**Politecnico di Milano**

**Scuola di Ingegneria Industriale e dell'Informazione**
**Corso di Laurea Magistrale in Ingegneria Informatica**

# Visual Landmark Recognition Using Binary Features

Relatore: Prof. Marco Tagliasacchi

Tesi di Laurea di:
Andrés Felipe Pérez Murcia
Matricola 781856

a.a. 2013/2014

“For those who search
though don’t find nothing.
For those who advance
though get lost.
For those who live
though die.”

Mario Benedetti

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Abstract

The need of alternative or complementary localization methods in a mobile environment has motivated the study of image-based approaches. Relevant works on the area often miss the inherent limitations of mobile devices resulting in a clear need of more adequate techniques. In this order of ideas the main purpose of this work is to investigate the use of binary features in the context of large-scale mobile visual landmark recognition.

In content-based image retrieval a popular approach for images description is the Bag-of-Features model which using a codebook learned from a set of local features produces global image descriptors by quantizing local ones. For codebook construction the BoF model typically relies on some variant of the k-means clustering algorithm which is well defined only for real-valued data. In the basis of a thorough discussion about the different issues regarding clustering binary data streams, in this work are proposed and evaluated two approaches to build codebooks starting from binary features.

The results of the evaluation of the proposed approaches reveal firstly that codebooks built from binary descriptors using Hierarchical K-Majority are less representative than their real-valued counterpart. Secondly they evidence that the most influential factor governing retrieval performance is the utilized detector and descriptor. Some identified strategies to overcome these issues are: to increase codebook's representativity by means of a denser images description, to keep high the ratio of the number of descriptors to the number of code vectors, and to use pairs of feature detectors and descriptors invariant only to the transformation effects present in the target images.

# Sommario

L'esigenza di metodi di localizzazione alternativi o complementari in un ambiente mobile ha motivato lo studio degli approcci basati sulle immagini. Studi rilevanti nell'area spesso ignorano le limitazioni intrinseche dei dispositivi mobili, tali circostanze evidenziano la necessitá di tecniche piú adeguate. In questo ordine di idee lo scopo principale di questo lavoro é quello di investigare l'uso di caratteristiche binarie nel contesto del riconoscimento visivo di punti di interesse su larga scala in ambiente mobile.

Nel recupero di immagini basato sul contenuto un approccio popolare per la descrizione di immagini é il modello della borsa di parole il quale utilizza un libro codice appreso da un insieme di caratteristiche locali per produrre descrittori globali di immagini quantizzando quelli locali. Per la costruzione del libro codice il modello BoF si basa tipicamente su qualche variante dell'algoritmo di clustering k-means il quale é ben definito solo per dati a valori reali. In base ad una discussione approfondita sui diversi problemi riguardanti il clustering di dati binari, in questo lavoro vengono proposti e valutati due approcci per costruire libri codice a partire da caratteristiche binarie.

I risultati della valutazione degli approcci proposti rivelano in primo luogo che i libri codice costruiti da descrittori binari utilizzando "Hierarchical K-Majority" sono meno rappresentativi della loro controparte a valori reali. In secondo luogo i risultati evidenziano che il fattore piú influente sulla capacitá di recupero é la combinazione di rivelatore e descrittore utilizzata. Alcune strategie individuate per superare questi problemi sono: aumentare la rappresentativitá del libro codice mediante una descrizione densa delle immagini, mantenere elevato il rapporto tra il numero di descrittori al numero di vettori di codice, e utilizzare coppie di rivelatori e descrittori di caratteristiche che siano invarianti solo agli effetti di trasformazione presenti nelle immagini sotto descrizione.

# Introduction

In recent years with the proliferation of mobile devices with increasing processing power and built-in context sensors[1] has increased the number of location and context-based services demanding for real-time self-localization. Traditional approaches to determine location use Global Positioning System (GPS) receivers or cellular towers proximity in outdoor environments and WiFi-based alternatives for indoors. Nevertheless most interesting location-based services (LBS) applications such as points of interest search, active mobile advertising, and social applications are delivered in densely populated environments like downtown areas with urban canyons or indoors scenarios where GPS readings are unreliable or unavailable [30]. The lack of enough support using satellite navigation claims for alternative or complementary positioning mechanisms capable of delivering location information almost in real-time and scaling up to city level.

The problem gains importance because its emerging commercial aperture with products such as Google Goggles, a mobile application for image recognition providing visual product search and landmark identification, and Google Glasses, a wearable computer with an optical head-mounted display, as well as the interest from the robotics community in providing robots the ability of understanding its surrounding in an unknown environment, task known as simultaneous localization and mapping (SLAM).

Recent work in content-based image retrieval (CBIR) and object-based image retrieval [31, 17, 20, 21], where the original image retrieval problem is re-formulated as a text retrieval one, has paved the way towards the location recognition problem [28, 30, 5, 2] on which an image or a continuous sequence of images is used as a visual fingerprint of the environment and matched to an existing georeferenced database in order to estimate user's location. The outcome might include scene location and viewing direction estimates. While very innovative and promising, location recognition using visual clues still relies on vague prior knowledge on location coming from sources such as wireless networks (WiFi) or cell towers (Cell-IDs).

Even though the use of contextual information benefits the recognition process [6] here I concentrate on the content analysis part of the problem. From now on we deal with an image matching task which can be approached either in a 2D-to-2D or 2D-to-3D manner; in this work and despite the growing trend towards using 3D information [2, 27, 9] I focus on finding 2D-to-2D correspondences.

[1]Inertial Measurement Units (IMU), Global Positioning System (GPS) receivers, and digital cameras.

A key challenge for visual localization is the rapid and accurate search of similar images to a query image by efficiently describing and retrieving objects exhibiting a number of variations. It's complexity lies on the large variation of scene visual appearance due to changes in lighting conditions, scale and viewpoint, dynamism of depicted scene objects, as well as occlusions and overlaps resulting from mapping 3D world objects onto a 2D space. Other particularities worth mentioning that introduce substantial changes to the scene visual appearance are visual clutter (e.g. cars, pedestrians, seasonal stuff, and ads), buildings casting shadows each other, reflections and glare in windows, and severe perspective distortion with extreme angles.

The inherent limitations of mobile devices introduce further challenges to the location recognition problem:

- **Fundamental differences between the query image and database images:** because of the properties of mobile device cameras, query images are typically affected by motion blur and provide a limited field of view, making difficult the task of matching them against database images.
- **Limited bandwidth (network performance)** claiming for low complexity approaches with efficient communication.
- **Low retrieval times:** due to rapidly changing field of view caused by user's motion and constantly changing user's attention.
- **Low processing power and battery capacity:** current mobile devices still lack sufficient on-board processing power to perform computationally intensive operations such as content analysis.
- **Limited storage capacity:** coupled with low processing power, limited storage inhibits the use of very high dimensional feature descriptors.
- **Real-time requirements:** user's real time requirements restrain the types of content and context analysis techniques that real systems can employ.

The use of the Scale Invariant Feature Transform (SIFT) for images description as part of the CBIR pipeline has proven to be very effective and discriminative for sparse feature-based matching but not necessarily optimal for image classification [18]. This supports the use of another type of descriptors more adequate to the mobile context where computing efficiency and descriptor size are main concerns.

In this work I address the problem of large-scale landmark recognition, a variation of the location recognition problem. Here I intend to investigate the use of binary features to alleviate the communication delay bottleneck described in [30] while addressing the properties of low processing power and limited bandwidth of mobile devices. The objectives of this study are twofold, first to determine how representative can be vocabularies built from binary features in the task of landmark recognition and second to identify which factors govern retrieval performance in a system using such vocabularies.

To this purpose I have implemented: 1) a state-of-art pipeline using upright SIFT descriptors paired with a DoG detector, a vocabulary tree as BoF approach and random sample consensus (RANSAC) for estimating a 2D projective model, 2) a fully parametrized tailored pipeline for the case of binary features, and 3) a novel proposal based on the ideas of [20]. In order to compare them I've implemented a performance evaluation framework based on precision and recall, two popular measures widely accepted by the information retrieval community.

During the first part of this work I introduce the theoretical foundations necessary to understand the rest of this work. In the first chapter I clearly define the landmark and location recognition problems and review some methods and techniques associated to them. In order to locate the current work with respect to the state-of-the-art, in the second chapter I assess some previous work on the area looking at the different aspects of the problem at hand. In chapter number three I review some algorithms to describe images using binary features, highlighting their invariance capabilities to common changes in scene objects.

In the second part I present my contributions starting in chapter four with a thorough review of the challenges of using the new class of local features to address the landmark recognition problem in the frame of the BoF model, followed by a detailed explanation of the two proposed pipelines to the case of binary features. Finally in the last chapter I present the performance evaluation framework used to assess the proposed pipelines, emphasizing the evaluation methodology and the considerations taken to render fair the comparison process, followed by the discussion of the results of the performed experiments prior description of the datasets used.

# 1 Associated methods and techniques

Large-scale landmark recognition is essentially a CBIR task with many challenges caused by the large variation of scene visual appearance, the nature of the images in the database and in the mobile scenario the inherent limitations of mobile devices. It is closely related to many computer vision problems and tasks ranging from image description to object recognition. In this chapter I introduce the CBIR foundations, present the standard retrieval scheme and formally define the landmark and location problems.

## 1.1 Feature-based image description

Image description lies at the core of any CBIR pipeline, it consist on extracting relevant information (features) from a digital image and encode it in a vector representation. In general a feature can be anything which has a high discriminant power, for example highly textured patches with strong contrast such as edges or corners.

The images characterization process becomes more challenging because landmark images possess unique characteristics which should be considered. For example landmarks such as buildings have many repetitive structures such as doors and windows which complicate finding discriminative features. Likewise landmark images captured using different camera models might vary in illumination, viewpoint, scale, and rotation. Other effects they might include are blurring, occlusion, background clutter and shadowing.

Broadly speaking visual features for image characterization can be classified as *global* or *local*. *Global* features represent the entire image and thus all pixels in the image are considered. *Local* features identify certain interest points or regions in the image and only utilize image properties characterizing those regions or local neighborhoods around those interest points for images description, the rest of the pixels are ignored. Because of their relatively higher robustness towards many changes in visual appearance like the ones above mentioned, *local* features are preferred over *global* features in the context of landmark and location recognition.

The *local* features extraction pipeline consists typically of two stages: *detection* of salient points or regions (keypoints) and *extraction* of descriptors to characterize the appearance of the detected local regions. Depending on the strategy for salient points or regions identification, *local* features can be classified as based on keypoints, segmentation, or either random or uniform sampling. Among these the keypoint-based approach has gained popularity over the other ones due to their capability of finding more discriminative while repeatable structures.

In order to faithful capture the essence of the underlying input images and encode their interesting structure using the local features keypoint-based approach, two important criteria must fulfilled: 1) the feature extraction process should be repeatable and precise, so that the same features are extracted on two images showing the same object, and 2) the extracted features should be distinctive, so that different image structures can be told apart from each other. Additionally to address partial occlusion it is required a sufficient number of features covering the object to be described. In the next two sections I briefly review some popular feature detection and description techniques.

### 1.1.1 Feature detection

The first stage of the *local* feature extraction pipeline consist in finding a set of distinctive keypoints that can be reliably localized under varying imaging conditions and in the presence of noise. Common considered changes in visual appearance include changes in illumination, in plane rotation, noise, scale, and perspective distortion.

#### Translation Invariant Features

The most basic type of detectors are those capable of finding corner keypoints invariant to isometric transformations. Examples of such detectors are the *Hessian detector* and the *Harris detector* which compute some saliency measure based on the second moment matrix in order to find keypoints exhibiting changes in orthogonal directions. Both of this detectors are remarkably robust to image plane rotations, illumination changes, and noise, however the returned locations by them are only repeatable up to a relatively small scale changes because they rely on Gaussian derivatives compute at a certain fixed base scale.

#### Scale Invariant Features

For scale invariant feature detection it is necessary to detect structures that can be reliably extracted under scale changes. To do so keypoint detection is performed in a scale-space which is constructed by smoothing the high resolution image with derivatives of Gaussian kernels of increasing size. Automatic scale selection is performed by selecting local maximum in the scale-space.

The most basic detector of this type is the *Laplacian-of-Gaussian (LoG)* detector, a multi-scale blob detector with automatic scale selection which detects local maxima/minima in the scale-normalized Laplacian space. Closely related there is the *Difference-of-Gaussians (DoG) detector* which approximates the scale-space Laplacian by a difference of Gaussians. Other popular detectors are the *Harris-Laplace* corner detector, a scale-invariant version of the Harris detector which typically returns fewer keypoints than the *LoG* or *DoG* detectors, and the *Hessian-Laplace* blob detector, a scale-invariant version of the *Hessian detector* which typically returns more interest regions than *Harris-Laplace* at a slightly lower repeatability.

Finally there is the *Speeded Up Robust Features (SURF) detector* which uses the Hessian matrix like the *Hessian-Laplace* detector, but uses box filters to approximate the second-order Gaussian derivatives, obtaining a significantly speed up in computation and produces similar results as *DoG* but several times faster.

### Affine invariant detectors

For many practical problems it becomes important to find features that can be reliably extracted under large viewpoint changes. This is the case of landmark images which typically exhibit high perspective distortion due to the narrow angles at which they are shot.

An affine invariant detector is a generalization of the scale-invariant ones to handle non-uniform scaling and skew changes. These detectors extract patches from images that are covariant with affine transformations. The shape of these regions is not fixed but automatically adapts, based on the underlying image intensities, so that they are the projection of the same 3D surface patch. Popular detectors of this type are:

- Harris-Affine detector.
- Hessian-Affine detector.
- Maximally Stable Extremal Regions (MSER) detector.
- Edge-based region (EBR) detector.

Typically affine detectors extract elliptical-shaped patches, others such as EBR and MSER extract parallelogram- and arbitrary-shaped patches correspondingly, these regions are approximated by an ellipse taking care of preserving the first and second moments of the originally detected region.

To achieve rotation invariance the content of the detected regions is normalized, typically by finding the region's dominant orientation and then rotating the region content according to this angle in order to bring the region into a canonical orientation.

### 1.1.2 Feature description

Also known as descriptors extraction, it is the second stage of the *local* feature extraction pipeline. It consist in the encoding into a high-dimensional vector of certain properties of the image in the local neighborhoods centered at the detected keypoints.

The most popular choice is the SIFT descriptor which is typically paired with a variant of the DoG detector. It measures gradient distribution in the detected regions as a histogram. It is computed as a set of orientation histograms on $4\times4$ pixel neighborhoods in the gradient image. The contribution of each pixel to the location and orientation bins is weighted by the gradient magnitude. The quantization of gradient locations and orientations makes the descriptor robust to small geometric distortions and small errors in the region detection. Each orientation histogram contains 8 bins. This leads to a feature vector with dimension $4 \times 4 \times 8 = 128$. To enhance invariance to changes in illumination, the resulting vector is normalized.

Another popular choice is the SURF descriptor which combined with SURF detector was designed as an efficient alter- native to SIFT. It pursues a similar spatial binning strategy than SIFT, dividing the feature region into a $4 \times 4$ grid. However, instead of building up a gradient orientation histogram for each bin, SURF only computes a set of summary statistics resulting in a 64-dimensional descriptor, or a slightly extended set resulting in a 128-dimensional descriptor version.

### 1.1.3 Descriptors matching

A simple approach for pairwise image matching consist in linearly comparing their descriptors and counting how many of them match each other. Descriptors can be compared by thresholding the distance between them which can be computed using some distance measure such as Euclidean, Manhattan or Cosine distance. An alternative criteria to evaluate the quality of a matching candidate was proposed in the original SIFT matching scheme were it was introduced the notion of ratio threshold, a measure where a pair of features is considered a match if the distance ratio between the closest and second closest matches is below some threshold $\Gamma$:

$$\frac{d\left(f, f_{1^{st}}\right)}{d\left(f, f_{2nd}\right)} < \Gamma$$

where $f$ is the descriptor to be matched and $f_{1^{st}}$ and $f_{2^{nd}}$ are the nearest and the second nearest descriptors from the model database, with $d\left(\ldots\right)$ denoting the Euclidean distance between two descriptors. An empirically determined threshold value of $\Gamma = 0.8$ is typically used.

