	✓	✓	✓	–	–	–	–
Video Clips	✓	–	–	✓	–	–	✓	✓
Business Cards	–	✓	✓	–	✓	✓	–	–
Text documents	–	✓	✓	–	✓	✓	–	–
Paintings	–	✓	✓	–	✓	✓	–	–

Table 6 – SMVS Data Set description: Each category and their correspondent query images. Check mark indicates when the image query is available for an specific category.

3.2 Feature Detector Evaluation

First of all, we have to mention that there is not a unique approach to evaluate feature detectors. The decision-making to use certain measures and tests in our procedure were highly influenced by the special attention we put to the trade-off between computational cost and performance, due to the specific context of mobile applications. To evaluate the performance of feature detectors we take into consideration three important aspects that will determine our outline procedure and scope. First, we will not consider the whole bunch of feature detectors presented in Section 2.1 because various of them are computationally inefficient to be considered good candidates for MVS applications. Then, we have to determine the minimum requirements for a feature detector to be considered as an appropriate candidate. Second, as feature detectors have a set of properties (e.g repeatability, locality, quantity, accuracy, etc), we have to find which of them will be considered more important for MVS applications. Finally, we have to establish a measure to evaluate the quality of feature detectors.

The organization of this section is as follows: initially, we describe our general framework to evaluate feature detectors. Then, we present a brief discussion to set the starting

point of feature detectors candidates. With this initial subset of candidates, we perform a comparison of time consumption, the results will help us to identify a smaller subset of good feature detector candidates. Lastly, we briefly discuss several methods to evaluate performance of feature detectors which will lead us to perform a repeatability test to determine the most suitable methods to be used in MVS applications.

3.2.1 Evaluation Workflow.

We briefly describe the evaluation procedure used. Our main goal with this evaluation is to identify a small set of fast and robust feature detectors to be used in MVS applications. Then, each step serves as a filter to reduce the number of candidates.

Step 1: Initial Selection. Due to the constraints of computing power and time consumption in mobile applications context, we select a subset of feature detector described in Chapter 2. This selection is based on a comparison of their computational efficiency.

Step 2: Time Comparison. In MVS applications, fast computation is a major requirement. Then, we perform a time consumption comparison. For this purpose, we select a subset of images from the *SMVS—Mobile Visual Search Data Set* [Chandrasekhar et al., 2011] to measure the time used in the computation of the detection process for each method. Finally, we compare the average time for each method in order to determine a subset of fast and therefore most suitable detectors for MVS applications.

Step 3: Quality Test. Whereas in the previous steps we were focused on fast computation, in this last step we will put focus on the feature detector's effectiveness. Initially, we briefly describe several methods used to evaluate the quality of feature detectors. Then, we perform the repeatability test on each method using the *Oxford Affine Covariant Regions DataSets* released by Mikolajczyk and Schmid [2005].

Both data sets were described in detail in Section 3.1.3.

3.2.2 Feature Detector Comparison

From all feature detectors studied in the Chapter 2 the most successful is the SIFT [Lowe, 2004], as we stated before, part of the importance of SIFT lies in its capability to be invariant to scale changes and its success being highly discriminant and robust to transformations. SIFT is also the first computationally efficient detector and descriptor. Then, since we are considering SIFT as a good candidate, we will discard feature detectors that came to light before SIFT (e.g. SUSAN, MIC, Salient Regions, IBR, Harris and Hessian detectors). In fact, after SIFT came to light, the main aim became to find a feature detector-descriptor

that perform as well as SIFT, but faster and cheaper. In this enterprise, the most remarkable contribution to compute a fast feature detector was proposed in 2008, SURF by Bay et al. [2008]. SURF uses a Hessian matrix approximation based on computing image integrals. Then, SIFT and SURF became standard methods for detection and description processes. After them, to validate any new contribution in feature detector-descriptor a comparison with SIFT and/or SURF on a common baseline is almost mandatory in practice. We will consider in this comparison the feature detectors grouped into optimized implementations (Section 2.1.3) because they are intended to be efficient to compute.

For any image retrieval application the execution time has a direct influence on the user experience and this is even more significant in mobile applications. Taking both, SIFT and SURF as reference, we perform an initial experiment to compare the average time of execution for each feature detector. Due to the computational resources constraints in a mobile environment, we have special interest in feature detectors that are faster and cheaper than SURF.

We compare the execution time of several feature detectors previously studied in section 2.1. The results obtained are presented in Table 7. Each "runtime" value represents the average time (in milliseconds) of feature detection for an image of the CD category from the SMVS data set. We choose to use the reference images (100 images) from the CD category, where all images have the same size (500×500 pixels).

	Runtime		Time Comparison	
Feature Detector	(ms)	# Keypoints	Respect to FAST	Respect to SURF
DoG (SIFT)	99.36	1426	53.7×	2.28×
Fast-Hessian (SURF)	43.46	1839	23.5×	1×
FAST	1.84	4655	1×	0.04×
STAR (CenSure)	6.08	306	3.3×	0.14×
MSER	19.60	261	10.6×	0.45×
AGAST (BRISK)	8.00	460	4.3×	0.18×
oFAST (ORB)	5.63	499	3×	0.13×

Table 7 – Results of execution time comparison on feature detectors in a CD category from SMVS data set. "Runtime" reports the average time (of 100 images) to compute the each feature detector. These processes were executed 10 times to guarantee their consistency. "Keypoints" shows the average number of keypoints obtained for each image. The two last columns show the time proportion between each method and the fastest one (FAST detector) and also to the *de facto* standard SURF feature detector, respectively.

Before we interpret the results and derive conclusions, we will briefly describe the experimental context. We choose to use an image subset from the MVS data set for this

experimental comparison because at the end we want to evaluate the performance of these feature detectors in a mobile visual system, i.e. evaluate the quality of feature detectors on images obtained by actual camera phones. On the other hand, we choose to use OpenCV (version 2.4.6.1) to perform the common comparison pipeline due to the several reasons explained in 3.1.2. SIFT and SURF detectors are commonly known as DoG and Fast-Hessian feature detectors and OpenCV has its own implementations that are consistent with the original ones, but not exactly the same. STAR feature detector is a slightly modification of CenSure detector. ORB and BRISK detectors correspond to an oriented FAST (oFAST) and adaptive FAST (AGAST), respectively. To measure the execution time, we choose to use the line_profiler[3] module which performs a line-by-line profiling of functions.

In Table 7 we show the average time detection for each method and their ratio relative to FAST. As expected, SIFT and SURF are expensive to compute (in time) and FAST is the fastest one. Because ORB and BRISK derive from FAST, they two are slower than FAST, but faster than SURF. STAR detector uses a generic integral image and performs almost 7× faster than SURF. Finally, MSER detector is faster than SURF and SIFT, but is almost 10× slower than FAST. Table 7 also reports the number of keypoints for each method. In general, as the number of keypoints increases so does the information to represent the image. This lead us to argue that small amount of keypoints will offer a poor representation of the image. Nevertheless, in practice, the number of keypoints tends to be larger than desired and has to be reduced by keeping only the bests. To reduce or limit the number of features detected, a common approach is use a measure of robustness (e.g. cornerness measure described in Harris detector). In the OpenCV environment this measure is provided as a *response* output by which the strongest keypoints can be selected.

For further experiments, SIFT will no longer be maintained as a reasonable candidate because of its computational expensiveness, and we will establish SURF as a reference feature detector. In fact, is the common practice to consider SURF as a reference to evaluate efficient implementations [Rublee et al., 2011; Leutenegger et al., 2011]. Also, we no longer use MSER, because this method obtains a small amount of keypoints and is expensive to compute in comparison to the remaining feature detection candidates. Then, we have a SURF detector as reference of performance and FAST, STAR, BRISK and ORB as feature detector candidates. Until this point, we only have looked at the efficiency and computational cost of the feature detector candidates. In the next subsection we will go into the comparison of the effectiveness of feature detectors using the well known repeatability test, explained in 3.1.1.

[3] Line_profiler: `https://github.com/rkern/line_profiler`

3.2.3 Repeatability Test

Over the years, several methods were proposed in the effort to measure the quality and robustness of feature detectors. This measure of quality has a strong relationship with the properties expected in a good feature detector and even with the feature detected itself (regions, blobs, corners). Among these methods, the most popular are those based on ground truth verification and visual inspection, but both methods are restricted to the human interpretation of the image since the ground-truth is created by a human and the visual inspection methods depend on the human evaluation. The successful of these methods and the complexity of the images used are limited due to their inherently subjective interpretation of the image. Other methods to evaluate feature detectors are based on a theoretical analysis. These methods use theoretical models to examine the behavior of detectors, but unfortunately, they can only be used with a small set of very specific features. Another non general methods are focused on the location accuracy and how well the features are prepared to be used in specific tasks (camera calibration, $3D$ reconstruction), but they cannot be generalized.