According to [3] the SIFT ratio threshold is effective for general recognition because correct discriminative keypoints often have the closest neighbor significantly closer than the closest incorrect match, however in the context of landmark recognition it doesn't work because landmark images have many repetitive structures, thus the original formulation ends up rejecting many genuine matches. An alternative formulation considers two key-points as matched when the cosine of the angle between the descriptors f and g is above some threshold $\Gamma_c$:

$$cos\left(f, g\right) = \frac{f^T g}{\|f\|_2 \cdot \|g\|_2} > \Gamma_c$$

An empirically determined value of $\Gamma_c = 0.97$ based on the study of the ROC curve associated with the threshold is typically used. In the case of binary features the SIFT ratio threshold doesn't apply and instead it is used a distance threshold:

$$d\left(f, f_{1^{st}}\right) < \tau$$

## 1.2 Image retrieval

A popular approach to the image retrieval problem is the bag-of-features (BoF) model proposed in [31]. The process starts by extracting local features from a collection of database images, then they are quantized into visual words, and finally are applied text search methods to retrieve similar images to a given query image. An optional re-ranking step based on geometric constraints is performed on the top retrieved results.

This scheme is generic both to the landmark and location recognition pipelines described in sec. 1.4 and sec. 1.5. Here it is addressed the generality while the details of its specific use either on the landmark or location recognition problem are addressed in their corresponding sections.

**Feature extraction**

It all starts with the extraction of local features from the images database which for the case of landmark and location recognition are preferred over the global ones due its robustness towards occlusion and background clutter, and invariance to some characteristics present on the images such as scale, rotation, lighting variation and perspective distortion. Many techniques can be found in literature (see sec. 1.1 for further details) but the use of SIFT is still considered the state-of-art thought there are more appealing options for some applications. To our problem the main criteria for choosing a certain feature descriptor are low computational complexity, low storage requirements, and invariance against visual appearance changes disregarding rotation since all images are assumed to be upright.

**Vocabulary building**

Also called quantization step, is the most expensive in the retrieval scheme as it involves the clustering of all the feature descriptors corresponding to the images database. The resulting cluster centers are the words of the vocabulary and altogether are called the bag-of-words (BoW) or more correctly bag-of-features (BoF). Refer to sec. 1.3 for further details about BoW model in computer vision.

**Database building**

Following the BoF model each database image is transformed into BoF vector representation by quantizing its feature descriptors to visual words. The results are indexed to speed up the classification step. To address that not all of the words are equally relevant it is used a weighting scheme, typically term frequency-inverse document frequency. See sec. 1.3 for further details about indexing and word weighting.

**BoF retrieval**

Given a query image it is transformed into it's BoF vector representation by quantizing into visual words the extracted visual features, and applying the corresponding words weights. In order to retrieve the most visually similar database images with respect to the query image, it's BoF vector representation is linearly compared against all the database BoF vectors and a similarity score is assigned to each of them. The database images are then sorted in descending order according to this score.

**Geometric verification**

Since BoF model ignores visual features layout, an optional geometric verification step is applied database images is applying geometric verification over the top ranked images, such that not only the most visually similar but also the most geometrically similar images are placed on the top.

The process uses RANSAC algorithm to compute a geometric transformation in the projective space between the keypoints of the query image and those of each candidate image in the short list of top ranked images, resulting in the classification of the candidate keypoints as inliers or outliers complying with the estimated homography. The short list of top candidates is then re-order based on the number of inliers.

**Algorithm 1.1** Automatic estimation of an homography between two images using RANSAC

```
(i)     Computation of interest points.
(ii)    Computation of putative correspondences based on a proximity and similarity criterion.
(iii)   Estimation of an homography using RANSAC where the number of samples is determined adaptively.
(iv)    Iterative re-estimation of the homography by minimizing a cost function and aiming at maximizing the number of inliers.
```

The basic algorithm for robust model estimation explained in [10] and here shown in Algorithm 1.1 is the same independent on the estimated geometric transformation.

The applied transformation might be an affine one as proposed in [20], which is a non-singular linear transformation followed by a translation

$$H_A = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$$

or an homography which is a general non-singular linear transformation of homogeneous coordinates

$$H_P = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & \nu \end{bmatrix} = \begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$$

capable of describing changes in perspective, something that occurs in practice for outdoor urban environment photos and cannot be captured by an affine transformation. Accounting for the degrees of freedom, which talks in the name of the query time, estimating an homography is more expensive than estimating an affinity having the former one 9 in comparison to 6 of the affinity.

## 1.3 Bag-of-Features model

It was originally introduced in [31] where the aim was detecting objects in a video sequence. To that end it reformulates the CBIR problem as a text retrieval one where the documents are the images in the database and the analog to a word in text documents is a visual word, a set of neighboring pixels sharing a texturing pattern.

The first step is to learn a *visual vocabulary* or *codebook* in an unsupervised manner from a training set of the same domain as the images of the database. It is typically addressed using k-means clustering, or some variant of it due to some inherent challenges because of the quantity and dimensionality of the data[1], namely:

1. The trade-off between distinctiveness (requiring small quantization cells) and repeatability (requiring large quantization cells).
2. The data sparsity condition due to the curse of dimensionality which complicates the k-nearest neighbors classification task [22] rendering clustering very inefficient.

Each obtained cluster center becomes a *visual word* (*codevector*) which altogether form the *visual vocabulary* (*codebook*). Provided a sufficiently representative training set it might be obtained an universal *visual vocabulary*.

[1]See chapter 2 to read about some approaches proposed in literature to handle this issues.

The second step is to use a vector quantizer to map or quantize each feature vector of an image to the nearest *codevector* in the *codebook* and store the frequency of occurrence of each visual word in a *visual words frequency histogram*, the so-called *BoF vector*. Each of this frequencies is called the *term frequency*.

In text retrieval it is usual to apply a term weighting scheme in order to increase or decrease the importance of some terms. In [31] three different weighting schemes are considered:

- **none:** use the raw histogram, i.e. only the term frequency counts.
- **binary:** binarize the histogram, i.e. register only whether the image has the visual word or not.
- **term frequency-inverse document frequency (tf-idf):** weight the counts in such a way to decrease the influence of more common words and increase the influence of the more distinctive ones. It is the product of two terms, the term frequency $\frac{n_{id}}{n_d}$ and the inverse document frequency $log\frac{N}{n_i}$ hence

  $$w_i = \frac{n_{id}}{n_d} log\frac{N}{n_i}$$

  where $i = \{1, \ldots, k\}$ is the index over the words of the vocabulary, $w_i$ is the weight of word $i$, $n_{id}$ is the number of occurrences of word $i$ in document $d$, $n_d$ is the total number of words in the document $d$, $n_i$ is the number of occurrences of word $i$ in the whole database and $N$ is the number of documents in the whole database.

Once the visual vocabulary has been defined, the database BoF vectors are ready and the word's weights have been computed, we are ready to proceed with BoF retrieval. The task is then: given a query image retrieve the most visually similar ones from an images database. This is achieved by assigning a similarity score to each database image and the sorting them in descending order according to its score. Under the BoF model the similarity between a query and database image is computed by taking the distance between their normalized BoF vector representation $q$ and $d$

$$q = (q_1, \ldots, q_k)^T = (w_1 n_1, \ldots, w_k n_k)^T$$

$$d = (d_1, \ldots, d_k)^T = (w_1 m_1, \ldots, w_k m_k)^T$$

$$d_? \left( \frac{q}{\|q\|_p}, \frac{d}{\|d\|_p} \right)$$

where $n_i$ and $m_i$ are the term frequency counts of $q$ and $d$ correspondingly, and $w_i$ the word weights which in case of *tf-idf* weighting are only the inverse document frequency part and in case of *none* it is 1 for all. In case of *binary* weighting, $q_i$ and

$d_i$ are either 0 or 1 as stated before. To achieve a score between 0 and 1 the BoF vectors are p-normalized typically using L1 or L2-norm.

As for scoring three possible schemes are considered in literature:

- **L1:** use the sum of absolute differences $d_{L1}(h, g) = \sum_i |h_i - g_i|$
- **L2:** use the sum of squared differences $d_{L2}(h, g) = \sum_i (h_i - g_i)^2$
- **Cosine:** use the dot product $d_{cos}(h, g) = 2 - \sum_i (h_i \cdot g_i)$ which is equivalent to the L2 distance for L2-normalized histograms.

To speed-up the scoring process against, the database BoF vectors are stored in an inverted index, a map structure supported by multiple inverted files, one per visual word. Its entries reference the images containing the visual word and times that word appears on the image.

Since the BoF vectors are typically sparse, specially when using large vocabularies (hundreds of thousands of words or above) e.g. 1M visual words vocabulary with an average of 5000 features per image, using an inverted index makes the retrieval very fast.

Despite the already mentioned challenges in vocabulary building the use of visual vocabularies as a method for images description carries many other issues:

1. **Visual queries contain many more words than text queries:** text queries typically consist of a few words chosen by the user whereas words in a visual queries are not under control of the user who simply submits an image.
2. **Visual words are noisier than words in text documents:** due to the coarseness of quantization it is common to obtain words which represent the same visual structure but under some undetected slight changes in appearance.
3. **Visual words in an image query encode vastly more spatial structure than a text query:** text queries do not constraint the position in the documents where the words appear while visual words define not only a visual structure but also an spatial configuration of the words in the image.

## 1.4 Landmark recognition

**Visual landmark recognition**, or simply **landmark recognition** is a binary classification problem, where the fundamental task is to determine whether a query image is relevant to a landmark. Fig. 1.1 shows the image retrieval scheme of sec. 1.2 applied to this problem. The process consists of two phases: *training* and *classification*, the first one corresponds to the vocabulary and database building steps whose output is the visual vocabulary and the indexed database including the words weights, the second one corresponds to the BoF retrieval step.

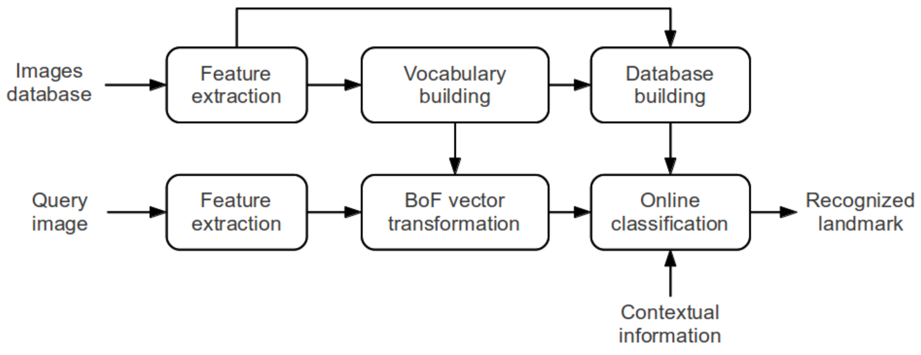


**Figure 1.1:** Pipeline of a basic landmark recognition system

Some remarkable particularities of a landmark recognition pipeline that distinguish it from a classical image retrieval system are:

- **It might use contextual information.** Incorporating vague prior knowledge such as rough GPS reading or direction information obtained from the digital compass into the retrieval process could significantly narrow down the search space.
- **It leverages the nature of the images.** using landmark images it can be assumed them to be upright, removing the requirement of rotation invariance, however buildings tend to have few discriminative visual features and many repetitive structures, and although this allows to use models based on projective geometry their 3D geometries are not easily captured by simple affine or projective transformations.
- **It involves an image classification task.** The BoF model is nothing more than a technique to generate a global image description out of local visual features, in order to tell whether an images depicts a particular landmark or not it is necessary to use some sort of classifier. Typically it is employed a k-NN classifier together with some efficient retrieval scheme, e.g. an inverted index supported by a set of inverted files. It finds the k nearest database images to the query image and assigns it to the class label that occurs maximum number of times in that neighborhood. This however could be computationally expensive because involves a linear comparison of the query BoF vector to all the database images BoF vectors. Alternatively it can be used a multi-class Support Vector Machine which uses a collection of binary classifiers[3].

## 1.5 Location recognition

**Visual location recognition**, or simply **location recognition** poses the task of estimating depicted scene location by matching a single or a sequence of images, used as a query, against a large database of georeferenced images. On the other hand **mobile location recognition** aims at estimating not only depicted scene location but also user's location and possibly viewing direction (that is the user's pose) given a single or sequence of images captured with a mobile device. Figure Fig. 1.2 shows the image retrieval scheme from sec. 1.2 applied to the mobile location recognition problem. As can be seen it involves a few more modules than the pipeline for a landmark recognition system, as they are: feature compression, constrained BoF retrieval (previously performed inside the online classification module), spatial re-ranking and pose estimation.

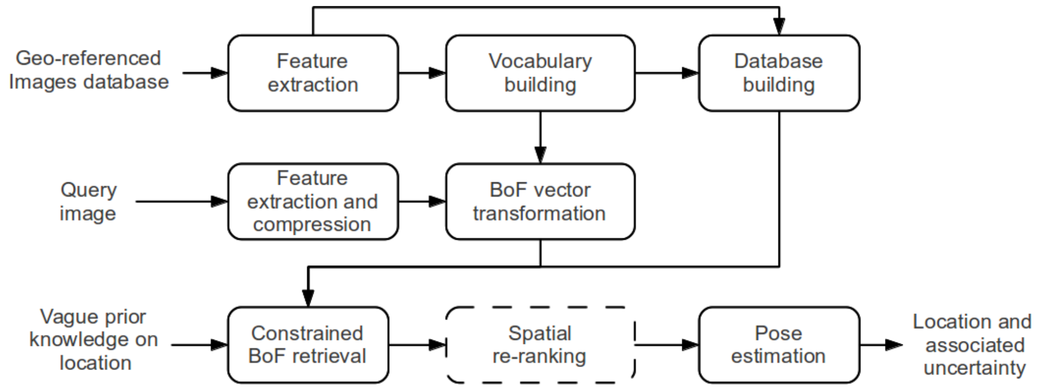

Geo-referenced Images database
Feature extraction
Vocabulary building
Database building
Query image
Feature extraction and compression
BoF vector transformation
Vague prior knowledge on location
Constrained BoF retrieval
Spatial re-ranking
Pose estimation
Location and associated uncertainty

**Figure 1.2:** Pipeline of a basic mobile location recognition system.

Whereas in a landmarks recognition system the output is the estimated label of the depicted landmark, in a location recognition system the output is the estimated user's location and possibly orientation as well as scene's location. Additionally to render useful such estimates, it is required to provide an uncertainty measure over them. The pose estimation task requires additional data to use analytic or geometric methods to determine pose, however in the image retrieval scenario usually the camera intrinsics and object geometry are unknown. Alternatively pose estimation is possible via image registration.

Popular benchmark datasets for location recognition consist in street-level panoramic data collected using surveying vehicles, such imagery constitutes a precisely calibrated and uniform city cover. However when using street view panoramic images as reference data we face a problem of wide baselines and close buildings matching becomes a difficult task.

A noteworthy difference in location recognition with respect to landmarks recognition is that relevance is defined by a given radius around the query location. This is because the interest is on evaluating the capability of the system to retrieve visually similar images located near the true position of the query image.

# 2 Related Work

This chapter surveys previous work in landmark and location recognition. In literature can be found many improvements over the standard retrieval scheme, some issues on which these works concentrate on are: increasing the number of visual words, improving retrieval performance, decreasing storage space and computational requirements, and lowering query time. While most of them build on top of the well known SIFT feature extraction pipeline, here I present as well some approaches addressing the case of binary-valued descriptors.

## 2.1 Vocabulary building

A key finding of [31] where BoF was firstly presented, is that using more visual words improves retrieval performance. In the next sub-sections are reviewed two popular and fundamentally different approaches to increase the number words.

### 2.1.1 Hierarchical K-Means

In [17] the authors present a new technique for object-recognition using a hierarchical structure built on top of iterative clustering and a tf-idf scoring scheme. This enables the use of large and more discriminatory vocabularies, thus increasing distinctiveness and reducing the query time.