Looking to the features characteristics: features detected should be local and accurate to reduce the probability of occlusion and also to allow the approximation of the geometric deformation. The number of features detected should be sufficient to provide a good image representation and features should be robust to geometric and photometric transformations. Finally, arguably the most important property, feature should have high repeatability, this means, for two images of an specific scene obtained under different viewing conditions, a high percentage of the features detected should be detected in both images. The repeatability does not depends on an specific feature and also is not limited by a human interpretation as the approaches mentioned before. Repeatability test was proposed by Schmid et al. [2000] to evaluate the performance of interest point detectors. Since then, it became the most meaningful measure to evaluate feature detectors. Mikolajczyk and Schmid [2005] released the *Oxford Affine Covariant Regions DataSets*[4] and the Matlab code[5] to carry out the performance test. The precomputed homography matrices between reference image and images under transformations were also released with the data set. In our experimental evaluation, we use the previously selected feature detector candidates: SURF, FAST, BRISK (AGAST), ORB (oFAST) and STAR (CenSure).

To perform the repeatability test we set a maximum number of keypoints (1000) for each feature detector in order to perform a fair comparison, i.e. if the feature detector returns more keypoints, only the best 1000 are retained based on their *response* measure. The released Matlab code sets an overlap error threshold of 40%. The overlap error measures how well

[4] Image Dataset: `http://www.robots.ox.ac.uk/~vgg/research/affine/`
[5] Detectors Evaluation: `http://www.robots.ox.ac.uk/~vgg/research/affine/evaluation.html`

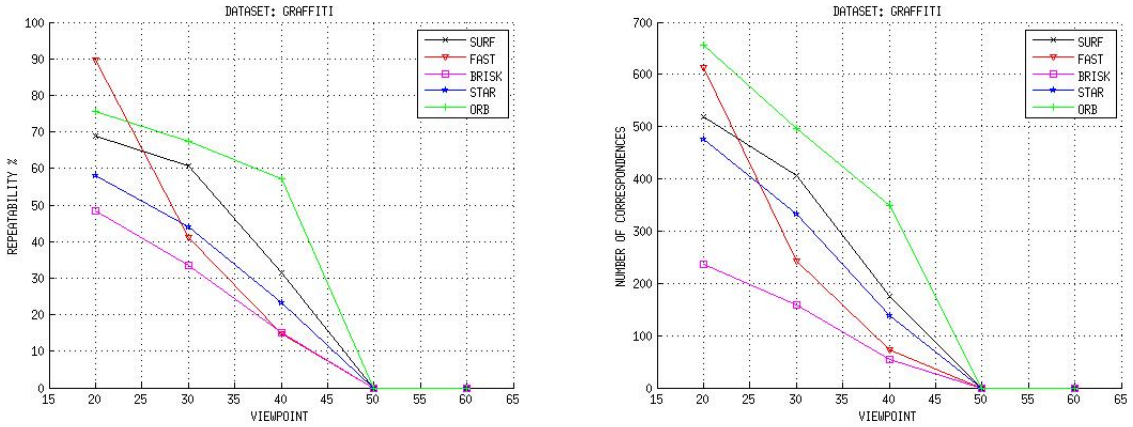


Figure 19 – Viewpoint change for structured scene: Graffiti data set. (left) Repeatability score for viewpoint changes. (right) Number of corresponding regions.

the regions correspond under a transformation (homography). It is defined by the ratio of the intersection and union of the regions. If the overlap error is bigger than the threshold the regions will not be considered convariant (repeated). Finally, the results of the test are expressed in percentages of repeatability (relative repeatability) and the actual number of correspondences between the reference image and the other images in the sequence (absolute repeatability).

We will present and discuss the repeatability test results by grouping the data sets as follows: graffiti and wall (viewpoint changes), bark and boat (zoom and rotation), bikes and trees (blurring), leuven (light changes) and UBC (JPEG compression). Note that the three first transformations have an structured and texture image sequences to be evaluated.

The ideal case: all features detected within the part of the scene visible in both images should be detected in both images (reference image and each other image in the sequence). This means, the ideal result for repeatability corresponds to an **horizontal line at 100%** in the figure (relative repeatability vs degree of transformation), but this cannot be accomplished in practice. In general, the repeatability decreases as the transformation becomes more severe as showed in Figures 19 to 26.

Viewpoint change: the effect of changing viewpoint is evaluated in two different image sequences: graffiti (structured scene) and Wall (texture scene). For both cases the reference image correspond to a frontal scene and the other 5 images are obtained under a viewpoint change of 20, 30, 40, 50 and 60 degrees. Graffiti sequence: as showed in Figure 19, in the structured scene ORB and FAST have high repeatability for small viewpoint changes, but FAST decreases rapidly in repeatability percentage and also in the actual number of corre-

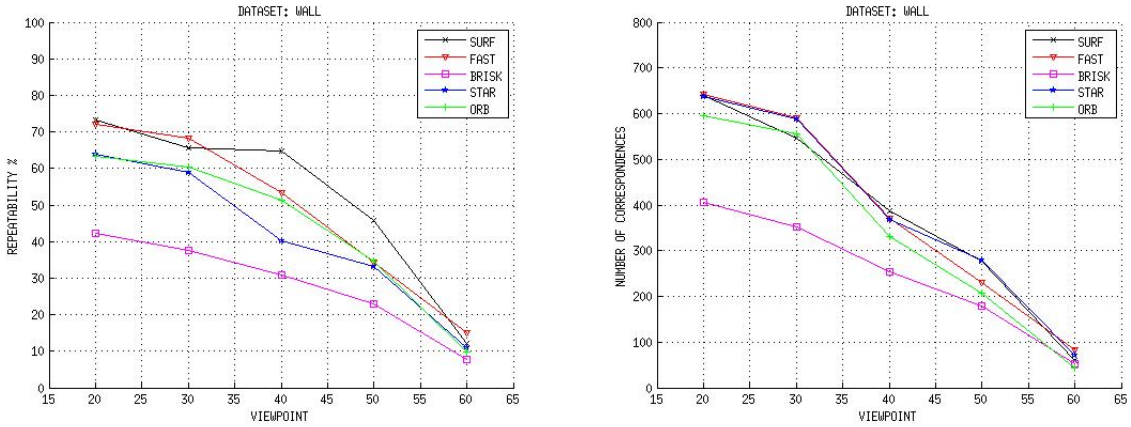


Figure 20 – Viewpoint change for texture scene: Wall data set. (left) Repeatability score for viewpoint changes. (right) Number of corresponding regions.

spondences while ORB continues to present the best repeatability percentages and highest number of correspondences. In fact, ORB detector has better repeatability than the reference SURF detector for all images in the sequence. BRISK detector (AGAST) presents the lowest repeatability and also have small values of correspondence. Wall sequence: The results of the evaluation of viewpoint changes for a texture scene are showed in Figure 20. The reference detector (SURF) performs better than the other methods, but FAST, ORB and STAR have similar results in repeatability percentage and number of correspondence. BRISK, similarly to the structured scene case, presents the worst repeatability values throughout the viewpoint changes.

The repeatability scores for the initial viewpoint change (20 degrees) are between 48% and 90% for graffiti and between 42% and 74% for wall sequence. These scores decrease quickly for larger viewpoint angles and the number of correspondences present a similar behavior. Looking to these results we can also derive that viewpoint changes modify strongly the image because the repeatability goes down rapidly while increasing the viewpoint angle. As we can see for graffiti sequence, independently of the detector used, the repeatability goes to zero for 50 degrees while for wall it goes to 10% at the most severe viewpoint change.

Scale change: the sequences used to evaluate scale changes have also in-plane rotations. The two image sequences are: boat (structured scene) and bark (texture scene). The maximum scale change for boat scene is about a factor of 2.8 and about 4 for bark sequence. Boat sequence: as showed in Figure 21, in the structured scene the repeatability of FAST and BRISK drop down really quickly while ORB and SURF have better results and also have a similar behavior as the scale grows. STAR detector have the lowest repeatability score at the beginning, but as the scale change gets bigger, it is not affected to much and its

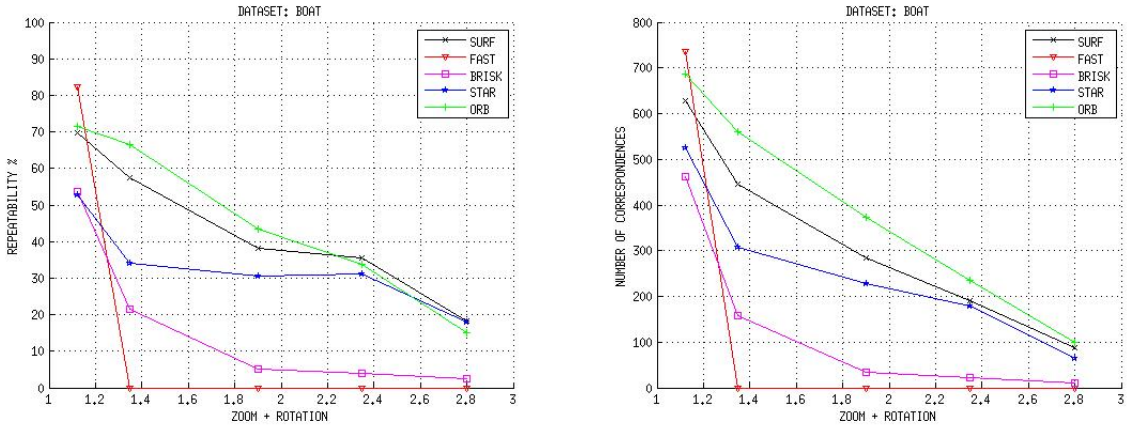


Figure 21 – Scale change and rotation for structured scene: Boat data set. (left) Repeatability score for viewpoint changes. (right) Number of corresponding regions.

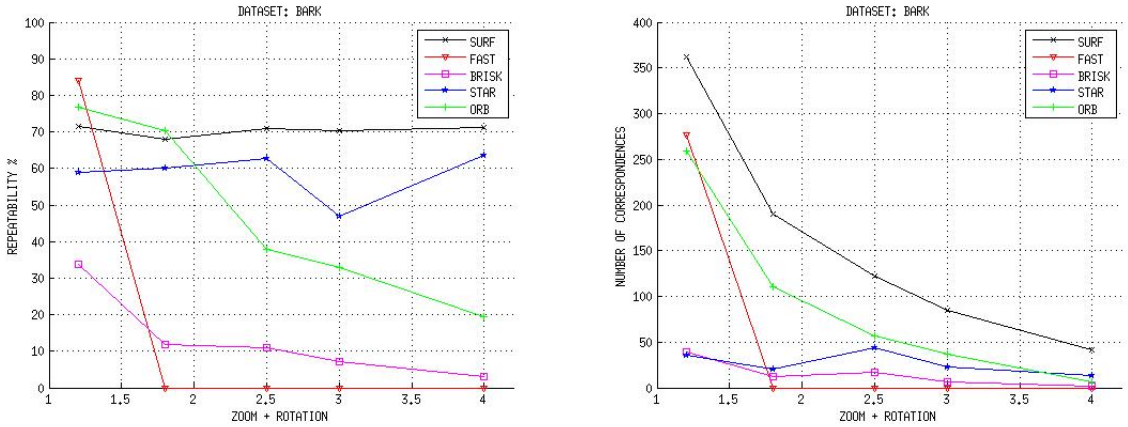


Figure 22 – Scale change and rotation for texture scene: Bark data set. (left) Repeatability score for viewpoint changes. (right) Number of corresponding regions.

behavior is almost horizontal. Bark sequence: in Figure 22, FAST drops down quickly while SURF outperforms other methods and has an quasi-horizontal behavior. ORB repeatability decreases as the scale change gets bigger and STAR detector presents an oscillating behavior. The number of actual correspondences is very small for STAR and BRISK detectors in all cases while for SURF detector the number of correspondences starts with almost 350 and decreases as the scale grows.

An interesting observation is that even if STAR detector presents good repeatability percentages, this not imply an actual good performance of the detector. This can be perceived looking at the actual number of correspondences for STAR detector and realize that this number is always small (under 50) and as we stated before, the number of features should

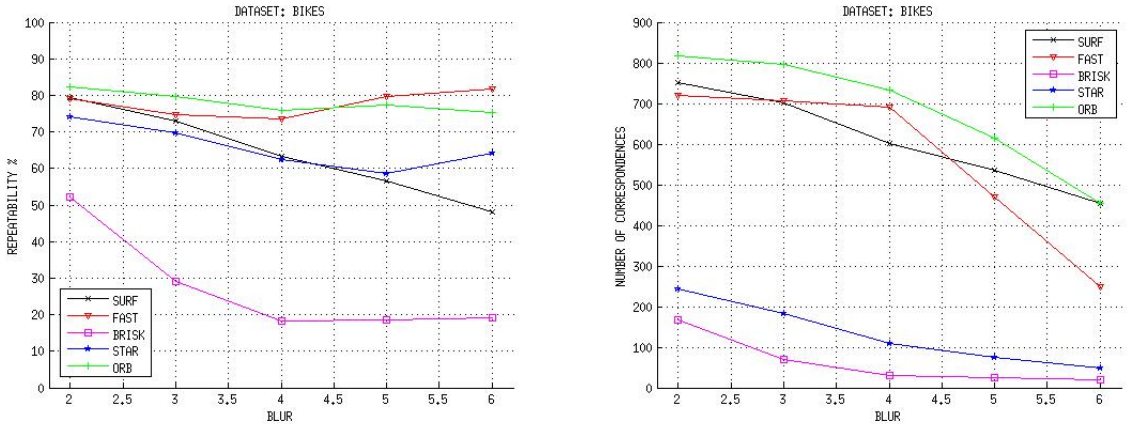


Figure 23 – Blur for structured scene: Bikes data set. (left) Repeatability score for viewpoint changes. (right) Number of corresponding regions.

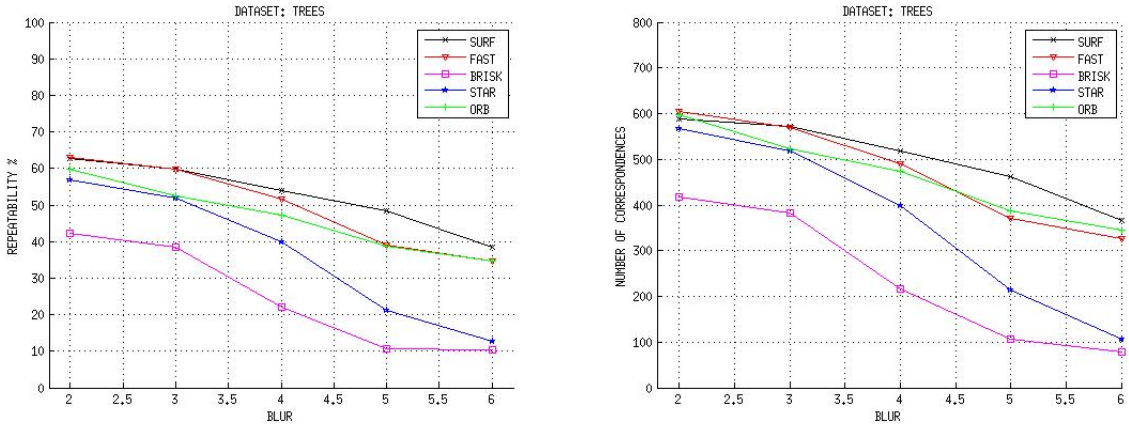


Figure 24 – Blur for texture scene: Trees data set. (left) Repeatability score for viewpoint changes. (right) Number of corresponding regions.

be big enough to provide a good representation of the image. Then, STAR has less than 50 common features (with correspondence) over a 1000 possible is not a good result of repeatability.

Blur: the effect of blurring an image is evaluated in two different image sequences: Bikes (structured scene) and Trees (texture scene). Bikes sequence: Figure 23 shows the results for the structured bikes scene. ORB and FAST have the better repeatability percentages and share a similar quasi-horizontal behavior. The repeatability score for SURF and STAR slightly decrease as the blur degree increases, but the number of correspondences of STAR is lower than the other three methods. Finally, the score and number of correspondences for BRISK result the smallest for all blurred images. Trees sequence: the results for this test

are showed in Figure 24. Again, BRISK losses against the other methods. ORB, FAST and SURF have a similar results and behavior in both cases (repeatability score and number of correspondence).

Until this point, we have already evaluated the performance of our feature detectors under three different transformations (two geometric transformations: scale+rotation and viewpoint change and one photometric transformation: blur) and using two types of scenes (structured scene and textured scene). So far, we can extract some important and non-explicit consequences of our results: for any sequence of textured scene evaluated, the SURF detector outperforms the rest of feature detectors compared, i.e. SURF detector is effective and robust to detect features at small scales under viewpoint changes, scale and blur transformations. But, for structured scenes SURF is partially surpassed by ORB detector or even FAST in the case of bikes sequence. FAST detector goes almost straight to zero repeatability under scale changes and this reveal its lack of scale invariance. The sequence with blurring images presents better results than the previous sequences. Then, we can infer that photometric transformations have less impact in the representation of the image, since the repeatability does not decrease as much as for geometric transformations. Also, two common characteristics along these results can be summarized: First, BRISK detector returns features with low repeatability and is outperformed by every other method used in the evaluation. Second, even if STAR detector obtains similar results than ORB and SURF, it generally lost its value due to the fewer number of correspondences. Then, till now, FAST and ORB are better candidates.

Light change: Figure 25 shows the repeatability score and the absolute number of correspondences obtained for Leuven sequence under illumination changes. The repeatability curves as well as the correspondence curves are nearly horizontal for all feature detector evaluated. This reveals a good degree of robustness to illumination changes. Since all methods obtain similar repeatability scores we have to use the total number of correspondence in order to distinguish which one of the candidates is better. Then, we can identify, in general, that FAST and ORB are better candidates than BRISK and STAR even when ORB detector has slightly less repeatability score than the others.