The vocabulary is a $k$-ary tree structure built in an unsupervised manner by hierarchical k-means (HKM) clustering. An initial k-means run is done over the training data, defining $k$ cluster centers represented by its mean which at fine quantization describe a texturing pattern that sufficiently represents the descriptors associated to it. The process is recursively applied to each of the resulting clusters, recursively defining quantization cells by splitting each quantization cell into $k$ new parts. This procedure is performed up to a maximum number of $L$ levels. In practice other termination conditions may raise, e.g. having less data than clusters, and thus there is no guarantee of achieving the full theoretical number of leaf nodes $k^L$. Fig. 2.1 illustrates an example of a vocabulary built in this manner. Under this scheme the words are the nodes of the tree and the quantization of a feature vector consist in propagating it down the tree choosing each time the closest cluster center.

As in the classic BoF approach here it is also used an inverted index together with some weighting scheme to address efficient retrieval. In the case of a vocabulary tree,

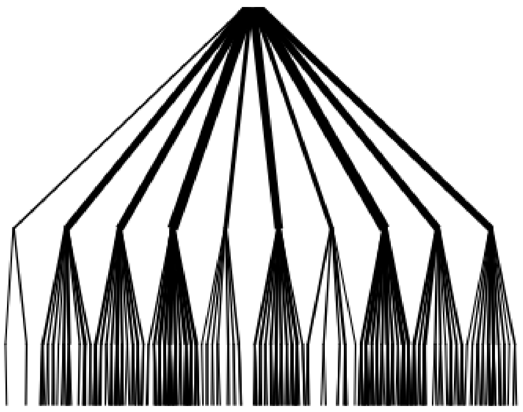


**Figure 2.1:** Illustration of a vocabulary tree with parameters $k = 10$ and $L = 2$, notice how $k$ defines the branch factor of the tree, i.e. the number of children of each node.

the leaf nodes have explicit inverted files and the inner nodes have virtual inverted files result of concatenating the inverted files of the leaf nodes as shown in Fig. 2.2. While for weighting, it is used a hierarchical scheme boosted by the use of inverted files where only leaf nodes are considered since they are much more powerful than the inner nodes according to the authors.

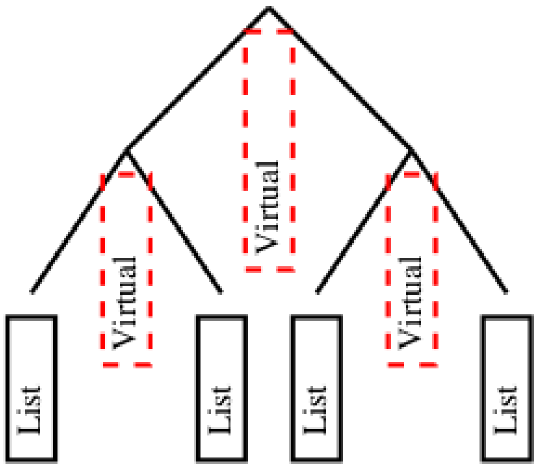


**Figure 2.2:** Inverted index in a vocabulary tree

To determine the relevance of a database image to a query image it is measured how similar the paths down the vocabulary tree are for the descriptors from the database image and the query image. This information is encoded inside the inverted files.

To efficiently compute the similarity score between a database and a query image it is taken the distance between their BoF vectors. The authors introduce an efficient way to compute the similarity score in the case the distance measure is the $L_p - norm$:

$$s\left(q,d\right)=\left\|\frac{q}{\|q\|}-\frac{d}{\|d\|}\right\|$$

For each non-zero query dimension $q_i = 0$, the inverted file can be used to traverse the corresponding non-zero database entries $d_i \neq 0$ and accumulate to the sum. Since the inverted files store only database images term frequency counts, all of them are non-zero hence many needless comparisons are skipped.

$$\left\|\hat{q}-\hat{d}\right\|_p^p=\sum_i\left|\hat{q}_i-\hat{d}_i\right|^p=2+\sum_{i|\hat{q}_i\neq0,\hat{d}_i\neq0}\left(\left|\hat{q}_i-\hat{d}_i\right|^p-\left|\hat{q}_i\right|^p-\left|\hat{d}_i\right|^p\right)$$

For the case of $L_2$-norm:

$$\left\|\hat{q}-\hat{d}\right\|_2^2=2-2\sum_{i|\hat{q}_i\neq0,\hat{d}_i\neq0}\hat{q}_i\hat{d}_i$$

Regarding vocabulary size, the authors confirm that larger ones (large number of leaf nodes) benefit retrieval quality. The idea behind this is that when vocabulary grows large the variability and noise in the feature vectors frequently move them between different quantization cells. The trade-off between distinctiveness and repeatability mentioned in previous chapter applies here as well but the risk of overdoing the size of the vocabulary is lessened by the use of the hierarchical scoring.

### 2.1.2 Approximate K-Means

Aiming at minimizing the overall distortion within the clusters defining the visual vocabulary and tackling down quantization effects in [20] it is proposed approximate k-means (AKM) clustering to reduce the computational complexity of learning a flat vocabulary.

To approximate nearest neighbor search for fast descriptor matching it is proposed the use of a forest of 8 randomized k-d trees built over the cluster centers at the beginning of each iteration[1]. In the randomized version, the splitting dimension and the splitting value are randomly, the first one among the set of dimensions with highest variance and the second one a point close to the median. Combining this trees creates an overlapping partition of the feature space and helps mitigating quantization effects by trying to find the correct quantization cell for features falling close to a partition boundary.

The robustness of the approach against quantization errors renders it particularly beneficial in high-dimensional spaces where due to the curse of dimensionality it is more likely that points lie close to a boundary. At quantization time a feature is

[1]It is done in this way because centroids change at each iteration and if they didn't its because convergence was achieved

assigned to the approximately closest cluster performing a backtracking process over all trees aided by a single priority queue. It is worth mentioning that this method has the same time and memory complexity than the hierarchical k-means approach.

Though recognition performance using a vocabulary tree with standard parameters (1M million leaf nodes, 6 levels of depth, and branch factor 10) has proven to be inferior to other approaches such as Approximate K-Means or Greedy N-best Paths, it has the lowest query time.

## 2.2 Improving retrieval performance

It was already shown in [20] that precision can be improved by running a geometric verification step over the short list of top retrieval results. However improving recall requires more sophisticated techniques, here I review some strategies to do so.

### 2.2.1 Greedy N-Best Paths Search

In [28] it is approached the quantization problem in vocabulary trees. Considering the case where the database consist of a fixed set of images, the authors propose a procedure to optimally build it using only most informative features in order to maximize performance of queries on the database. They develop as well an efficient greedy strategy for traversing several branches at the same time, albeit at the cost of increased query time.

The typical nearest neighbor search procedure in metric trees compares the query against the $k$ nodes of the tree at each of the $L$ levels for a total of $kL$ comparisons. In a different way Greedy N-best Paths (GNP) algorithm follows multiple branches at each level rather than just the branch whose parent is closest to the query. This generalization is described in Algorithm 2.1.

**Algorithm 2.1** Greedy N-Best Paths

```
Input: query feature $q$, level $l = 1$, number of paths followed $N$
Output: the cell where $q$ is quantized
Compute distance from $q$ to all $k$ children of root node
while($l < L$){
    $l = l + 1$
    candidates $\leftarrow$ children of closest $N$ nodes at level $l - 1$
    compute distance from $q$ to all $kN$ nodes in candidates
}
return candidate with the smallest distance to $q$
```

As can be seen in Fig. 2.3, the algorithm performs $k$ comparisons at the root level and $kN$ comparisons at each of the remaining $L - 1$ levels for a total of $k + kN(L - 1)$

comparisons. The case where $N = 1$ corresponds to the the traditional search procedure. Varying the new parameter $N$ gives the flexibility to specify the amount of computation per search.

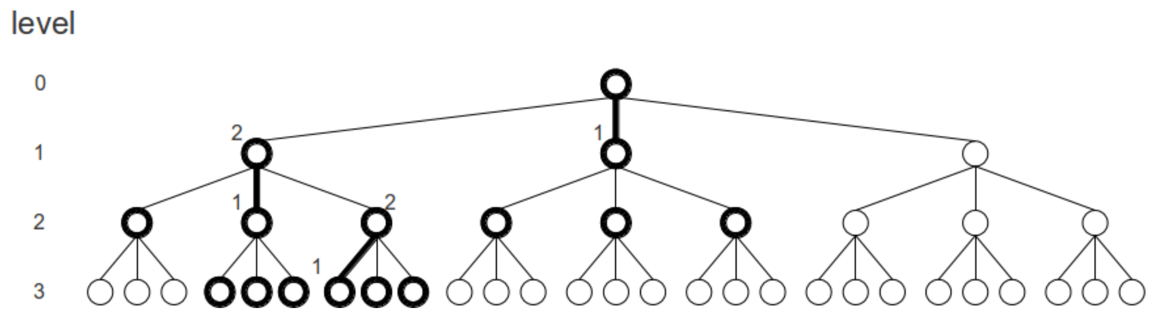


**Figure 2.3:** Illustration of GNP in a vocabulary tree with branch factor $k = 3$ and depth $L = 3$

According to the authors, GNP improves quality of search results since more nodes are being considered when traversing a tree, this confirms some of the findings in [17] saying that increasing the branching factor for a fixed vocabulary size brings some improvement.

On the premise that typically database images contain considerable overlap since they might depict the same locations but from slightly different viewpoints, the authors propose using only the most informative ones features to build a vocabulary tree. However the improvement gained with the optimal vocabulary construction constraints the database to be fixed and cancels the potential on-the-fly insertion of new objects into the database.

The intuition behind selecting the most informative features is finding those which occur in all images of some specific location, but rarely or never occur anywhere outside of that single location. To this end they propose using the formal concept of information gain. I'll not get into any details on the computation of this gain, it is enough saying that it measures how informative a visual word is about a specific location.

Once the gain of each location with respect to the visual words describing it is computed, the most informative features from each image are selected, and finally a tree containing only these informative features is constructed. The authors report that a vocabulary tree built in this manner degrades slower with the increasing size of the database.

### 2.2.2 RootSIFT, DQE and SPAUG

In [1] it is considered the problem of large scale landmark recognition. In order to improve recall in the retrieval pipeline the authors propose three technique to address the problems of feature detection drop-out, presence of noise in the descriptors, inappropriate metrics for descriptor comparison, and loss due to quantization[2].

**RootSIFT**

Noticing that Euclidean distance often yields inferior performance in the task of histogram comparison, the authors propose using a Hellinger kernel instead of the standard Euclidean kernel to measure similarity between descriptors.

The Euclidean distance $d_E(x,y)^2$ for a pair of L2 normalized histograms $x$ and $y$ is

$$d_E(x,y)^2 = \|x-y\|_2^2 = \|x\|_2^2 + \|y\|_2^2 - 2x^Ty = 2 - 2S_E(x,y)$$

where $S_E(x,y) = x^Ty$ is the Euclidean kernel. On the other hand the Hellinger kernel for two L1 normalized histograms, $x$ and $y$, is defined as

$$H\left(x,y\right) = \sum_{i=1}^{n} \sqrt{x_i y_i}$$

Computing SIFT descriptors similarity using a Hellinger kernel can be easily in two steps: 1) map descriptors from the original SIFT space to the RootSIFT space by taking the element wise square root of the L1 normalized SIFT descriptors and 2) computing the Euclidean distance. Such scheme is equivalent to using the Hellinger kernel to compare the original SIFT vectors: $d_E\left(\sqrt{x},\sqrt{y}\right)^2 = 2 - 2H\left(x,y\right)$, this is because $S_E\left(\sqrt{x},\sqrt{y}\right) = \sqrt{x}^T\sqrt{y} = H\left(x,y\right)$ and $S_E\left(\sqrt{x},\sqrt{x}\right) = \sum_i^n x_i = 1$, i.e. the resulting RootSIFT vectors are L2 normalized.

RootSIFT can be thought as an explicit feature map from the original SIFT space to the RootSIFT space, such that performing the scalar product there is equivalent to computing similarity with the Hellinger kernel in the original space. This mapping might be seen as a variance stabilizing transformation since its effect is that of reducing the large bin values relative to the smaller bin values. A great advantage of this technique is that it can be effortlessly involved in any retrieval pipeline by simply replacing SIFT with RootSIFT.

**Discriminative Query Expansion (DQE)**

Query expansion methods combine BoF vectors results to overcome problems such as feature detection drop-out, occlusion, noisy description and quantization errors. The standard Average Query Expansion (AQE) combines the top spatially verified BoF retrieval results by averaging them together with the query BoF vector, then uses the resulting expanded query BoF vector to re-query the database. Discriminative Query Expansion (DQE) works similarly except that instead of idf-weighting the expanded query BoF vector, it is used a vector of weights learned by training a linear Support Vector Machine (SVM) over the BoF retrieval results where the BoF

[2]Quantization problems: 1) loss of information about the original descriptors, 2) misassignment of descriptors to visual words.

vectors used to enrich the query conform the positive training data while BoF vectors corresponding to images with low scores provide the negative training data. Having a linear vector of weights enables ranking images using the inverted index in the same manner than in classical BoF retrieval.

Some advantages of DQE in comparison with other query expansion methods are:

- It is at least as computationally efficient as AQE with a negligible overhead of training a linear SVM.
- It benefits from adding distracting images since it might learn a better weight vector in case the new images are picked as negative examples.
- It distinguish between positive and negative words, since the vector of weights can assume either positive or negative values.

**Spatial database-side feature augmentation (SPAUG)**

Somewhat complementary to query expansion, database-side feature augmentation (AUG) tries to overcome the same problems as query expansion but on the database side. It can be thought of as query expanding each image of the database and replacing the original BoF vector of the image by its average query expanded version. However, here the augmentation process considers not only spatially verified visual words but all visual words from neighboring images in a matching graph[3]. Doing so might include many visual words corresponding to areas outside of the image, to tackle down this issue, spatial-database feature augmentation (SPAUG) augments the query using only visual words estimated to be visible in the augmented image. To determine the visibility of visual words, it is employed an homography between the two images under consideration, such transformation is already available from the spatial verification stage of the matching graph construction. SPAUG achieves a considerable improvement in retrieval performance albeit at the cost of additional storage requirement.

## 2.3 Low computational resources

As we move the problem to the mobile scenario, increases the need of adequate techniques which are able to cope with the unique characteristics of mobiles. The solutions range from using descriptors with a small footprint to alternative quantization techniques. In this section are reviewed some of this techniques.

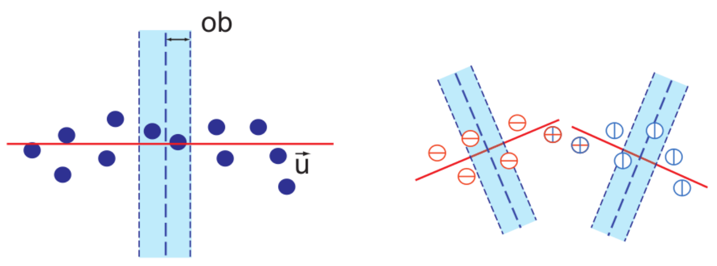
### 2.3.1 Multiple Hypothesis Vocabulary Tree

In the frame of the mobile location recognition problem a popular option to achieve very low retrieval times is performing the feature quantization on the handheld de-

[3]In a matching graph nodes represent images while edges signify that images have an object in common

vice and transferring compressed BoF vectors to the server. To do so it is necessary to cope with the limited processing capabilities of handhelds, and hence the quantization of high dimensional feature vectors should be done at very low complexity. In this order of ideas, in [29] the authors propose the Multiple Hypothesis Vocabulary Tree (MHVT) which increases the probability of assigning matching feature descriptors to the same visual word by iteratively separating the space with hyperplanes and introducing an overlapping buffer around the separating hyperplanes at each level of tree. This scheme for data organization, illustrated in Fig. 2.4, allows for a soft quantization and an adaptive clustering approach.

The MHVT defines a binary tree where each node defines a separating hyperplane, being it binary the query times are reduced. The tree is iteratively constructed by determining at each node an hyperplane that separates in two the data points assigned to the node, resulting in two mutually exclusive child nodes. To find such a plane it is taken the median value of the projection of all data points (assigned to the node) onto a vector $\tilde{u}$ in the direction of maximum variance. The normalized vector $\tilde{u}/\left\|\tilde{u}\right\|$ is the normal vector of the separating hyperplane at the median value.



(a) Quantization step 1 (b) Quantization step 2

**Figure 2.4:** Illustration of a separating hyperplane and an overlapping buffer

To increase the probability of assigning matching descriptors to the same visual it is used an *overlapping buffer* with width $\tau \cdot \left\|\tilde{u}\right\|$ around the separating hyperplane. In this way descriptors lying close to the splitting boundaries, which have a high probability of matching to a descriptor in the neighboring node, are assigned to both child nodes. The achieved result is that the differentiation is delegated to the child nodes where the probability of lying far from the separating hyperplane is larger. The overlapping buffer is used only to quantize the database feature vectors since using it on the query feature vectors would clearly increase the query time.

The iterative vocabulary construction procedure continues until: 1) there are less descriptors than a maximum *naive count* or 2) until a certain percentage $p$ of features fall in the buffer, case in which the cluster is assumed to be self-contained cluster, i.e. the variance is similar to any direction and hence there is no gain in further splitting.

The quantization of a feature vector consist in propagating it down tree by at each level performing the dot product with $\tilde{u}$ to evaluate on which side of the hyperplane it is located.

Instead of a hierarchical weighing as done with vocabulary trees, here is used a weighting scheme based on the plausibility of the query descriptor quantization. The weight of a visual word $\alpha_i$ is the probability of assigning matching descriptors to it. This probability corresponds to the probability of quantizing matching features to the same node $(1 - P_f)$ in all quantization steps $m$

$$\alpha_i = \prod_m (1 - P_{f,m})$$

where $P_f$ is the probability that a matching feature is incorrectly quantized to the neighboring child node. An efficient scoring scheme where only non-zero terms are considered, supported by inverted files, and including word weighting, goes as follows:

$$\sum_i \alpha_i |q_i - d_i|^P =$$
$$\sum_i \alpha_i |q_i|^P - \sum_{i|d_i \neq 0} \alpha_i |q_i|^P + \sum_i c |d_i|^P - \sum_{i|q_i \neq 0} c |d_i|^P + \sum_{i|q_i \neq 0 \wedge d_i \neq 0} \alpha_i |q_i - d_i|^P$$

The multiple hypothetical paths approach concentrates the scoring energy on the leaf nodes, avoiding extensive time consuming random memory accesses. Its quality in terms of retrieval performance is somewhat comparable with AKM albeit a considerable decrease in computational complexity.

### 2.3.2 Adaptive partial vocabularies

In [30] the authors develop a mobile location recognition system receiving taking as input a video sequence. They notice that not all of the vocabulary words are necessary to quantize a sequence of images but rather just a part of it. Only the words present in a single query frame are necessary to quantize it, more formally if $F = \{f_1, f_2, \ldots, f_N\}$ is the set of features of one query frame, $V = \{v_1, v_2, \ldots, v_L\}$ is the full set of visual words, and $Q(F|V)$ determines the subset of visual words representing a particular video frame or partial vocabulary then it holds that $Q(F|V_F) = Q(F|V)$ where the partial vocabulary $V_F$ is the subset of visual words representing the frame itself, i.e. the result of the quantization would not change.

In practice having a partial vocabulary that contains only the words of a certain video sequence is not possible and hence the current partial vocabulary $V_F$ is extended by other subsets $S$ of the full vocabulary $V$ which have a sufficiently high probability of including the unknown final vocabulary, that is

$$Q(F|V_F \cup S) = Q(F|V)$$

The partial vocabularies used to extend $V_F$ are composed by the visual words of the panoramas from the periodically retrieved top K location estimates which work as prior knowledge on the location. The probability that the partial vocabularies associated to the frames recorded at these locations are part of $V_F$ is very high. Such words together with their associated inverted files are periodically transmitted to the client to allow for a local pose estimation on the handheld within a limited area.

Further improvements include using a priority queue of locations governed by the probability of representing the actual location and a temporal filtering approach such as Bayesian filters to determine such probabilities.

Since there is a high probability that the correct location is among the top K results, but possibly poorly ranked, this approach can be thought as a resorting among this short list. This might result in an increase in precision when compared to using the full vocabulary.

Though this approach is not followed in the current work it stands as a state-of-art technique when comes to implement a mobile visual location recognition system.

### 2.3.3 Binary BoF using Hierarchical K-Medians

In [7] the authors deal with the problem of close loop detection, a common problem in SLAM that consists in detecting when the robot has returned to a past location after discovered new terrain for a while. The authors present an algorithm to detect loops and establish point correspondences between images in real time using the BoF model followed by geometrical check.

In particular they concentrate on the problem of feature extraction which is the most expensive one in any SLAM algorithm. To achieve the goal of low computation time they propose the use of binary features as input to the BoF model. This approach is an adaptation of the vocabulary tree technique where they discretize a binary descriptor space instead of a real-valued one as in the case of SIFT descriptors. To address the task of clustering binary data they perform k-medians with k-means++ seeding, truncating to zero the non-binary medians, and using hamming distance to measure similarity between feature vectors. For the rest of the BoF model it is the same than usual, database stored in an inverted index, hierarchical weighting applied only to leaf nodes and based on entropy relative to the root node, and L1 distance to compute BoF vectors similarity.

A remarkable contribution is the use of a direct index (see Fig. 2.5) to speed up geometrical verification. It consist in a map structure that for each image stores the nodes at a certain level that are ancestors of the words present in the image, as well as the list of the local features associated to each node. The direct index acts as an approximation to nearest neighbors search, it is updated whenever a new image is added to the database.

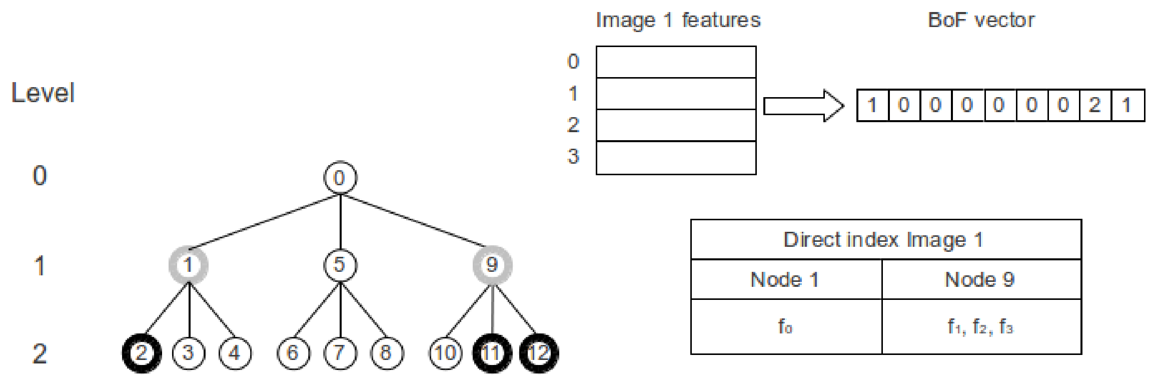


**Figure 2.5:** Example of a vocabulary tree and a direct index

### 2.3.4 Binary BoF using K-Majority

In [8] it is proposed a model for using binary features as input to the BoF model. They concentrate in solving the clustering problem since classical k-means is not applicable to binary data streams as explained in sec. 4.1. They propose k-majority, here presented in Algorithm 2.2, a variation of the k-means algorithm where:

1. Instead of computing the Euclidean distance they compute the Hamming distance, a more reasonable choice for comparing binary strings.
2. A voting scheme, based on Hamming distance, to compute a centroid from a set of binary vectors assigned to the cluster it represents.

**Algorithm 2.2** K-majority algorithm

```
Input: a collection D of binary vectors
Output: cluster assignments
Randomly generate $k$ binary centroids $C$
while centroids not changed do
    for $d \in D$ do
        $c_d \leftarrow \underset{c \in C}{argmin}\, HammingDistance\,(c, d)$
    end for
    for $c \in C$ do
        for $d \in D | c_d = c$ do
            $v$ accumulates $d$ votes
        end for
        $c' \leftarrow Majority\,(v)$
    end for
end while
```

The voting scheme, illustrated in Fig. 2.6, operates as follows: 1) given a set of binary vectors assigned to some cluster $C$, they are bitwise accumulated into an accumulator $v$ such that each of its elements represents the count of bits on that position set to 1, 2) to compute the cluster centroid the majority rule is applied for each element $v_i$ in $v$

$$c_i = \begin{cases} 1 & if \quad v_i \geqslant \left\lceil \frac{|C|}{2} \right\rceil \\ 0 & otherwise \end{cases}$$

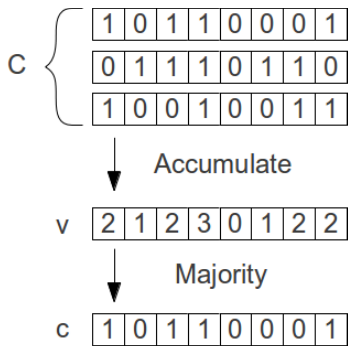


**Figure 2.6:** Voting scheme to compute centroids from clusters of binary data

An important characteristic of this approach is that both the Hamming distance and the majority voting step work directly on the byte packed vector string rendering k-majority implementation more efficient than others working over real-valued data.

## 2.4 3D models

Either building a 3D Structure-from-Motion (SfM) model for 2D-to-3D matching or using an images dataset generated from projecting panoramic images to a 3D city model, using 3D information is an effective way to tackle the problems result of mapping 3D world objects onto a 2D space. In this section I review some of the recently proposed state-of-art methods for visual localization using 3D information.

### 2.4.1 Image matching using aligned panoramic images

Database construction is a challenging yet crucial task before any image-based localization pipeline takes place. A popular approach consist in crawling photos from online collections such as Flickr, nevertheless this images tend to be poorly labeled, disorganized and sparse in the real world. A different approach uses high quality panoramic images aligned to a 3D city model consisting of extruded buildings from footprint and elevation data.

### Facade-aligned images

In [2] the recognition problem is reduced to a homothetic one where feature invariance is simplified by fully removing 3D rotation and only scale and offset are considered. In this sense, it is proposed a fast stratified geometric verification algorithm considering only scale and offset variability. Further leveraging homotheticity achieved through geometric rectification, the authors contribute an scheme for pose estimation from 2D-2D features correspondences.

The key behind their approach is using an orthophoto representation of the database. To obtain it they projected image data consisting of panoramic images systematically captured by a vehicle driving through the streets onto the 3D city model result of extruding 2D buildings floor plans by means of height data followed by rendering synthetic orthoviews from them. Then local features are extracted from database of images, in this case DoG keypoints and upright SIFT which result in more discriminative descriptors. Doing so all effects of 3D rotation and perspective from the image data are removed.

Following the pipeline of a basic landmark recognition system, visual vocabulary and database are built using vocabulary trees approach. In the online stage before the query image enters into the pipeline all 3D rotation effects must to be removed. To do so it is employed an homography that maps vanishing points, extracted from automatic detected line segments (filtered using camera calibration matrix obtained with the assumed available information from the images: focal length and rough estimate of the orientation), to canonical directions. Once ready it is performed BoF retrieval and a short list of most visually similar images is obtained.

The proposed geometric verification scheme is a 3 DOF approach enhanced for the homothetic case and inspired by the kernel density estimation technique. It subsequently estimates three position related parameters: distance, horizontal offset and vertical offset with respect to building's surface. The basic idea is that true correspondences should exhibit consistent differences. To determine the distance between the image plane and the building facade we look at scale ratio differences $\rho i := \sigma_{query,i}/\sigma_{facade,i}$ where $\sigma_{query,i}$ and $\sigma_{facade,i}$ are the scale of a pair of features from the query and database images correspondingly. Then it is required that the differences of logarithmic scale ratios $|log\,\rho_i - log\,\rho_j| \leq log\,t$ agree up to a threshold $\sigma = log\,t$ which depends on the expected scale estimation uncertainty, e.g. $log\,t = 0.15$. The the scale ratio $\rho^*$ with the most support is determined by finding the argument that maximizes the kernel density function

$$\rho^* = arg\,\underset{\rho}{max} \sum_i e^{-\frac{(\rho-\rho_i)^2}{2\sigma^2}}$$

All the pairs of features corresponding to scale ratios in the interval $\rho^* \pm 2\sigma$ are considered inliers. Using this estimate and the database image true scale, the features coordinates of both query and database images are expressed in meters.

The horizontal and vertical offset are determined analogously, the coordinates differences $\xi_i := x_{query,i} - x_{facade,i}$ and $v_i := y_{query,i} - y_{facade,i}$ are required to exhibit consistent behavior, hence $|\xi_i - \xi_j| \leq d$ and $|v_i - v_j| \leq d$ where $d$ is a resolution independent translation tolerance expressed in a known unit, e.g. $d = 0.3m$. The horizontal and vertical offset estimates are obtained using the same scheme as with the scale ratio, maximizing a kernel density function. Using these estimates the set of scale inliers is filtered in two inlier sets (one for each coordinate) which are then intersected into the final inlier set of the geometric verification, its cardinality is used to re-rank the candidates in the short list of candidates.

Fully determining camera pose with respect to the facade results straightforward considering: 1) that the query image is rectified (and hence the image plane is parallel to the facade plane) and 2) the scale ratio, horizontal and vertical offset estimates (obtained from the 3 dof geometric verification).

- **Position:** assuming the camera is calibrated with focal length 1 pixel and principal point at zero, the perpendicular distance of the camera from the facade is

$$pos_z = res_{facade} \cdot \sigma_{facade} / \sigma_{query}$$

  where $res_{facade}$ is the resolution of the orthophoto in pixel/meter. Analogously the camera's parallel distance to the facade can directly be computed from the feature position

$$pos_x = res_{facade} \cdot (x_{facade} - \sigma_{facade} / \sigma_{query} \cdot x_{query})$$
$$pos_y = res_{facade} \cdot (y_{facade} - \sigma_{facade} / \sigma_{query} \cdot y_{query})$$

- **Orientation:** the local camera orientation with respect to the wall is the inverse vanishing point rotation.

Finally facade's pose in the world are used to convert the relative coordinates with respect to the faced to absolute world coordinates.

### Combining viewpoint- and facade-aligned images

In [5] the authors propose an image retrieval pipeline fusing two popular representations of street-level image data, facade-aligned and viewpoint-aligned, containing complementary information. Doing so they improve recall rates while obtaining a retrieval performance competitive with previous results but at city scale. They also improve feature detection by enhancing low contrast parts of database images using histogram equalization. In addition they develop an effective method to incorporate priors on user's position (e.g. using cell towers information or GPS coordinates associated with a query image) either on GPS-aware or GPS-agnostic schemes.

With the aid of specialized equipment it were captured many panoramic images conforming a uniform city cover. Then by means of approximate 3D building models of the city it were generated masks to select visible buildings. Using this masks the panoramas were converted to a set of perspective images, either perspective central (PCI) or perspective frontal (PFI).

The recognition pipeline consist in two parallel pipelines, see Fig. 2.7, one for PCIs and another for PFIs.

- The PCI pipeline is just as the pipeline of a basic landmark recognition system using SIFT descriptors with the additional filtering step to exclude geographically distant landmarks using GPS coordinates associated with the query image (when GPS is available).
- The PFI pipeline works analogously, its difference lies in the fact that it works on top of facade-aligned images. Before the PFI pipeline is run, the query image must be rectified by detecting line segments and then estimating vanishing points.

A remarkable difference between the PCI and PFI pipelines is in the geometric verification stage: PCI uses RANSAC to estimate a 2D affine model, while PFI uses a 3 degree-of-freedom (DOF) scale and offset check.

The result of both pipelines is merged by taking their short lists of inlier counts after geometric verification, $\{N_{pci,1}, \ldots, N_{pci,n}\}$ for PCIs and $\{N_{pfi,1}, \ldots, N_{pfi,n}\}$ for PFIs, weighting the PFI inlier counts by a parameter $\alpha$ then sorting the concatenated lists $\{N_{pci,1}, \ldots, N_{pci,n}, \alpha N_{pfi,1}, \ldots, \alpha N_{pfi,n}\}$ and retaining the top $n$ candidates out of the $2n$ candidates.