JPEG Compression: Figure 26 shows the scores for the structured UBC sequence (large homogeneous areas and distinctive corners) under JPEG compression transformation. The repeatability curves are almost horizontals and evidence the robustness of the candidates under JPEG compression. BRISK and STAR are clearly outperformed by the other candidates. Overall, the scores for JPEG compression are the bests (from 90% to 70% along the degrees of JPEG compression present in the image sequence).

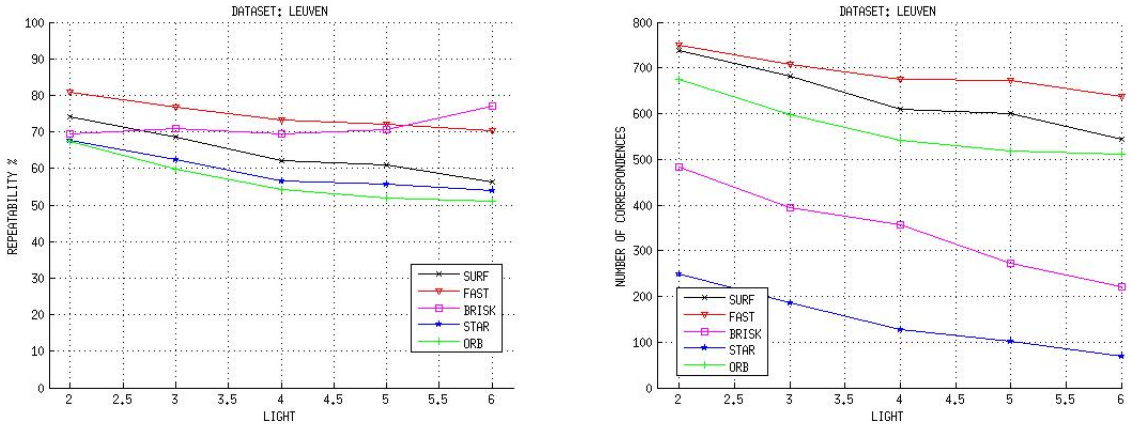


Figure 25 – Illumination change for Leuven data set. (left) Repeatability score for viewpoint changes. (right) Number of corresponding regions.

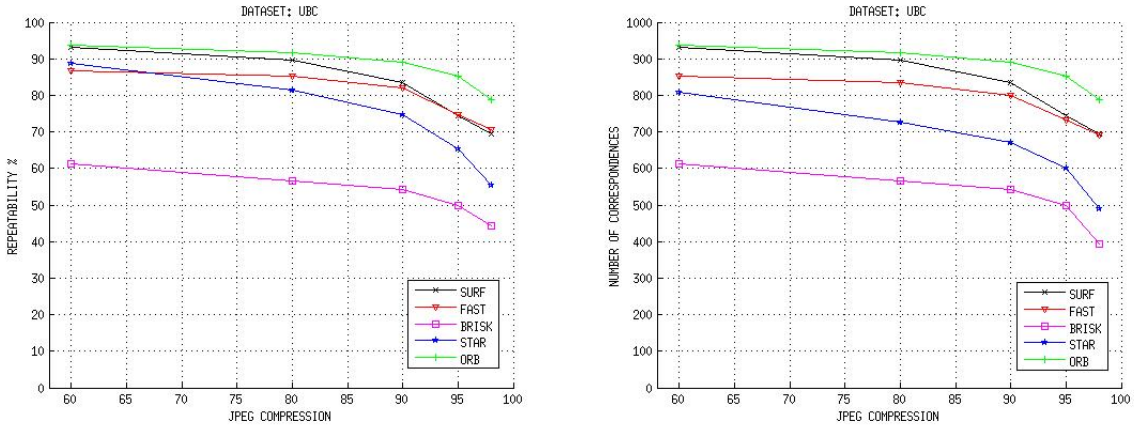


Figure 26 – JPEG compression for UBC data set. (left) Repeatability score for viewpoint changes. (right) Number of corresponding regions.

3.2.4 Summary

As we initially stated, in image retrieval applications, SIFT detector is a standard of performance, but in the context of mobile applications the major constraints of time and computational power lead us to use SURF feature detector as standard of comparison throughout the experiments. Then, we use SURF to select a set of candidates that are cheaper to compute and faster than SURF. We also set SURF as the reference of performance in the repeatability test. This allow us to compare the performance of the remaining candidates against SURF in different scenarios (various transformations).

SURF detector obtains better repeatability results for texture scenes in comparison to structured scenes. This reflects its capability to detect features at small scales. On the

other hand, in structured scenes, ORB performs better than the other feature detectors and evidently is good for detecting features at bigger scales. The last two photometric transformations reinforce previous observations: feature detectors, in general, have higher level of robustness to photometric transformations (illumination) and signal degradation (blur and JPEG compression) than geometric transformations (rotation, scale, viewpoint). Second, BRISK detector have poor performance for all transformations evaluated. Third, ORB and FAST are the best candidates, even though FAST performs poorly for scale change transformations.

The Table 8 summarizes the main observations for each transformation evaluated.

Transformation(s)	Category	Detectors sorted by their repeatability %
Viewpoint Change	Graffiti	1. ORB, FAST 2. SURF 3. STAR 4. BRISK
Viewpoint Change	Wall	1. FAST 2. SURF 3. ORB 4. STAR 5. BRISK
Scale and Rotation	Boat	1. ORB 2. SURF 3. STAR 4. BRISK 5. FAST
Scale and Rotation	Bark	1. ORB 2. SURF 3. STAR 4. FAST 5. BRISK
Blur	Bikes	1. ORB 2. FAST 3. SURF 4. STAR 5. BRISK
Blur	Trees	1. SURF, FAST 2. ORB 3. STAR 4. BRISK
Illumination	Leuven	1. FAST 2. BRISK 3. SURF 4. STAR 5. ORB
JPEG compression	UBC	1. ORB 2. SURF 3. FAST 4. STAR 5. BRISK

Table 8 – Results obtained using the repeatability test: we choose to put two or more detectors in the same rank when there is no absolute dominance of one of them in the interval evaluated. In viewpoint changes we are considering changes until 30 degrees to determine the order of feature detectors evaluated. In scale changes, we put FAST at the end because this detector drops quickly when the transformation become bigger. In Blur changes, the result of BRISK is drastically worst than other feature detectors. In illumination changes, the order presented is less meaningful because all detectors have good values of repeatability.

3.3 Feature Descriptor Evaluation

While in the previous section we have performed an experimental procedure to determine the most suitable detectors for mobile visual applications. In this section we will present an experimental procedure to compare the results in a target retrieval experiment. Similar to the case of feature detection evaluation, there is not a unique approach to measure the effectiveness of feature descriptors. For this experimental comparison, we will use the SMVS data set. This data set will help us to figure out the performance of feature descriptors in a mobile devices environment because their images were obtained by different sources of camera phones.

Note: the procedure and results presented below are still incomplete. We have decided to include this content to show some interesting results and possible extensions of this work. The inclusion of this section was also encouraged by the judging committee.

3.3.1 Evaluation Workflow.

We briefly describe the evaluation procedure used in this section. Our main goal with this evaluation is to identify the best combination of detection-descriptor to be used in MVS applications.

Step 1: Database Description. We compute and store the feature vectors for the complete database using each feature detector-descriptor combination. We also create an index of database images and their relative feature vectors.

Step 2: Query Description. We compute and store the feature vector for each one of the query images using all combinations of detector-descriptor considered plausible to be used in mobile application context.

Step 3: Matching. We execute a brute-force matcher between query image and database images for each detector-descriptor combination.

Step 4: Voting. Using the list obtained in the previous process and the index generated during the database description, we are able to attribute votes to the database images. A vote is assigned to a certain data set image whenever one of their feature vectors match (is the nearest one) to one feature vector of the query image.

Step 5: Ranking. Finally, for each query image, we are able to create a sorted list of the most voted database images.

The first two steps are execute once (offline process) for each detector-descriptor combination. We store this feature vectors to be able to use an efficient Hamming distance algorithm implemented in OpenCV. In our implementation, we set the simplest scheme for voting and ranking. Each query feature vector votes only for their nearest data set feature vector. The ranking is performed by summing the votes for each image in the data set (candidates) and sorting them by their number of votes. We consider as a correct target retrieval (success) if the correct image is within the 10 top ranked candidates.

3.3.2 Retrieval Experiments

We chose to use the workflow presented before because this procedure give us an actual measure of the quality of the detector-descriptor combination in a real target retrieval task with images obtained by camera phones. In the experimental comparison we will use the

detector-descriptor combinations presented in Table 9. These combinations include the three feature detectors selected in Section 3.2: FAST, oFAST(ORB), SURF(Fast-Hessian). For the description stage we will use the highly compact (low bit-rate) binary feature descriptors studied in Section 2.2.2: BRIEF, ORB, BRISK and FREAK.