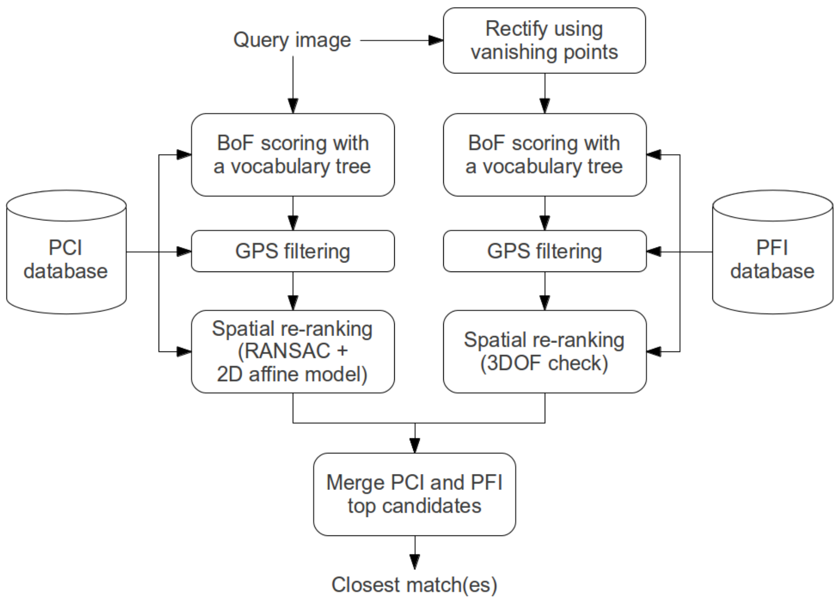

Query image
Rectify using vanishing points
BoF scoring with a vocabulary tree
BoF scoring with a vocabulary tree
PCI database
GPS filtering
GPS filtering
PFI database
Spatial re-ranking (RANSAC + 2D affine model)
Spatial re-ranking (3DOF check)
Merge PCI and PFI top candidates
Closest match(es)

**Figure 2.7:** Recognition pipeline using PCIs and PFIs

An interesting contribution is an scheme for incorporating prior vague location knowledge in the pipeline. Given noisy GPS coordinates for a query image, only the database images within a radius $R$ 300 meters (empirically determined value) of $Q$ are scored. To address geographic proximity search it is used a single approximate nearest neighbor tree trained for the whole city.

### 2.4.2 Matching images to 3D SfM models

In [14] the authors implement a visual localization system based on results of previous works. The system addresses two different problems: 1) an image matching problem of the query image to the database images which is addressed using the BoF model, and 2) an image registration problem of the query image. After assigning the visual similarity score to all the database images with respect to a query image, the reconstructed 3D scene models associated to the candidates in the short list of top candidates are retrieved. These 3D scene models are used to register the query images and compute scene's location.

For the offline construction of the 3D scenes models the authors use a joint geo-visual clustering for scene detection and a SfM algorithm implemented in Bundler[4]. For the online retrieval of the short list of candidates they use the BoF model with a vocabulary built using Hierarchical K-Means, plus a geometry verification step run over the short list. For 3D scene models retrieval they implement a simple voting scheme based on the majority of coincident descriptors between query image and database images.

In contrast with previous approaches this one provides additionally:

- User location: it is the outcome of the location recognition problem, but here is computed in a particular way using the reconstructed scene geometry, more concretely, the estimated pose (3-point method using focal length) and the decomposition of the estimated projection matrix $P = K\,[R|t]$ associated to the camera from which the query image was shot.
- Scene location: estimated as the mean of the points that can be observed from the query image viewpoint.
- Viewing direction: estimated using the rotation matrix $R$ result of the decomposition of $P$.

[4]Bundler: Structure from Motion (SfM) for Unordered Image Collections - http://www.cs.cornell.edu/~snavely/bundler/

# 3 Binary Features

Binary features [4, 13, 26, 32] are a relatively recent class of local feature descriptors aiming at reducing processing and storage requirements by directly computing the descriptors from pixel-level comparisons. They are both fast to compute and to match, but more importantly they provide a compact representation and have small memory requirements. They have also proven to compete at the same level of state-of-art techniques such as SURF and SIFT, even outperforming them on cases where rotation and scale invariance are not strict requirements [4].

Some worth mentioning common properties of this class of descriptors are: 1) they are bit-strings typically result of pairwise intensity comparisons, 2) as similarity measure it is typically used Hamming distance because it is meaningful when comparing binary strings, further it may be efficiently implemented using SSE[1] instructions and specific bitwise hardware instructions, and 3) they generally employ some some sort of sampling pattern.

In mobile location recognition latency due to transmission during feature uploading is a major bottleneck as noted in [30], in this sense binary visual features pose as an improvement due to it's compactness. In the next sections some of the state-of-art techniques for keypoints description using binary strings are addressed encompassing as well real-time detectors which lie at the core of this features speed.

## 3.1 Feature detectors

## FAST (Features from Accelerated Segment Test)

Was initially proposed in [24] and subsequently improved in [25] by the same authors, it applies a pixel-level criterion acting on the intensity values and subsequently employs a machine learning algorithm to improve speed. This detector belongs to a class of detectors that work by examining small image patches and determining if it "looks" like a corner. It has become a popular method in cases where real-time constraints exist.

The original detector proposed in [24] introduces the segment test criterion, for a candidate pixel $p$ considers a surrounding circle of 16 pixels (a Bresenham circle of

[1]Streaming Single Input Multiple data (SIMD) extensions

radius 3) and applies a classification rule to decide if $p$ is a corner or not. The rule goes as follows, if there exist a set of $n$ contiguous pixels in the circle which are all brighter than the candidate pixel plus a threshold, i.e. $I_p + t$, or all darker than the candidate pixel minus a threshold, i.e. $I_p - t$, then $p$ is a corner. Fig. 3.1 illustrates the idea.

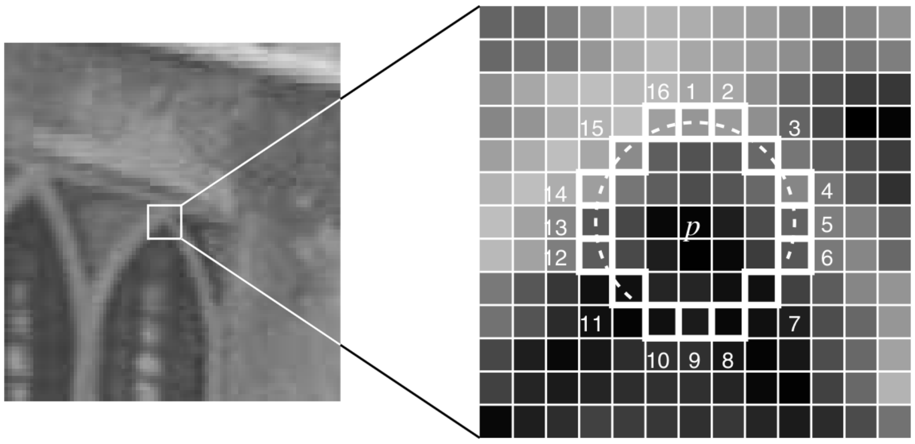


**Figure 3.1: FAST corner detection active principle.** 12 point segment test corner detection in an image patch. The highlighted squares are the pixels used in the corner detection. The pixel at $p$ is the center of a candidate corner. The arc is indicated by the dashed line passing through 12 contiguous pixels which are brighter than $p$ by more than the threshold.

The authors empirically determined the number $n$ of contiguous pixels to be 12 because it enables a high-speed test for non-corners rejection where only four pixels are examined, first 1 and 9 then 5 and 13. Since at least 12 contiguous pixels of the 16 need to be all brighter than $I_p + t$ or darker than $I_p - t$ then if less than three of the four examined pixels don't approve the test criterion $p$ is not a corner, otherwise $p$ might be a corner and the full segment test criterion is performed for the remaining pixels.

The segment test detector is further improved in [25] addressing some identified weaknesses. It is said to run in two stages. First it starts by running the original detector based on the full segment test criterion on a set of training images which is suggested to be from the target application domain. For each candidate corner $p$ is obtained a boolean variable $K_p$ indicating if it was classified as a corner or not.

In the second stage it uses algorithm ID3 (Iterative Dichotomoiser 3) for learning a decision tree from a dataset based on an information measure. It starts by selecting a pixel $x \epsilon \{1 \dots 16\}$ relative to a candidate pixel $p$, denoted by $p \to x$, and classifying it in one of three states $d$, $s$, or $b$ according to the following rule:

$$S_{p\to x} = \begin{cases} d & I_{p\to x} \leq I_p - t \qquad (darker) \\ s & I_p - t < I_{p\to x} < I_p + t \quad (similar) \\ b & I_p + t \leq I_{p\to x} \qquad (brighter) \end{cases}$$

Applying this classification rule for a given $x$ and for all $p \epsilon P$, where $P$ is the set of all pixels in the set of training images, yields three subsets of $P$ namely $P_d$, $P_s$, $P_b$. The selection of $x$ is done such that the information gain about whether $p$ is a corner, $IG\left(K\right)$, is maximized

$$IG\left(K\right) = H\left(P\right) - H\left(P_d\right) - H\left(P_s\right) - H\left(P_b\right)$$

this means an exhaustive search through all possible values of $x$ is performed. The entropy with respect to an attribute, in this case $K$, for a set, in this case $P$, is computed as:

$$H\left(P\right) = -c\log_2\left(\tfrac{c}{c+\bar{c}}\right) - \bar{c}\log_2\left(\tfrac{\bar{c}}{c+\bar{c}}\right)$$

$$\text{where } c = \left|\{p \mid K_p\}\ is\,true\right| \text{ (number of corners)}$$

$$\text{and } \bar{c} = \left|\{p \mid K_p\}\ is\,false\right| \text{ (number of non corners)}$$

Once $x$ was selected for $P$ the process is iteratively repeated for each of the resulting subsets $P_d$, $P_s$, $P_b$ until the entropy of a subset is zero, that is when the elements in a subset are either all corners or non-corners. This results in a decision tree that classifies correctly all pixels from the training set of images and can be used to classify pixels from other images. At the implementation level the authors proposed to convert the resulting decision tree into a long string of nested if-else statements in C-code which is further optimized by means profile-guided optimization.

In order to eliminate detected features on unseen images which are too close one another it is carried out a process of non-maximal suppression. Since the segment test doesn't compute any corner response measure, a scoring function, $V$, must be computed for each detected corner to eliminate features which have an adjacent corner with higher $V$. The employed function is a modified version of the sum of the absolute difference between the pixels in the contiguous arc and the center pixel:

$$V = max\left( \sum_{x \epsilon S_{bright}} \left|I_{p\to x} - I_p\right| - t \quad , \quad \sum_{x \epsilon S_{dark}} \left|I_{p\to x} - I_p\right| - t \right)$$

$$\text{where} \quad \begin{array}{c} S_{bright} = \{x \mid I_{p\to x} \geq I_p + t\} \\ S_{dark} = \{x \mid I_{p\to x} \leq I_p + t\} \end{array}$$

According to the authors experiments using $n = 9$ provides the highest repeatability and lowest timing hence providing optimal performance. The final detector achieves a big speed-up over other detectors without any sacrifice in terms of quality making it suitable for real-time applications. Likewise it is highly repeatable and reliable, i.e. its able to obtain features corresponding to the same real-world 3D locations out of images with different viewpoints of the same scene. In the other hand it is not robust to high levels of noise since no noise reduction step is considered, it tends to perform poorly on images with only large-scale features, and it critically depends on the selected threshold $t$.

Proven to be a very efficient basis for feature extraction a number of improvements to it have been proposed, one of the most remarkable is AGAST[15] (Adaptive Generic Accelerated Segment Test) a high speed improved version of FAST which constructs an optimal decision tree in a extended configuration[2] space and achieves a generic scene independent detector by combining specialized trees, hence overcoming the scene dependence of FAST.

## 3.2 Feature descriptors

## BRIEF (Binary Robust Independent Elementary Features)

Originally introduced in [4] is one of the earliest works using binary strings for feature points description. It computes directly a binary string whose individual bits are obtained by comparing the intensities of $n_d$ test locations in a certain sampling pattern applied to an image patch around the keypoint under description. The binary test $\tau$ on patch $p$ of size $S \times S$ is defined by:

$$\tau\left(p; x, y\right) := \begin{cases} 1 & p\left(x\right) < p\left(y\right) \\ 0 & otherwise \end{cases}$$

Where $p\left(x\right)$ and $p\left(y\right)$ are the pixel intensities in a smoothed version of $p$ at locations $x$ and $x$ correspondingly. The descriptor is an $n_d$-dimensional bit-string[3]:

$$f_{n_d}\left(p\right) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau\left(p; x, y\right)$$

Since the tests take into account only single pixels information they are very sensitive to noise,

to reduce it the patches are pre-smoothed with a Gaussian kernel with variance $\sigma^2 = 2$ and a discrete kernel window of $9 \times 9$ pixels. According to the authors

[2]Configuration in this context should be understood as the possible combinations of $N$ state pixels on the circle surrounding a keypoint $p$ as considered by FAST and $n_s$ possible pixel states, i.e. ${n_s}^N$

[3]As a matter of notation BRIEF-k, being $k = n_d/8$, stands for the number of bytes required to store the descriptor.

experiments such combination of parameters achieves a constant recognition rate in the task of descriptor's matching. The obtained effect is a reduction of the noise and an increase in the descriptor's stability and repeatability.

As for the spatial arrangement of the test locations, BRIEF uses a non-local, non-symmetrical and non-regular sampling pattern in which the test locations are randomly selected from an isotropic Gaussian distribution with zero mean and variance $\sigma^2 = \frac{1}{25}S^2$ which according to the authors achieves the best recognition rates. Fig. 3.2 shows an example of such pattern.

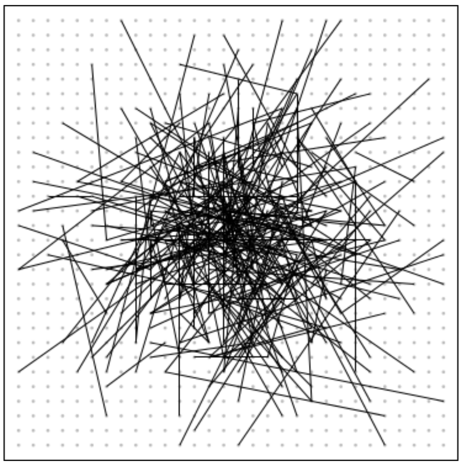


**Figure 3.2:** BRIEF sampling pattern

Some limitations of the approach pointed by the authors are:

- **Invariance to scale and rotation.** It was not designed to be neither rotation or scale invariant but it tolerates small amounts of rotation, up to 10 to 15 degrees as quantified by the authors, at the cost of speed and recognition rate. It is actually suggested to avoid orientation correction when it is not actually required, argument that is further confirmed on [11] where a comparative study of combinations of traditional and novel binary feature detector/descriptor pairs is performed using a set of comprehensive and fair performance measures.
- **Uninformative in case of large monochrome areas.** The performed intensity difference tests rely on areas with enough texture changes, something typically meet on outdoor images.
- **The number of test locations.** There is no rule of thumb on what is an ideal number of test locations but using very few ones might lead to a not very reliable descriptor, BRIEF-16 is a reasonable lower bound to the approach. The number of test locations is related in the case of matching to the baseline of images under matching. According the authors using $n_d$=128, 256, or 512 achieves good keypoint description and yields a good compromise between speed and storage efficiency.

A success key of BRIEF is the fact that the carried tests can be thought of as evaluating the sign of the derivatives within a patch along the lines specified by the test locations, which reminisces the histogram of local gradient directions employed by SIFT.

## ORB (Oriented FAST and Rotated BRIEF)

Aiming at providing an orientation estimator to the BRIEF descriptor in [26] the authors propose ORB (Oriented FAST and Rotated BRIEF). It is the result of combining: 1) Oriented FAST (oFAST), a scale and rotation invariant version of the FAST keypoint detector together with 2) Rotated BRIEF (rBRIEF), a steered version of BRIEF where the test locations are fixed to 256 and improved by selecting them via machine learning in order to maximize its variance and minimize its correlation, hence more discriminative and informative.