Detector	Descriptor	Maximum # of feature vectors
FAST	BRIEF	1000
FAST	ORB	1000
FAST	BRISK	1000
FAST	FREAK	1000
oFAST (ORB)	BRIEF	1000
oFAST (ORB)	ORB	1000
oFAST (ORB)	BRISK	1000
oFAST (ORB)	FREAK	1000
Fast-Hessian (SURF)	BRIEF	1000
Fast-Hessian (SURF)	ORB	1000
Fast-Hessian (SURF)	BRISK	1000
Fast-Hessian (SURF)	FREAK	1000

Table 9 – Feature detector-descriptor combinations. We use the three feature detector: FAST and oFAST (ORB) was selected as a best candidates in Section 3.2 and the reference SURF detector. The feature descriptors compared are the four binary feature descriptors.

The target retrieval experiments were performed using the *book* category of the SMS data set described in 3.1.3 and the results were expressed as the percentage of correct retrieval (success: correct image within the top 10 ranked candidates). Table 10 shows the results for FAST, ORB and SURF (fast-Hessian) detectors used in combination with the four feature descriptors. SURF descriptor and SURF detector was also evaluated to establish a reference.

These results show a good effectiveness of the SURF detector when combined with BRISK and FREAK descriptors. These combinations obtain similar results to the SURF-SURF (reference). For ORB detector the combinations with BRISK and FREAK also have good results. This is an evidence that the combination of two highly efficient detector-descriptor as ORB-BRISK and ORB-FREAK can obtain similar results (effectiveness) than the SURF detector-descriptor, but much more faster and using less computational resources.

Detector	Descriptor	Query: 5800	Query: Canon	Query: Droid	Query: iPhone
FAST	BRIEF	15.8	17.8	24.7	29.7
FAST	ORB	18.8	14.8	15.8	33.6
FAST	BRISK	15.8	13.8	12.8	33.6
FAST	FREAK	22.7	17.8	17.8	56.4
oFAST (ORB)	BRIEF	17	15.8	15.8	29
oFAST (ORB)	ORB	85	24	69	92
oFAST (ORB)	BRISK	97	20	71	94
oFAST (ORB)	FREAK	61	20	61	85
Fast-Hessian (SURF)	ORB	27.7	15.8	22.7	64.3
Fast-Hessian (SURF)	BRISK	91	82.1	93	97
Fast-Hessian (SURF)	FREAK	85.1	77.2	89	96
Fast-Hessian (SURF)	SURF	90	84.1	94	98

Table 10 – Results for feature detector-descriptor combinations. The queries correspond to images obtained using the cameras 5800, canon, droid and iPhone, respectively. The numbers express the percentage of correct retrieval using the book category from the SMS data set.

4 Conclusions

Throughout this work, we have explored several topics related to Mobile Visual Search. We mainly explore the state of the art of feature detectors and feature descriptors, establishing a line of evolution over the years in terms of performance, invariance and efficiency. In this final chapter we will highlight the key content presented throughout this work and comment some inherent implications. We will also present ongoing works and possible future works.

4.1 Lessons Learned

First and foremost, we have to highlight the satisfaction it is to work in this field of computer vision (image understanding). The active and highly productive community in this area over the latest years turned our work educational as well as pleasant. The constant improvement of algorithms and advances of techniques, have been leading this field to a deeper understanding of images, human visual system and how humans perceive and interpret images. The success of this field is not limited to the academic community. Over the last years, we have seen the emergence of companies focused in the visual search market, some of them mentioned in Chapter 1.

Throughout this work, we have experienced an interesting phenomena of interaction with the state of the art (back to the 50's) and the huge amount of new content in this area. As we observed before, binary descriptors show to be very compact (low bit-rate) compared to the traditional state of the art. These compact descriptors achieve reasonable good results by using few amount of bits. Nevertheless, there is a trade-off between performance and bit-rate to be considered.

The whole image retrieval process can be divided into various steps using several approaches: **computational processes**: detection, description and query search (matching). **Application status**: offline, online (query search). **Theoretical**: Low-level, Mid-level, High-level (semantic interpretation). The more extensive way to characterize a retrieval process consists in the last approach mentioned. So far, the content we have explored is mainly concentrated in the low-level interpretation of images. This is an important observation to positioning our work into the general and vast context of retrieval by using image content.

The main computational load that affect the user experience lies in the online search process. The time to compute this process corresponds to the time used to compute a comparison between query feature vectors and data set reference feature vectors, this impacts directly

the actual user interaction. This search time has a direct relationship with the amount of data to be processed, i.e. if the bit-rate of the feature descriptor increases and/or the number of images in the data set increases, then the cost to compute the search time increases as well. One way to avoid this problem is the use of compact descriptors (low bit-rate), as we have seen through this work, the use of binary descriptors provides a compact representation of an image.

As stated before, the search process can be extremely expensive (slow). Then, we have to explore a little bit more the search process. From the descriptors studied in chapter 2, the standard SIFT/SURF descriptor utilize the L2 norm to perform a comparison process between feature vectors (one from the query and one from the database). CHoG goes forward and evaluates three different measures to compute distance: Euclidean (L2-norm), Earth Mover's Distance (EMD) which is a total cost to transform one histogram into another and Kullback-Leibler divergence (KL), a bin-by-bin distance commonly used to measure the difference between two probability distributions. Overall, CHoG shows that for normalized histograms KL divergence outperforms the other two alternatives. Binary descriptors return a string of bits as feature vector and the computation of distance is performed just by using a XOR operation followed by a simple population count or the total number "1"s (Hamming distance). Then, binary descriptors overall, not only present a fast algorithm to detect and describe features, but also are far more efficient for the search process.

From all feature detectors evaluated, FAST and ORB are the two best alternatives to be used in mobile applications because they are highly efficient without greater loss of repeatability (trade-off of efficiency and performance). BRISK and FREAK feature descriptors in combination with the ORB detector show to be effective in a target retrieval application (obtaining good results, similar to the SURF). On the other hand, these detector-descriptor combinations offer an efficient use of computational resources in processing and storage because they are **binary** and **compact**. The distance between two feature vectors can be computed using the Hamming distance, which is more efficient than the use the L2 distance or the KL divergence. The amount of data to be stored is also drastically improved because these binary descriptors provide a compact representation of the feature vector, i.e. a string of bits (commonly 128, 256 and 512 bits) against the common SIFT feature vector of 128 dimensions ($128 \times 8 = 1024$ bits, considering 8 bits per each *int* value in the vector).

Finally, for MVS application, the feature detector-descriptor used must be **efficient** to compute, **effective** in target retrieval tasks and **compact** to be able to work with big amounts of data. These requirements are accomplished by the use of fast and efficient feature detectors (FAST, ORB) and binary and compact "low bit-rate" descriptors (BRISK, FREAK).

4.2 Future Work

We are working in a more exhaustive and detailed survey of binary feature descriptors including an actual theoretical analysis to determine the complexity and cost of each algorithm. We are using a similar protocol presented in this dissertation to perform the experimental comparison between feature detectors-descriptors and will include an extended experimental procedure to evaluate the performance in matching and retrieval processes. This work will also be implemented to perform experiments with big data sets (big scale experiments). Looking to mobile applications, we will implement a target retrieval application in an actual mobile device. For this purpose we will take advantage of the cross-platform design of OpenCV.

A possible further work is related to the study and use of Hamming distance computation in current hardware architectures. This can allow us to use some build-in instructions to accelerate this process of comparison between binary feature vectors. As we know, modern Intel processors (i5, i7) have a set of instructions SSE 4.2 (optimized popcount).

Bibliography

Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision–ECCV 2008*, pages 102–115. Springer, 2008. Cited 4 times in pages xiii, 14, 16, and 19.

Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. Ieee, 2012. Cited 4 times in pages xiv, 2, 31, and 34.

Fred Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3): 183, 1954. Cited 2 times in pages 7 and 8.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. Cited 9 times in pages xiii, 2, 14, 15, 16, 19, 23, 34, and 43.

Paul Brasnett and Miroslaw Bober. Robust visual identifier using the trace transform. In *Proc. of IET visual information engineering conference (VIE)*, 2007. Cited in page 2.

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010*, pages 778–792. Springer, 2010. Cited 5 times in pages xiv, 2, 28, 29, and 34.

Vijay Chandrasekhar, Mina Makar, Gabriel Takacs, David Chen, Sam S Tsai, Ngai-Man Cheung, Radek Grzeszczuk, Yuriy Reznik, and Bernd Girod. Survey of sift compression schemes. In *Proc. Int. Workshop Mobile Multimedia Processing*. Citeseer, 2010a. Cited in page 24.

Vijay Chandrasekhar, Yuriy Reznik, Gabriel Takacs, David Chen, Sam Tsai, Radek Grzeszczuk, and Bernd Girod. Quantization schemes for low bitrate compressed histogram of gradients descriptors. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 33–40. IEEE, 2010b. Cited 2 times in pages 26 and 64.

Vijay Chandrasekhar, Gabriel Takacs, David M Chen, Sam S Tsai, Yuriy Reznik, Radek Grzeszczuk, and Bernd Girod. Compressed histogram of gradients: A low-bitrate descriptor. *International Journal of Computer Vision*, 96(3):384–399, 2012. Cited 8 times in pages xiv, xv, 2, 20, 24, 25, 34, and 67.

Vijay R Chandrasekhar, David M Chen, Sam S Tsai, Ngai-Man Cheung, Huizhong Chen, Gabriel Takacs, Yuriy Reznik, Ramakrishna Vedantham, Radek Grzeszczuk, Jeff Bach, et al. The stanford mobile visual search data set. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 117–122. ACM, 2011. Cited 2 times in pages 39 and 42.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. Cited 2 times in pages 22 and 34.

Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. Cited in page 4.

Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94 (3):335–360, 2011. Cited in page 7.

Bernd Girod, Vijay Chandrasekhar, David M Chen, Ngai-Man Cheung, Radek Grzeszczuk, Yuriy Reznik, Gabriel Takacs, Sam S Tsai, and Ramakrishna Vedantham. Mobile visual search. *Signal Processing Magazine, IEEE*, 28(4):61–76, 2011. Cited 4 times in pages xiii, 3, 4, and 24.

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988. Cited 2 times in pages 8 and 19.

Kota Iwamoto, Ryota Mase, and Toshiyuki Nomura. Bright: A scalable and compact binary descriptor for low-latency and high accuracy object identification. In *ICIP*, pages 2915–2919, 2013. Cited 3 times in pages xiv, 27, and 34.

Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001. Cited 2 times in pages 12 and 19.

Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011. Cited 8 times in pages xiv, 2, 18, 19, 20, 30, 34, and 44.

Tony Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994. Cited 2 times in pages 9 and 10.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. Cited 9 times in pages 2, 4, 14, 16, 19, 21, 34, 42, and 69.

Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Computer Vision–ECCV 2010*, pages 183–196. Springer, 2010. Cited 3 times in pages 18, 19, and 30.

Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004. Cited 2 times in pages 12 and 19.

Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004. Cited 4 times in pages 8, 10, 12, and 19.

Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005. Cited 7 times in pages 2, 22, 24, 34, 37, 42, and 45.

Spiros Nikolopoulos, Stavri G Nikolov, and Ioannis Kompatsiaris. Study on mobile image search. *NEM Summit: Implementing Future Media Internet*, 2011. Cited in page 1.

Yuriy A Reznik, Vijay Chandrasekhar, Gabriel Takacs, David M Chen, Sam S Tsai, and Bernd Girod. Fast quantization and matching of histogram-based image features. In *Proc. of SPIE Vol*, volume 7798, pages 77980L–1, 2010. Cited 2 times in pages 26 and 64.

Jos BTM Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae*, 41(1):187–228, 2000. Cited in page 12.

Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006. Cited 4 times in pages xiii, 14, 17, and 19.

Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, 2010. Cited in page 18.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011. Cited 6 times in pages 2, 18, 19, 29, 34, and 44.

Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172, 2000. Cited 3 times in pages 7, 35, and 45.

Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003. Cited in page 4.

Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997. Cited 4 times in pages 8, 9, 17, and 19.

Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. Cited 2 times in pages 23 and 34.

Miroslav Trajković and Mark Hedley. Fast corner detection. *Image and vision computing*, 16 (2):75–87, 1998. Cited 3 times in pages 9, 17, and 19.

Tomasz Trzcinski, Mario Christoudias, Pascal Fua, and Vincent Lepetit. Boosting binary keypoint descriptors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2874–2881. Ieee, 2013. Cited 3 times in pages 2, 26, and 34.

Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280, 2008. Cited 2 times in pages 7 and 11.

Tinne Tuytelaars and Luc J Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *BMVC*, volume 412, 2000. Cited 3 times in pages xiii, 13, and 14.

Zhenhua Wang, Bin Fan, and Fuchao Wu. Frif: Fast robust invariant feature. 2013. Cited 2 times in pages 32 and 34.

Simon Winder, Gang Hua, and Matthew Brown. Picking the best daisy. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 178–185. IEEE, 2009. Cited 5 times in pages 2, 20, 23, 24, and 74.

Simon AJ Winder and Matthew Brown. Learning local image descriptors. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. Cited 3 times in pages 2, 23, and 25.

Appendix

APPENDIX A – CHoG: detailed study

The CHoG configuration that performs slightly better than SIFT correspond to 9-dimensional spatial binning and 7-dimensional histogram binning. Then, this $9 \times 7 = 63$-dimensional descriptor is named as UCHoG (Uncompressed Histogram of Gradients). This UCHoG is still high dimensional and remains inappropriate for mobile visual retrieval implementations. Compression schemes typically are applied on the final feature descriptor or feature vector. But, since UCHoG has a fixed-length normalized histogram binning for each spatial bin, it is feasible to apply the compression directly on these normalized histograms. Three compression encoding were employed: Huffman Tree Coding, Entropy Constrained Vector Quantization (ECVQ) and Type Quantization Chandrasekhar et al. [2010b] Reznik et al. [2010]. ECVQ method will not be described because it performs as well as Type Quantization, but results much more expensive (its costs being comparable to a k-means algorithm). In the following, we will consider that for each of the 9 spatial bins we obtain a normalized accumulated 7-bin histogram distribution as follows:

$$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}.$$

Normalization implies that $p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 = 1$.

CHoG's aim is to represent (or approximate) this 7D vector with as few bits as possible. This can be generalized whatever the number of spatial or histogram bins. In the description that follows, we consider that each spatial bin has a m-bin histogram of gradients.

A.1 Huffman Tree Coding

Huffman Tree Coding is a classic algorithm to encode data that uses weights or frequency of occurrence for each symbol (character) in the input and assign a variable-length bit string to each symbol. The characters to be encoded are arranged on a strict binary tree (each node has zero or two children) and an encoding for each symbol is found by following a binary tree from the route "root node" to the symbol in the "leaf node". Hence, The symbols with highest weights have shorter encoding lenght because these symbols are more close to the root node. The basic recursive operation to arrange those symbols on a binary tree is: combine the two least weight nodes into a tree which is assigned the sum of the two leaf node weights as the weight for this new node.

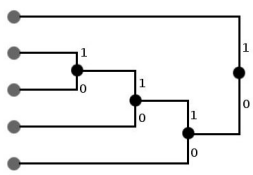
P	Huffman Tree	Depth	Code	Q
0.8271		1	1	0.5000
0.0011		4	0111	0.0625
0.0312		4	0110	0.0625
0.0625		3	010	0.1250
0.0781		2	00	0.2500

Figure 27 – Example of Huffman tree coding for 5 leaves ($m = 5$). P is the initial normalized vector and Q is the encoded vector.

In general for m leaves:

$$P = \{p_1, p_2, \cdots, p_m\}$$
$$Q = \{q_1, q_2, \cdots, q_m\}.$$

Each normalized histogram is considered as a probability distribution or weight distribution. Then, this distribution, P, is used to construct a Huffman tree as shown in Figure 27. The reconstructed distribution, Q, are obtained as $q_i = 2^{-b_i}$ where b_i is the depth of i-*th* symbol. Depth is defined as the number of bits used to encode this symbol or the distance from the route to the leaf node. The Huffman tree can be stored as follows:

- Case 1. Store the depth of each leaf node in the Huffman tree:
  Considering a distribution with m leaf nodes, the maximum depth in the Huffman tree constructed is $m - 1$. Hence, each symbol needs $\lceil log_2(m-1) \rceil$ bits to store the depth of each symbol. Then, to store the Huffman tree, we need $m \lceil log_2(m-1) \rceil$ bits. But, as the $\sum q_i = 1$, the last leaf node can be computed using the other $m-1$ leaves and does not need to be stored. Finally, the Huffman tree that represents a normalized histogram distribution can be stored with a fixed length code in $(m-1) \lceil log_2(m-1) \rceil$ bits.

- Case 2. (Optimization) Stores the *index* of the Huffman tree.
  Considering m elements in the probability distribution, the upper bound of the total number of Huffman codes can be estimated. The problem is stated as follows: "With m leaves, how many Huffman trees can be found?". There is not a closed form to solve this problem, but, an upper bound can be established using a combinatorial result of the number of non-isomorphic trees with m leaves known as Catalan number (C_{m-1} for m leaves). Unlike non-isomorphic tree problem, the order of elements is important

because the same elements in a different order denotes a different tree. Therefore, the upper bound of the number of Huffman trees $T(m)$ can be estimated as:

$$T(m) < m!\, C_{m-1}.$$

Where $C_n = \frac{1}{n+1}\binom{2n}{n}$ is the Catalan number. The index of Huffman tree representing a normalized histogram distribution $P = \{p_1, p_2, \cdots, p_m\}$ with a fixed-length encoding that requires *at most* $\lceil log_2 T(m) \rceil$ bits to encode.