### oFAST

The proposed detector is a variant of the original FAST keypoint detector, further explained in section sec. 3.1, using a circular radius of 9 pixels and augmented with a pyramid scheme to address scale invariance and a Harris corner filter as cornerness measure to reject edges then rank the resulting keypoints and choose the top N which are more likely to be a corner according to the adopted measure.

Keypoint orientation estimation is estimated using the intensity centroid, a measure of corner orientation grabbed from the work of Rosin in [23] which assumes that corner's intensity $C$ is offset from its center $O$ and defines it as

$$C = \left(\tfrac{m_{10}}{m_{00}}, \tfrac{m_{01}}{m_{00}}\right)$$

where $m_{pq}$ is the moment of a patch in a circular region of radius $r$ equal to the patch size and is defined as

$$m_{pq} = \sum_{x,y} x^p y^q I\left(x, y\right)$$

The keypoint's orientation is then the angle formed between the horizontal and the vector from the feature location to the intensity centroid $\vec{OC}$, more formally

$$\theta = atan2\left(m_{01}, m_{10}\right)$$

Accounting for stability of the measure, the authors mention that it might become unstable as $|C|$ approaches 0, i.e. when the assumption about the corner's intensity is not meet, something that is unlikely to occur with keypoints detected using FAST but should be taken into account when paired with other detectors.

## steered BRIEF

To allow BRIEF to be invariant to in-plane rotation the estimated patch orientation $\theta$ is used to produce an steered version of BRIEF. It is determined by transforming the set $S$ of $n$ test locations by means of the rotation matrix $R_\theta$ associated to the estimated patch orientation.

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad S = \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{pmatrix} \quad S_\theta = R_\theta S$$

The resulting steered BRIEF operator forms the base to the final proposed descriptor.

$$g_{n_d}(p,\theta) := f_{n_d}(p) \mid (x_i, y_i) \in S_\theta$$

## rBRIEF

To achieve a descriptive and informative descriptor it is employed a learning method for choosing a good subset of binary tests. The method searches through all possible test locations to find those which have both high variance and low correlation. Before the algorithm starts, a training set of $k$ keypoints is set up and all possible pairs of non-overlapping $5\times5$ sub-windows (tests) from patch of size $31\times31$ are enumerated.

The algorithm runs as follows:

1. All the enumerated tests are run against the patches associated to the training set of keypoints. The result is a $k-length$ binary string for each possible test pair.
2. For each of the obtained $k-length$ binary strings the mean is computed and the associated test pairs are ordered according to the shortest distance of this mean to 0.5[4] forming a vector $T$. This vector $T$ are the test pairs ordered from the most to the less discriminative.
3. A greedy search on $T$ looking for the less correlated tests is performed. It starts by adding to the result vector $R$ the first element of $T$ then picking the next test from $T$ and considering its addition to $R$ by evaluating if its absolute correlation with respect to $R$ is within a threshold. This picking process is repeated until there are 256 tests on $R$ or until there are no more tests to choose, case in which the threshold is increased and the greedy search restarts.

The produced test locations are more evenly distributed and less correlated as noted by the authors after analyzing the distribution of eigenvalues in the PCA decomposition over 100K keypoints described using BRIEF, steered BRIEF, and rBRIEF.

A key advantage of ORB is that it is an order of magnitude faster than SURF, and over two orders faster than SIFT, as well as more resistant to image noise.

[4]The authors established 0.5 as the mean value with the highest variance

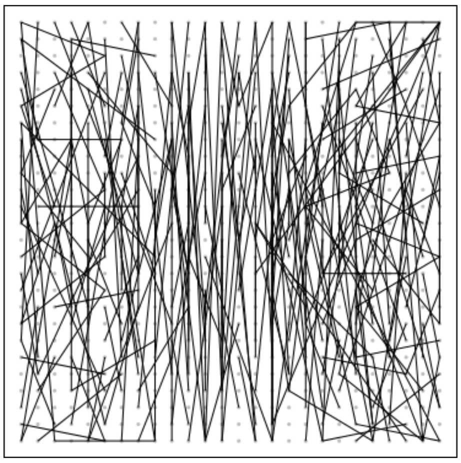


**Figure 3.3:** ORB sampling pattern

## BRISK (Binary Robust Invariant Scalable Keypoint)

Inspired by the efficiency of BRIEF and FAST in [13] the authors proposed a novel modular method for high-quality, fast keypoint detection, description and matching. The resulting keypoints are scale invariant (provided by the detector) and rotation invariant (provided by the descriptor) to a significant extent.

### Feature detection

To detect features it uses an scale invariant version of the FAST corner detectorsec. 3.1 augmented by means of a scale-space pyramid, performing non-maximal suppression and sub-pixel interpolation across all scales using a corner response function (measure of saliency) to indicate the cornerness of a pixel.

The considered scale-space pyramid consists of $n$ octaves $c_i$ and $n$ intra-octaves $d_i$ where $i = 0, 1, \ldots, n-1$ and typically $n = 4$. The octaves $c_i$ are the result of progressively downsampling the original image $c_0$ by a factor of 2, whereas the intra-octaves $d_i$, located between layers $c_i$ and $c_i + 1$, are obtained by downsampling also by a factor of 2 the initial intra-octave $d_0$ which in turn is the result of downsampling the original image $c_0$ by a factor of 1.5. In short if $t$ denotes scale then $t(c_i) = 2^i$ and $t(d_i) = 2^i \cdot 1.5$.

The keypoint detection algorithm runs as follows:

1. Apply separately the FAST 9-16[5] detector to each octave and intra-octave using a threshold $T$ to identify potential regions of interest.
2. Apply non-maximal suppression in scale-space to the obtained putative keypoints:

[5]FAST 9-16 means FAST detector using a surrounding circle around keypoint $p$ of 16 pixels radius and minimum 9 contiguous pixels required to declare $p$ as a corner.

a) The associated FAST score $s^6$ of a putative keypoint at scale $c_i$ is enforced to fulfill a maximum condition with respect to the FAST score of its 8 neighbors[7] at the same layer.

b) Same maximum condition is enforced over the corresponding $s$ on the layers above and below (intra layers), and over the s at $c_0$ with respect to the scores at its neighboring layers (inter layers)[8].

c) A special case is considered for the detection of maxima at octave $c_0$. To obtain FAST scores at virtual intra-octave $d_{-1}$ is applied FAST 5-8 on octave $c_0$ but the resulting scores are not enforced to be lower than the score of the keypoint under analysis at octave $c_0$.

3. Apply sub-pixel refinement to each detected maximum by fitting a 2D quadratic function in the least-squares sense to each of the three scores-patches (keypoint layer, the one above, and the one below) and finding the global maximum. This results in three sub-pixel refined saliency maxima.

4. Use the refined scores to fit a 1D parabola along the scale axis which yields at its maximum the final score and scale estimates.

   Fig. 3.4 illustrates the procedure of interest point detection in the scale-space.

5. Re-interpolate the image coordinates of the patches in the layers next to the determined scale.

## Keypoint description

Like BRIEF the BRISK descriptor is a binary string result of pixel intensities comparisons with two differences, it uses a fixed sampling pattern focused on maximizing descriptiveness and provides rotation invariance by estimating keypoint's characteristic direction.

The sampling pattern, illustrated in Fig. 3.5, defines $N$ sampling location $p_i \mid i = \left\{ 1 \quad \ldots \quad N \right\}$ equally spaced on circles of varying radius and blurring level, concentric with the keypoint. The applied Gaussian smoothing at each circle is for avoiding aliasing effects when sampling the image intensity of a point $p_i$ in the pattern, the applied quantity of noise $\sigma_i$ is proportional to the distance between the points on the respective circle.

[6] According to the authors the score $s$ is defined as the maximum threshold still considering an image point a corner without providing further details. The original authors of FAST proposed a scoring function to measure corner response which fits the purpose of the BRISK detector, see section sec. 3.1 for more details.

[7] Neighbors of the center in a patch of size $3 \times 3$

[8] In practice interpolation is applied at the boundaries of the patch since the neighboring layers, and so its FAST scores, are represented with a different discretization

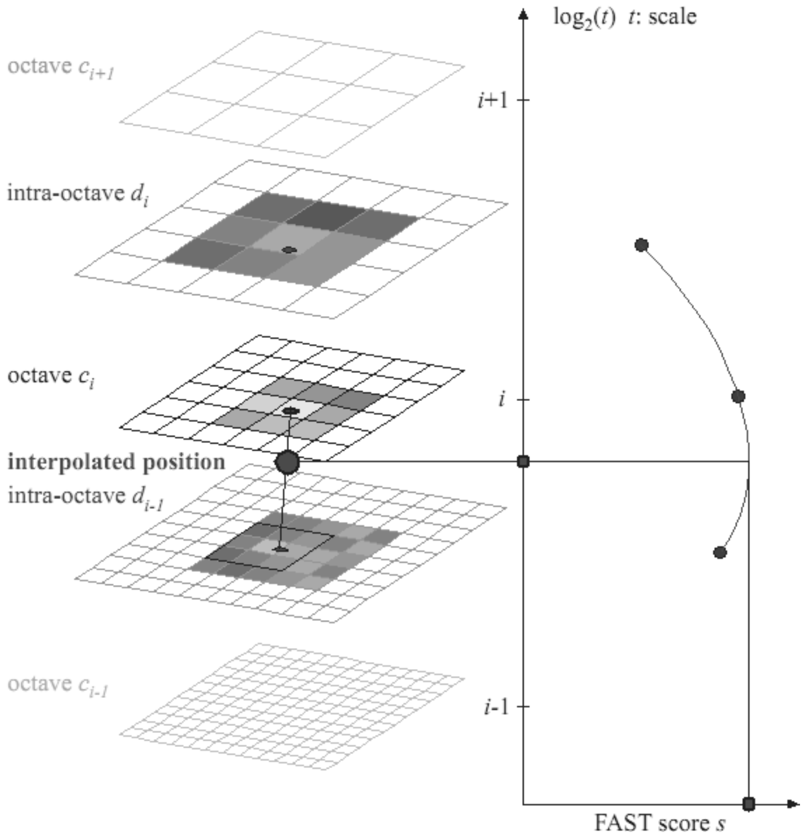


**Figure 3.4: Scale-space interest point detection.** A keypoint (i.e. saliency maximum) is identified at octave $c_i$ by analyzing the 8 neighboring saliency scores in $c_i$ as well as in the corresponding scores-patches in the immediately-neighboring layers above and below. In all three layers of interest, the local saliency maximum is sub-pixel refined before a 1D parabola is fitted along the scale-axis to determine the true scale of the keypoint. The location of the keypoint is then also re-interpolated between the patch maxima closest to the determined scale.

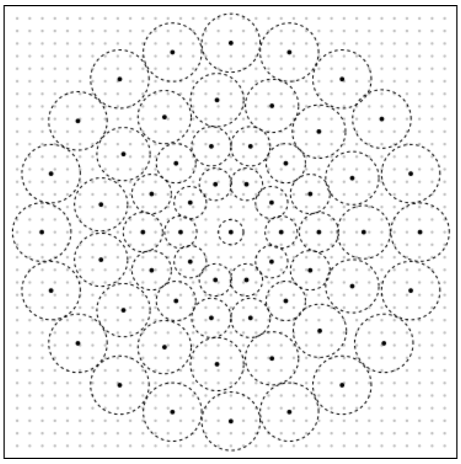


**Figure 3.5:** BRISK sampling pattern

The set of sampling-point pairs $\mathcal{A} = \{(p_i, p_j)\, \epsilon \mathbb{R}^2 \times \mathbb{R}^2 \mid i < N \wedge j < i \wedge i, j \epsilon \mathbb{N}\}$ is divided into short-distance pairings $\mathcal{S}$ and long-distance pairings $\mathcal{L}$:

$$\mathcal{S} = \{(p_i, p_j)\, \epsilon \mathcal{A} \mid \|p_j - p_i\| < \delta_{max}\}$$

$$\mathcal{L} = \{(p_i, p_j)\, \epsilon \mathcal{A} \mid \|p_j - p_i\| > \delta_{min}\}$$

using as threshold distances $\delta_{max} = 9.75t$ and $\delta_{min} = 13.67t$ where $t$ is the keypoint's scale. The set of long-distance pairings is used to estimate keypoint orientation $\alpha$ by averaging local gradients $g\left(p_i, p_j\right)$ which in turn is computed using smoothed intensity values $I\left(p_i, \sigma_i\right)$ and $I\left(p_j, \sigma_j\right)$:

$$\alpha = arctan2\left(g_y, g_x\right)$$

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \tfrac{1}{L} \cdot \sum_{(p_i, p_j)} g\left(p_i, p_j\right)$$

$$g\left(p_i, p_j\right) = \left(p_j - p_i\right) \cdot \tfrac{I(p_j, \sigma_j) - I(p_i, \sigma_i)}{\|p_j - p_i\|^2}$$

The estimated keypoint orientation is employed for rotating the sampling pattern around the keypoint $k$. Finally the resulting pattern is applied to the keypoint and the bit-string $d_k$ is computed by performing all the short-distance intensity comparisons of point pairs $\left(p_i^\alpha, p_j^\alpha\right) \epsilon \mathcal{S}$ such that each bit $b$ corresponds to:

$$b = \begin{cases} 1, & I\left(p_j^\alpha, \sigma_j\right) > I\left(p_i^\alpha, \sigma_i\right) \\ 0, & otherwise \end{cases} \quad \forall \left(p_i^\alpha, p_j^\alpha\right) \epsilon \mathcal{S}$$

BRISK methodology has some remarkable advantages:

- It is more tolerant to image distortion and transformations, in particular to in-plane rotation and scale change, main weaknesses of the combination of BRIEF and FAST.
- Easily scalable for faster execution by using less sampling-points in the pattern at the expense of matching quality.
- Competitive to state-of-art methods while reducing the computational cost.
- Equivalent repeatability as the SURF detector under not too large image transformations.

However it also suffers from a number of disadvantages:

- BRISK detector as based on FAST is more sensitive to blur than blob-like detectors.

## D-BRIEF (Discriminative BRIEF)

Aiming at bridging the performance gap of decreased robustness of the binary descriptors in [32] the authors propose D-BRIEF a non rotation neither scale invariant compact descriptor based on a discriminative approach.

The descriptor is an N-dimensional string of concatenated bits $b_i$ computed by applying a set of projections $w_i$ to a real-valued vector $x$ made of intensities of an image patch and then thresholding by $\tau_i$ the results:

$$\forall i \epsilon 1, \ldots, N \quad b_i = sign\left(w_i^T x + \tau_i\right)$$

The task at hand is then how to optimize over $\{w_i, \tau_i\}$ in order to obtain an efficient and discriminative descriptor. The overall optimization goal is that of minimizing the expected Hamming distance between descriptors that describe similar keypoints while maximizing it for those that describe different keypoints. To this end starting from two sets of corresponding and not corresponding training image patches, $\mathcal{P}$ and $\mathcal{N}$, it is learned a set of discriminative orthogonal linear projections that map image patches to a more discriminative subspace and builds a bit-string by thresholding the result.

Since projecting image patches is computationally expensive and hurts the performance gained by the binary descriptors the projections $w_i$ are trained to be a linear combination of a few elements from a predefined set or dictionary $D$ which is designed to contain elements for which the responses $w_i = Ds_i$ can be computed fast.

The associated optimization function is:

$$\min_{\{(s_i,\tau_i)\}} \sum_{i \epsilon 1,\ldots,N} \left( \sum_{(x,x')\epsilon \mathcal{N}} sign\left((Ds_i)^T x + \tau_i\right) sign\left((Ds_i)^T x' + \tau_i\right) \right.$$
$$\left. - \sum_{(x,x')\epsilon \mathcal{P}} sign\left((Ds_i)^T x + \tau_i\right) sign\left((Ds_i)^T x' + \tau_i\right) + \lambda \left|s_i\right|_1 \right)$$

$$\text{subject to } (Ds_i)^T (Ds_j)^T = d_{ij}$$

In big words this function attempts to make the correlation of the binary codes as negative as possible for $\mathcal{N}$ and as positive as possible for $\mathcal{P}$, thus achieving more discriminative projections, while encouraging sparsity of the $s_i$ vectors, with $\lambda$ determining its sparsity level, and orthogonality of the projections by enforcing $d_{ij} = \begin{cases} 1 & i = j \\ 0 & otherwise \end{cases}$ 1.

Since the objective function is non-differentiable due to the sign function it is simplified to an equivalent one that gets rid of the calls to the *sign* function and is independent of the thresholds $\tau_i$:

$$\underset{\{s_i\}}{min}\sum_i \frac{\sum\limits_{(x,x')\epsilon\mathcal{P}}\left((Ds_i)^T(x-x')\right)^2}{\sum\limits_{(x,x')\epsilon\mathcal{N}}\left((Ds_i)^T(x-x')\right)^2} + \lambda\left|s_i\right|_1$$

$$\text{subject to } (Ds_i)^T\,(Ds_j) = d_{ij}$$

This new objective function is solved using Stochastic Gradient Descent, with soft-thresholding as the proximal operator of the $L1$ norm. Once an optimal value for the projections $w_i$ is found the original objective function is optimized only for the threshold values $\tau_i$ following a simple one-dimensional search. Being an heuristic the new objective function might get trapped into local minimum thus a proper initialization is necessary.

Proven by experimentation the authors vote for a stepwise initialization scheme of the optimization process rather than using random values:

1. An initial set of discriminant projections $\{w_i^0\}$ is obtained by minimizing the first term of the simplified objective function using Linear Discriminant Embedding, which includes already the orthogonality condition.

$$\{w_i^0\} = \underset{\{w_i\}}{argmin}\sum_i \frac{\sum\limits_{(x,x')\epsilon\mathcal{P}}\left(w_i^T(x-x')\right)^2}{\sum\limits_{(x,x')\epsilon\mathcal{N}}\left(w_i^T(x-x')\right)^2}$$

2. Use a technique for efficiently solving the convex optimization problem with sparsity-induced penalties result of addressing the sparsity constraint and approximating each projection $w_i^0$ with a sparse linear combination of elements from the dictionary $D$:

$$\{s_i^0\} = \underset{s_i}{argmin}\left\|w_i^0 - Ds_i\right\|_2^2 + \lambda\left|s_i\right|_1$$

   In practice the authors use LASSO (Least Absolute Selection and Shrinkage Operator) and control sparsity using the more convenient user defined $|s|_0^{max}$, the maximal number of non-zero coefficients in the representation.

By experimentation the authors determined that applying a global optimization on the initial set of projections $\left\{w_i^S\right\}$ does not lead to any significant improvement and hence it is used as the final estimated projections before optimizing the threshold values.

The dictionary $D$ is designed so that the dot product $D_i^T x$ can be computed efficiently. The authors tried three different types of dictionaries: box filters and Gaussian filters, both implemented as the result of a convolution, and rectangular filters implemented using integral images.

The best performances as reported by the authors are obtained when using the first 32 projections, beyond this performances start deteriorating. The resulting descriptor is then made of 32 bits. In addition using the dictionary of rectangular filters with 64 elements shows superior performance over other combinations of dictionary types and number of elements while still enabling real-time applications.

Bearing in mind this considerations the algorithm for computing D-BRIEF descriptor turns out to be:

1. Pre-define the dictionary to use.
2. Compute the initial set of projections $\{w_i^0\}$ using LASSO.
3. Optimize for the threshold values $\tau_i$.
4. Compute the descriptor by applying the projections to the vector representation of the image patch and thresholding the results.

This technique produces a very compact descriptor with enough discriminative power at the expense of data dependence, which is not a big problem as shown by the authors in a real-time application where projections and threshold where learned from a different dataset with significantly differing quality achieving still good recognition results. Its compactness places it in an advantageous position on resource restricted architectures since it enables to perform many operations in once single clock cycle. Compared with its direct competitors D-Brief outperforms BRIEF and BRISK in the task of matching providing an improvement from 11% to 32%, while requiring less memory, hence more accurate. It is as well reported to be 2-3 times faster to compute than BRISK and slightly more than BRIEF with a proper dictionary, hence more efficient.

# 4 Solution approach

While the BoF model as explained in sec. 1.3 is well defined for real-valued data streams, it requires a bit more effort to plug binary descriptors into it. The main concern lies in providing a clustering method able to deal with the nature of the binary descriptors and yet produce a vocabulary with sufficient descriptive power and a structure that minimizes loss due to quantization. Popular approaches consist in variations of the k-means algorithm using a distance measure other than Euclidean and a centers computation technique other than the arithmetic mean, some of this have been integrated into commercial applications such as Matlab[1]. In this chapter I discuss challenges related to the use of binary features as input to the BoF model and present my approaches to it, highlighting their advantages, describing their implementation details, and discussing related issues.

## 4.1 Binary BoF

The classical approach for vocabulary building following the BoF model consists in using Lloyd's algorithm as an heuristic for k-means clustering where Euclidean distance is used as measure of dissimilarity between observations, arithmetic mean for clusters centers computation, and overall intra-cluster variance as measure of clusters quality. This approach is well defined for real-valued data where Euclidean distance is meaningful and guarantees convergence at least heuristically[2].

In the case of binary features as those reviewed on the previous chapter, the direct application of Lloyd's k-means clustering algorithm is not possible due to the nature of the data and hence some considerations need to be taken. In concrete two things need to be revisited: distance measure and clusters centers computation method.

On the one hand using a dissimilarity function such as Euclidean distance as distance measure is a very natural and reasonable choice given that the task of clustering consist in grouping together observations sharing similar properties. Nevertheless when comes to evaluate it for vectors of binary features, it is not the metric of choice[8]. The reason is simple, using Euclidean distance to classify binary data often results in ties and arbitrary assignment to clusters because observations are always a bit part of very cluster, hence doing so becomes in unstable results which

[1]K-means clustering, MATLAB kmeans, MathWorks

[2]Cross Validated Q&A site - What to use, k-means or hierarchical clustering for presence absence data?

even in the long run provide no guarantee of convergence. The arithmetic mean is optimal only for the Euclidean distance, for Manhattan distance the median is said to be better, and for other metrics it is hard to say[3].

On the other hand using arithmetic mean as clusters centers computation method for binary data is not neither a reasonable choice. Produced clusters centers in this way are: 1) not binary anymore, 2) hardly representative of the data assigned to the clusters they represent, and 3) likely not to be sparse anymore. Among these only the second and third one concern us since in the BoF framework having interpretable clusters centers is a negligible property because what matters about them is to be enough representative as for describing a certain texturing pattern. Under Euclidean distance such centers are more similar to each other than the actual observations to each other and even less the observations to the center of the cluster their assigned to[4]. In summary, clusters obtained using arithmetic mean are not discriminative since all observations belong a bit to all clusters.

Although negligible, a popular alternative to produce clusters centers which preserve the binary nature of the data is the k-medoids clustering algorithm, commonly realized through the Partition Around Medoids (PAM) algorithm, which like k-means, attempts to minimize the distance between clusters centers and the observations assigned to the cluster it represents. Its main difference with respect to k-means, lies in the fact that instead of computing centroids (clusters centers using the arithmetic mean) it computes medoids (actual dataset instances) by using a matrix of distances between observations to heuristically choose as centers those observations which result in the lowest cost configuration. This algorithm is more robust to noise and outliers compared to k-means because minimizes a sum of pairwise dissimilarities, however holding a full matrix of distance implies an unaffordable increase in memory requirements, worsening the problem.

Other clustering algorithms similar to PAM are FANNY (Fuzzy Analysis) or CLARA (CLustering LARge Applications) which also compute medoids as clusters centers, but suffer from the same problem of needing to hold a huge distances matrix.

Regarding distance measures a popular one widely used for binary strings comparison is the Hamming distance, a dissimilarity measure grabbed from information theory which given two strings of equal length tells the number of positions at which the corresponding symbols are different. Such operation can be done extremely fast performing a XOR operation between the pair of bit strings followed by a bit count operation. This two instructions are implemented in the SSE4 (Streaming SIMD Extensions 4) instructions set often supported in modern CPUs. In particular the bit count is directly implemented as the POPCNT instruction from SSE4.2, only supported by the latest Intel Core i7 CPUs, which where unavailable can be replaced by an implementation based on SSE2/SSE4.1 instructions. While other

[3]IBM SPSS Statistics Technote 25479 - Clustering binary data with K-Means (should be avoided)

[4]Cross Validated Q&A Site - What to use, k-means or hierarchical clustering for presence absence data?

similarity/dissimilarity measures more suitable for dichotomous (binary) variables such as Sokal-Michener, Sokal-Sneath, Rogers-Tinimoto, Russell-Raova, Jaccard, Czekanowski or Sokal-Sneath can be used, they don't offer guarantee of convergence even heuristically.

## 4.2 Previous approaches

Typically proposed solutions to solve the problem described in the previous section involve some variation of the Lloyd's k-means clustering algorithm. This is the case of k-medians, often confused with k-medoids, where instead of using the arithmetic mean to compute clusters centers it is taken the dimension-wise median value. This modification results in an optimal scheme where it is minimized the error over all clusters with respect to the $L_1 - norm$ or Manhattan distance.

The k-medians algorithm is more reliable for discrete or binary datasets because the clusters centers are computed in each single dimension by operating over individual attributes of dataset instances. Centers computed in this way might conserve the binary nature of the dataset and are very unlikely to be dataset instances as the attributes come from different observations. However they might contain non-binary values since it is possible that there are clusters with an even number of observations. A straight-forward workaround to deal with ties is applying a threshold of 0.5 to medians resulting in a non-binary value, doing so would however result in an arbitrary selection of individual attributes of the cluster center.

Another alternative for clustering binary data is that proposed by [12] where binary data is transformed into some real-valued suitable format such that classical k-means is applicable. Though it helps to overcome the problem about the applicability of k-means, such transformation might decrease the descriptive power of the original features achieving low recognition results.

Hierarchical clustering methods such as AGNES (Agglomerative Nesting), DIANA (Divisive Analysis), and MONA (Monothetic Analysis), construct a hierarchy of clusters, with the number of clusters ranging from one to the number of observations. They constitute another reasonable alternative with the advantage that they also keep binary the clusters centers, unfortunately they suffer from the same problem of PAM about using a full distance matrix of the observations as clustering criteria.

An interesting proposal which I didn't explore is that of [19] where the author presents three variants of the k-means algorithm to cluster binary data streams using sufficient statistics and sparse distance computation.

Among the explored alternatives, k-majority is the most reasonable one because it constitutes a natural extension of k-means to the binary case. The majority voting step that replaces the arithmetic mean helps in keeping binary the clusters centers, and enables the use of Hamming distance which is not only meaningful for the binary case but also very efficient, providing a speedup of over 100 times with respect to floating point distance as reported by [8].

Regarding ties in the majority voting step, they are broken randomly for every dimension independently, and the authors claim that they didn't really happen more than a few times when working with hundreds of thousands of descriptors. This means that results achieved using k-majority are not just a matter of coincidence but due to the adopted strategy.

Finally since Hamming distance is meaningful in the binary case, it corresponds to using squared Euclidean distance when working with binary data. So if clusters centers are kept binary, what is true for k-means, is true also for k-majority. Proving one and for all the convergence of k-majority, at least heuristically.

In the following sections I will discuss the adaptations of two popular algorithms used for vocabulary construction, Hierarchical K-Means and Approximate K-Means.

## 4.3 Hierarchical K-Majority

It is an adaptation of the Hierarchical K-Means technique described in sec. 2.1.1 that enables using BoF model for binary data streams. Since the original technique doesn't care about minimizing the overall distortion within the final clusters but only within the clusters at each branch level of the tree, it is in theory possible at each level to use any suitable clustering algorithm.

The introduced modifications comprehend using Hamming distance for descriptors comparison instead of Euclidean distance and a majority voting step for clusters centers computation instead of the arithmetic mean, as described in sec. 2.3.4. The rest of the technique goes as usual, inverted index for holding database BoF vectors together with some weighting scheme. In Fig. 4.1 can be seen an example of a vocabulary built using hierarchical k-majority.

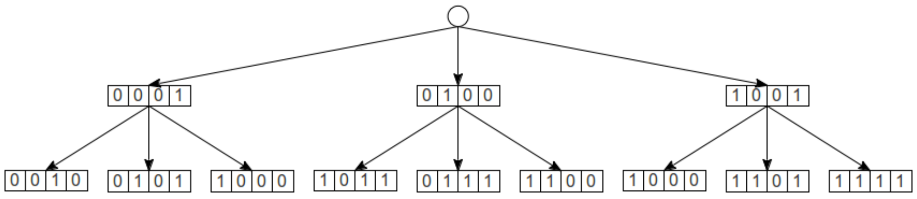


**Figure 4.1: Illustration of a vocabulary tree based on k-majority with branch factor 3 and depth 2.** Tree nodes are binary strings.

A remarkable advantage of this adaptation, is it's speed over classical Hierarchical K-Means which is already on its own a rapid solution for descriptors quantization; it suffer however from the same problem of loss due to quantization and other disadvantages that will be discussed in the next chapter. This solution is comparable to that described in [7], with the exception that instead of using a majority voting step for clusters centers computation it is computed the dimension-wise median and resulting non-binary values are truncated to zero. It is therefore not a very innovative contribution.

## 4.4 Approximate K-Majority

Inspired by the work of [20] where it is explored the possibility of using a flat vocabulary in order to minimize the overall distortion within final clusters lessening the chance of descriptors misassignment at quantization time, I propose Approximate K-Majority an adaptation of the K-Majority algorithm which uses an HCT (Hierarchical Clustering Trees) index to reduce the complexity of searching nearest neighbors at quantization time.

The adaptation is a simple extension of the K-Majority algorithm where it is utilized an HCT index instead of a linear index to find the right quantization cell on each case. The HCT index acts as quantizer and is built over the cluster centers at the beginning of each iteration, the last built HCT index is saved for future quantization.

Typical approaches for indexing binary descriptors are using a linear index or some hashing technique such as LSH (Local-sensitive hashing) or MinHash (min-wise independent permutations locality sensitive hashing). Here I opted for the HCT technique proposed in [16], due to its higher performance over LSH at higher search precision.

**Algorithm 4.1** Building one hierarchical clustering tree

```
Input: features dataset $D$
Output: hierarchical clustering tree
Parameters: branching factor $K$, maximum leaf size $S_L$
if size of $D < S_L$ then
    create leaf node with the points in $D$
else
    P ← select $K$ points at random from $D$
    C ← cluster the points in $D$ around nearest centers $P$
    for each cluster $C_i$ e $C$ do
        create non-leaf node with center $P_i$
        recursively apply the algorithm to the points in $C_i$
    end for
end if
```

The HCT technique consists in a suitable data structure used to narrow down the search space and a nearest neighbors search algorithm. The data structure is a set of multiple k-ary trees, named hierarchical clustering trees, result of a hierarchical decomposing the search space by recursively applying over the input dataset a clustering algorithm similar to k-medoids. Unlike k-medoids, HCT does not consider any cluster update step[5] and the recursion runs until the number of points on each cluster is below a certain threshold (the maximum leaf size). The complete process of building a single tree is summarized in Algorithm 4.1.

With a single hierarchical clustering tree it is possible to find the nearest neighbors to a given descriptor by propagating down the input data and linearly scanning descriptors on the correct domain. When the nearest neighbor to a query descriptor lies just across a boundary from the explored domain, the search needs to backtrack in order to reach the correct domain. Exploring multiple tree instances in parallel and combining the results helps to overcome this problem because as the trees are different enough, the closest neighbor likely lies in different domains in different trees, increasing the likelihood of quickly reaching the correct search domain.