Comparison:

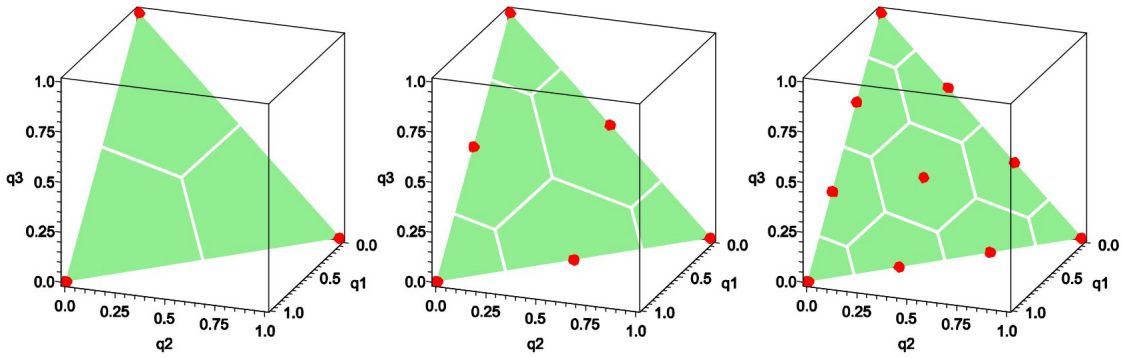
For the CHoG configuration with $m = 7$, the depth of each leaf node of a Huffman tree can be stored in $6\lceil log_2(6) \rceil = 6\lceil 2.585 \rceil = 6 \times 3 = 18$ bits. On the other hand, the number of Huffman trees for $m = 7$ is $T(m) = 4347$. Then, the index of Huffman tree can be stored in $\lceil log_2 4347 \rceil = \lceil 12.086 \rceil = 13$ bits. Therefore, storing the index of the Huffman tree is better to encode these normalized histograms distribution because the bit-rate is lower than storing depths of each leaf node. This particular conclusion is true for small values of m.

A.2 Type Quantization Coding

The basic idea of type coding is to quantize the normalized histogram distributions, P, using a set of reconstruction points (or codebook), that must have some regular structure. This codebook is obtained constructing a Lattice of distributions or Types Q using type coding. The lattice of distribution $Q(k_1, k_1, \cdots, k_m)$ has probabilities $q_i = \frac{k_i}{n}$, where $k_i \in \mathbb{Z}_+$ (non-negative integer), $\sum_i k_i = n$, and n is a parameter used to control the density of the codebook (how many points) over the space. The lattice is defined within a bounded subset of the R^m space, which is the unit $(m-1)$-simplex that contains all possible input probability distributions.

Then, for $m = 3$ ($P = \{p_1, p_2, p_3\}$) the 2-simplex unit is a triangle and contain all possible normalized histogram distribution as shown in Figure 28. Notice that as n increases, the number of types also increases.
For $n = 1$: $q_i = k_i$ and $k_1 + k_2 + k_3 = 1 \Rightarrow k_i = \{0, 1\}$. Then, $q_1 + q_2 + q_3 = 1$ and $q_i = \{0, 1\}$.
For $n = 2$: $q_i = \frac{k_i}{2}$ and $k_1 + k_2 + k_3 = 2 \Rightarrow k_i = \{0, 1, 2\}$. Then, $q_1 + q_2 + q_3 = 1$ and $q_i = \left\{0, \frac{1}{2}, 1\right\}$.

For $n = 3$: $q_i = \frac{k_i}{3}$ and $k_1 + k_2 + k_3 = 3 \Rightarrow k_i = \{0, 1, 2, 3\}$. Then, $q_1 + q_2 + q_3 = 1$ and $q_i = \left\{0, \frac{1}{3}, \frac{2}{3}, 1\right\}$.
In 3-Dimensions these 2-simplex surfaces has a normal vector $\left\{\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right\}$, because the plane equation for all is: $q_1 + q_2 + q_3 = 1$



$(n = 1)$ $(n = 2)$ $(n = 3)$

Figure 28 – Type lattices and their Voronoi partitions in 3-D. $m = 3$ and $n = 1, 2, 3$. (reproduced from Chandrasekhar *et al.* Chandrasekhar et al. [2012]).

The original normalized histogram distribution P is quantized into the closest type Q. To perform this quantization, we use the algorithm described below, that is similar to Conway and Sloane's quantizer for A_n lattice.
1. Compute values $(i = 1, 2, \cdots, m)$

$$k_i^{'} = \left\lfloor np_i + \tfrac{1}{2} \right\rfloor, n' = \textstyle\sum_i k_i^{'}.$$

2. If $n = n^{'}$ the nearest type is given by: $k_i = k_i^{'}$. Otherwise, compute errors:

$$\delta_i = k_i^{'} - npi,$$

and sort them such that

$$-\tfrac{1}{2} \leq \delta_{j1} \leq \delta_{j2} \leq \cdots \leq \delta_{jm} < \tfrac{1}{2}.$$

3. Let $\Delta = n^{'} - n$. If $\Delta > 0$ then decrement Δ values $k_i^{'}$ with largest errors

$$k_{ji} = \begin{cases} k_{ji}^{'} & j = 1, \cdots, m - \Delta - 1. \\ k_{ji}^{'} - 1 & i = m - \Delta, \cdots, m. \end{cases}$$

Otherwise, If $\Delta < 0$ then increment Δ values $k_i^{'}$ with smallest errors

$$k_{ji} = \begin{cases} k'_{ji} + 1 & i = 1, \cdots, |\Delta|. \\ k'_{ji} & i = |\Delta| + 1, \cdots, m. \end{cases}$$

4. Return quantities $k_1, \cdots, k_m$.

To store these quantized distribution we need to enumerate all possible type lattices in order to index them. The total number of types is a combinatorial problem: "how many m-dimensional vector exists that the sum of elements (nonnegative integer) is n?" and its result is given by a multiset coefficient. The total number of type lattices for m-dimensional distributions with parameter associated n, is a multiset coefficient and can be computed as an equivalent combination: $\left(\!\binom{m}{n}\!\right) = \binom{m+n-1}{m-1}$. Therefore, the normalized histogram distribution requires at most $\left\lceil log_2 \left(\!\binom{m}{n}\!\right) \right\rceil$ bits to be encoded in a type lattice. The closed solution to enumerate each type with an index $\xi(k_1, \cdots, k_m)$ is:

$$\xi(k_1, \cdots, k_m) = \sum_{j=1}^{n-2} \sum_{i=0}^{k_j - 1} \left(\!\binom{m-j}{n-i-\sum_{l=1}^{j-1}}\!\right) + k_{n-1}.$$

This direct enumeration allows to encode without storing any "codebook" or index of reconstruction points. For CHoG configuration with $m = 7$ the number of type lattices for $n = 7$ is $\left(\!\binom{7+7-1}{7-1}\!\right) = \left(\!\binom{13}{6}\!\right) = 1716$. Then, the index of type lattice can be stored in $\lceil log_2 1716 \rceil = \lceil 10.745 \rceil = 11$ bits.

A binary version[1] of CHoG descriptor is available, but the sources belong to Nokia (the mobile visual search project that was developed at Nokia Research Center — Palo Alto) and have not been released. The main author (then PhD. Vijay Chandrasekhar) is no longer reachable to comment on parameters and implementation details. Therefore we have faced a very laborious work of reimplementing the technique from scratch and we are still working on a complex parameter guesswork in order to match their results.

[1] `http://www.stanford.edu/~dmchen/documents/chog-release.zip`

APPENDIX B – CHoG Implementation

The implementation can be divided into detector and descriptor steps. In the original article the authors explain the description process of CHoG assuming as an input normalized patch regions of 64×64 pixels, but no further information about the detection process of these regions is provided. This detection step is very important because it has a direct influence on the performance of descriptor in retrieval tasks.

To implement the CHoG descriptor we use the OpenCV–Python interface and the Numpy package to perform operations on the image arrays. The decision to work with OpenCV–Python is due the good trade-off between convenience, portability and performance.

B.1 Interest Point Detection

The reference Linux binary implementation CHoG offers two methods for feature detection: DoG, and Fast Hessian. Since, those seem to correspond to the techniques used in SIFT and SURF, we have tried to establish such correspodence, but with less-than-perfect results. OpenCV implementation of SIFT and SURF depends on several parameters, which we are still to exhaust.

To evaluate the detectors we selected a collection of 20 real images of VallePics image dataset including outdoor scenes, human presence and aerial photographs, the results for 5 of these 20 images are presented in the Table 11. For SIFT, OpenCV's implementation with default paramenters detect almost $10\times$ more features than CHoG binary's DoG.

B.1.1 Parameter Estimation

The procedure used is to vary the parameters in the detector to get a similar number of interest points to the binary version. Each parameter is described in the OpenCV documentation[1] as follows:

- *nfeatures*: the number of best features to retain. The features are ranked by their scores.
- *nOctaveLayers*: the number of layers in each octave. 3 is the value used in D. Lowe paper Lowe [2004].

[1] OpenCV Documentation: `http://docs.opencv.org/modules/nonfree/doc/feature_detection.html`

Image	CHoG-DoG (binary version) number of keypoints	SIFT-OpenCV (default parameters) number of keypoints	SIFT-OpenCV (guessed parameters, trying to match CHoG) number of keypoints
qcam2F	5260	19713	5312
qcam3D	1488	19667	1885
qcam14	3542	16044	3721
qcel01	3126	13599	3364
qcel6F	3731	9648	3905

Table 11 – Number of interest points for 5 images from VallePics using DoG (binary) and SIFT OpenCV implementation (default parameters values and estimated parameter values).

– *contrastThreshold*: the contrast threshold used to filter out weak features in semi-uniform (low-contrast) regions. The larger the threshold, the less features are produced by the detector.
– *edgeThreshold*: the threshold used to filter out edge-like features. The larger the edgeThreshold, the less features are filtered out (more features are retained).
– *sigma*: the sigma of the Gaussian applied to the input image at the octave #0. If your image is captured with a weak camera with soft lenses, you might want to reduce the number.

The default values for these parameters are: SIFT(*nfeatures*= 0, *nOctaveLayers*= 3, *contrastThreshold*= 0.04, *edgeThreshold*= 10, *sigma*= 1.6). Where, *nfeatures*= 0 indicates that the number of features is not limited.

We have explored those parameters, testing all combinations of values as showed in Table 12. Thus, for each probe image, $5 \times 9 \times 10 \times 10 = 4500$ trials were run. We then selected the configuration $[0, 3, 0.11, 10, 16]$, which returns the closest number of interest points slightly above the binary version (Table 11).

Parameter	Type	Default	Values tested
nfeatures	int	0	0
nOctaveLayers	int	3	$[1, 2, 3, 4, 5]$
contrastThreshold	double	0.04	$[0.01, 0.02, 0.04, 0.06, 0.08, 0.09, 0.10, 0.11, 0.15]$
edgeThreshold	double	10	$[2, 4, 6, 8, 9, 10, 11, 12, 14, 16]$
sigma	double	1.6	$[0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2, 2.2, 2.4]$

Table 12 – Parameter description and its values used to perform the estimation process.

Typically, detection methods return the interest point location (x, y), the scale or size (σ) and the orientation (θ). From these values, we can establish a correspondence between some keypoints detected using DoG (binary) and SIFT OpenCV as follows:

- OpenCV SIFT: $x_1, y_1, \sigma_1, \theta_1$.
- Binary CHOG (DoG): $x_2, y_2, \sigma_2, \theta_2$.
  Where, $x_1 = x_2, y_1 = y_2, \sigma_1 = 2 \times \sigma_2$ and $\theta_2 = \theta_1 \times \pi \times 180^{-1}$.

That is, location units are transposed, OpenCV considers the diameter while the reference binary considers the radius as size, and while OpenCV measures orientation in degrees, the reference binary uses radians. Making the appropriate conversions we obtain very similar keypoints distribution for both detectors (using the parameter specified above for SIFT OpenCV) as shown in Figure 29.

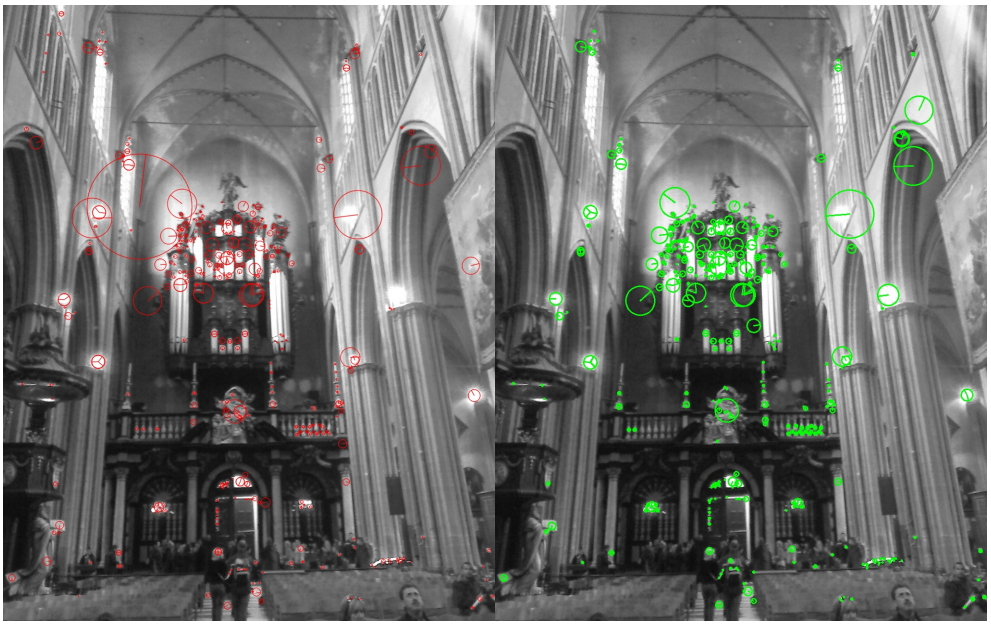


Figure 29 – Correspondence between keypoints obtained using SIFT OpenCV (left) and DoG of binary CHoG (right).

B.1.2 Patch size selection

We take a patch centralized on the detected point, with a radius of 3 times the DoG detector size (σ in SIFT OpenCV). This is justified by the Gaussian weighting, which uses σ as standard deviation: $3 \times \sigma$ from the mean retains almost all useful information. The patch is scaled and rotated such that it is normalized into 64×64 pixels, with its main orientation pointing upright. Those implementation choices were based on those of VLFeat[2] and Daisy[3].

B.2 Descriptor Implementation

The descriptor implementation follows the pipeline explained in the original article, as listed below:

- A set of patchs of 64×64 are used as input.
- The patch illumination is normalized in mean and variance.
- We apply to the patch a Gaussian smoothing, with $\sigma = 2.7$ pixels.
- Image gradients dx and dy are computed with a sobel convolutional filter.

Then, we use the configuration of orientation and spatial bins to compute a descriptor.

[2] VLFeat: `http://www.vlfeat.org/overview/sift.html`
[3] Optimized parameters for DAISY: `http://www.cs.ubc.ca/~mbrown/patchdata/tutorial.pdf`

B.2.1 Histogram Binning

As described in Section 2.2.1.6 the configuration of histogram bin is slightly more oriented over the y axis. Using the Figure 11 as reference, we can set the bin centers for VQ-7 as follows:

$$\begin{gathered}\text{HistBinCenters} \\ = (0, 0.1), (-0.07, 0.05), (0.07, 0.05), (0, 0), (-0.07, -0.05), (0.07, -0.05), (0, -0.1).\end{gathered}$$

The value of σ used for the Gaussian soft-assigment corresponds to the $d_{min}/3$ between histogram bin centers: $d_{min}/3 = 0.028675$.

B.2.2 Spatial Binning

There is no definitive reference to determine the spatial bin centers. The most comprehensive source for good configurations of spatial bin center, is the non-refeereed tutorial parameter optimization for DAISY but this tutorial mentions no configuration for DAISY-9 (9 centers). The most similar configuration corresponds to DAISY-13. Adapting DAISY-13 by removing the outermost centers, we have decided the following bin centers:

$$\begin{gathered}\text{SpatBinCenters} \\ = (-15, 15), (15, 15), (0, 8), (-8, 0), (0, 0), (8, 0), (0, -8), (-15, -15), (15, -15).\end{gathered}$$

The value of σ used for the Gaussian soft-assigment corresponds to the $d_{min}/3$ between spatial bin centers: $d_{min}/3 = 2.666667$.

B.2.3 Compression

Huffman code and type quantization were implemented. They are applied independently over each accumulated normalized histogram distribution (one for each spatial bin).

— Huffman Code. To enumerate all possible trees we wrote a program to determine the number of tree structures with n leaves, and then another piece of code to determine the number of labeled trees that can be constructed using n leaves. To index a distribution we implemented a program that construct a tree based on this distribution and compare to the enumerated trees and save the number or index of the correspondent tree as encoded value.

— Type Quantization. The algorithm described before (on the theory section of type quantization) was implemented using Python. Then, when a normalized histogram is

received the program use the closed solution described in Appendix A to determine the number or index of this normalized distribution.

We performed a retrieval experiment using the reference binary CHoG (with both DoG and Fast Hessian detectors) and also testing the initial (non-optimal parameters) versions of the implemented CHoG. The results of these retrieval procedures are shown in the Table 13, where an image was considered as correctly retrieved when the correct image is among the top 10 ranked (most voted). As explained before, the effort to obtain an optimal parameters for SURF was unsuccessful and is evidenced in the lower percentage of the correctly retrieved images compared to its counterpart CHoG (Fast Hessian).

All experiments described in this section were performed using images from VallePics[4] dataset and its related query images[5].

	CHoG Fast Hessian	CHoG DoG	CHoG (SURF-OpenCV)
% of images successfully retrieved	89	97	62

Table 13 – Results of perfomance in retrieval task.

The methodology of parameter evaluation and learning that was proposed for Daisy [Winder et al., 2009] is applied with slight modifications for CHoG. Once we have our technique working, we want to translate such methodology for CHoG and test different parameter configurations. Having a reference implementation in a mobile device is one of our aims, for both tasks of retrieval and classification.

[4] VallePics: `http://www.dca.fee.unicamp.br/~dovalle/VallePics/base.tar`
[5] Queries: `http://www.dca.fee.unicamp.br/~dovalle/VallePics/queries.tar.gz`