**Algorithm 4.2** Searching parallel hierarchical clustering trees

```
Input: hierarchical clustering trees Ti, query point Q
Output: K nearest approximate neighbors of query point
Parameters: max number of points to examine $L_{max}$
$L \leftarrow 0$ {L = number of points searched}
$PQ \leftarrow$ empty priority queue
$R \leftarrow$ empty priority queue
for each tree $T_i$ do
    call $TraverseTree(T_i, PQ, R)$
end for
while $PQ$ not empty and $L < L_{max}$ do
    $N \leftarrow$top of $PQ$
    call $TraverseTree(N, PQ, R)$
end while
return $K$ top points from $R$
procedure $TraverseTree(N, PQ, R)$
if node $N$ is a leaf node then
    search all the points in $N$ and add them to $R$
    $L \leftarrow L + |N|$
else
    $C \leftarrow$ child nodes of $N$
    $C_q \leftarrow$ closest node of $C$ to query $Q$
    $C_p \leftarrow C \setminus C_q$
    add all nodes in $C_p$ to $PQ$
    call $TraverseTree(C_q, PQ, R)$
end if
```

The process of nearest neighbors search using multiple hierarchical clustering trees

[5]It is as if the cluster centers would be randomly selected.

is presented in Algorithm 4.2. It starts by individually traversing all the trees in order to find the nearest neighbor on each case while adding unexplored nodes to a priority queue. Once done, the search is continued by resuming from the next node in the priority queue that is closest to the query descriptor. The search ends when a certain number of points, given as a parameter to the search algorithm, has been examined.

Two remarkable properties of the HCT technique are that it scales well with the size of the dataset (millions or even tens of millions of descriptors), and that it is able to operate at different search precisions by tuning a set of parameters.

# 5 Experimental results

In this chapter I concentrate on evaluating the retrieval performance of the clustering approaches described in the previous chapter over a large scale database of landmark images exhibiting many changes in visual appearance. In particular I look at their capabilities of retrieving relevant images inside the shortlist of top candidates which is further refined using as criterion the features spatial distribution.

Given the cost in large scale databases of accessing $n$ random places at disk corresponding to the short list of top $n$ ranked images, the retrieval stage of the pipeline must roughly retrieve first the most visually similar images. This moves the focus on evaluating the BoF retrieval, since the overall performance depends largely on it.

By undertaking the evaluation of this algorithms I intend to respond questions such as, how does varying pipeline modules affects system behavior? how does vocabulary size influences retrieval performance? and what is the effect of tuning the different algorithms parameters over retrieval performance?

To address such questions, in the following sections I will explain the environment setup for the execution of the experiments -datasets and pipelines-, then I will introduce the measures used to evaluate the performance and the evaluation procedure. Finally I will briefly explain what do the performed experiments consist, and I will analyze the obtained results based on the theoretical framework exposed in previous chapters.

## 5.1 Experimental setup

### 5.1.1 Datasets used

Benchmark datasets used for landmark and location recognition typically originate from one of two sources: images crawled from a photo-sharing sites such as Flickr or an acquisition process of panoramic images using a mobile mapping vehicle equipped with a number of sensors among which a panoramic camera and high-definition cameras. In order to evaluate retrieval performance and individually assess the effects of the pipelines, the evaluation is done over three datasets classified on the first category: Oxford5K, Paris6K and Rome10K. The first one is used to run the main experiments, the second and third ones are used to test generalization capabilities of a visual vocabulary built from binary descriptors.

**Oxford5K.** Introduced in [20], the Oxford buildings dataset[1] is one of the most popular benchmark datasets for landmark recognition. It consists of 5.062 images collected from Flickr depicting 11 different landmarks from Oxford city, each represented by 5 possible text queries resulting in 55 query images and 5.007 training images. A sample of query images is shown in Fig. 5.1.

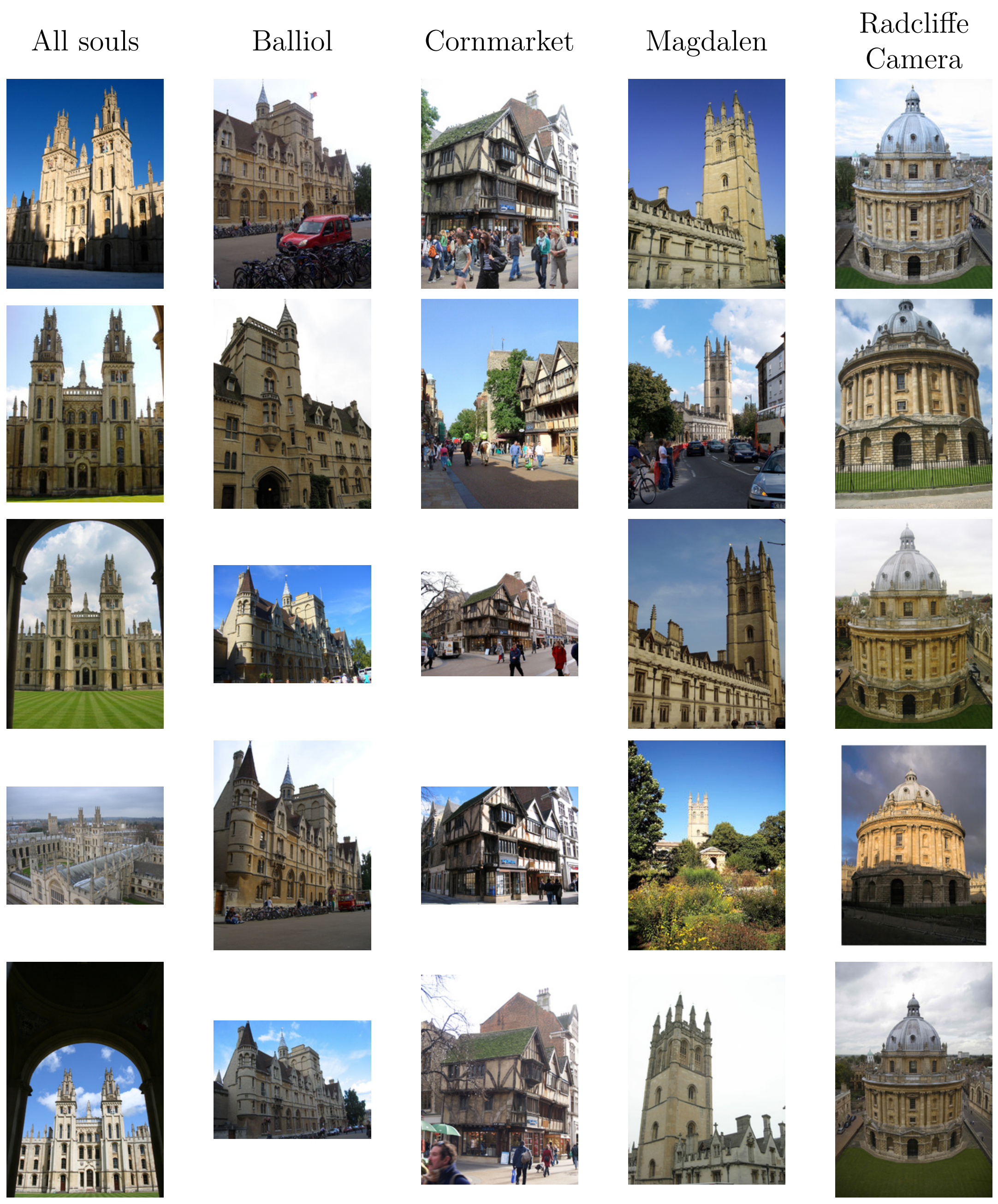


**Figure 5.1:** Sample query images from the Oxford5K dataset

[1]Oxford5K dataset is publicly available from http://www.robots.ox.ac.uk/∼vgg/data/oxbuildings/

All the dataset images were originated from the same acquisition process (camera setup, lighting conditions, shooting time) resulting though in a quite unrealistic testing scenario. Together with the images and the ground truth data containing images quality labels (*good*, *ok*, *junk*, *bad*) and landmark information, the authors published in binary format a set of SIFT descriptors extracted from detected MSER regions, as well as a visual vocabulary used for their own tests.

In computing the average precision over this dataset, the *good* and *ok* images are used as relevant documents, *bad* images as non-relevant documents, and *junk* images as null examples which are treated as if they were not present in the database and hence the score is unaffected whether they are retrieved or not.

The images in this corpus exhibit substantial differences in viewpoint, scale, lighting, perspective distortion, and occlusion. Additionally some images might represent more than one landmark at the same time, rendering the visual vocabulary building and retrieval processes much more complex since visual structures present in an image might also be present in another image corresponding to a different landmark. In the other hand some images do not depict any landmark and hence act just as distractors. Such characteristics make this dataset ideal to evaluate retrieval capabilities of a landmark recognition system, but rather not to use it as reference for a location recognition system.

**Paris6K.** Presented by first time in [21], the Paris buildings dataset[2] was introduced to examine the generalization of their system when the images used for building the visual vocabulary differ from those used to create the database. It consists of 6.412 high-resolution (1024×768) images depicting 12 different landmarks from the city of Paris. All of them were obtained from Flickr in the same manner than in Oxford5K. Example images from this dataset are shown in Fig. 5.2. Together with the dataset the authors provided ground truth data formatted in the same manner than Oxford5K.

[2]Paris6K dataset is publicly available from http://www.robots.ox.ac.uk/∼vgg/data/parisbuildings/

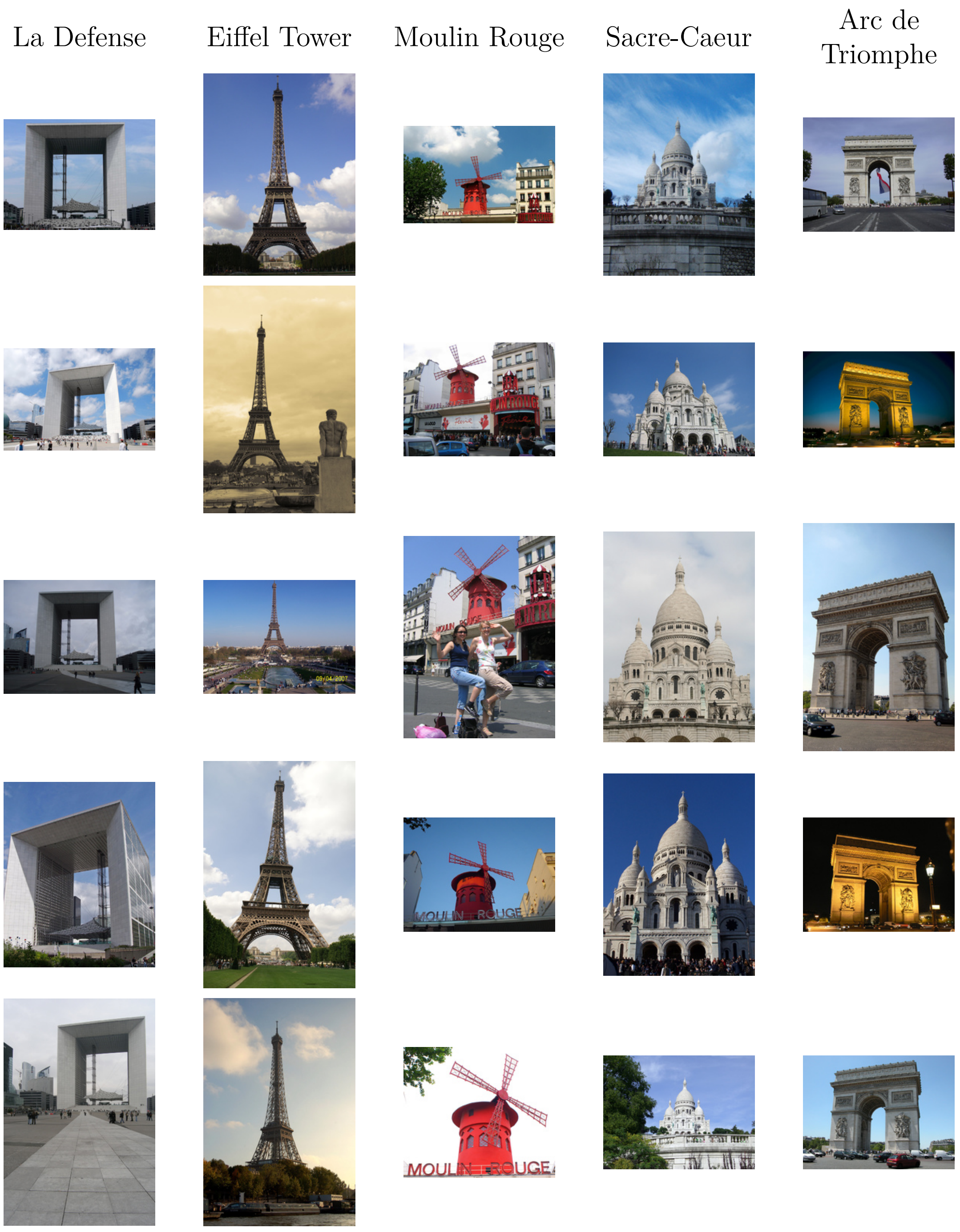


**Figure 5.2:** Sample query images from the Paris6K dataset

**Rome10K.** It makes part of an unpublished work in the topic of binary descriptors compression. It consist of 10 videos depicting 10 different landmarks from the city of Rome and 10.000 low-resolution (500×300) images among which 84 are relevant to the videos since they contain the landmarks depicted on them, the rest act simply as distractors. The relevant images were crawled from Google Images whereas the distracting ones were randomly sampled from the MIRFLICKR image collection. This means that they were taken under heterogeneous imaging conditions and hence constitutes a challenging dataset. Example images from this dataset are shown in Fig. 5.3.

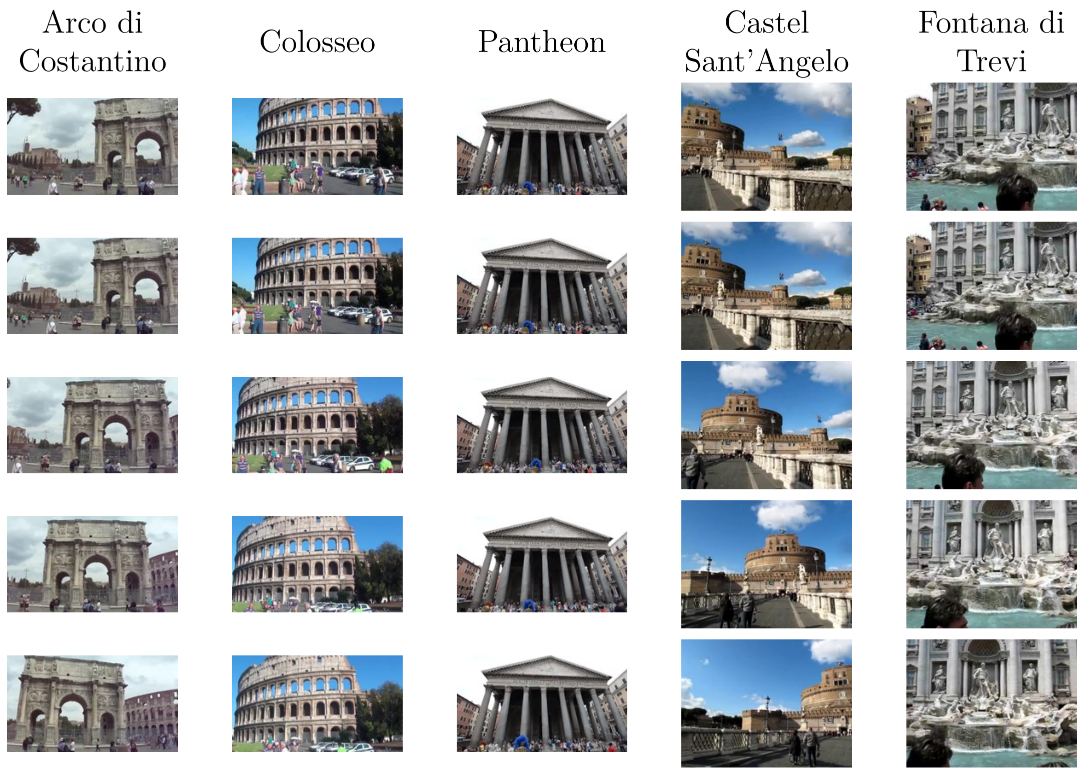


**Figure 5.3: Sample photograms of the query videos from the Rome10K dataset.** To use this dataset as input to my implementation I extracted photograms at each second from the query videos and used them as query images.

To the sake of the comparison process and to keep a controlled environment I extracted my own descriptors using in all cases the same set of keypoints to render meaningful the comparison process. In Tab. 5.1 are shown the counts of extracted features for the Oxford5K dataset.

### 5.1.2 Tested pipelines

To evaluate the clustering methods presented in the last chapter, I plug them into the standard landmark recognition pipeline and compare them against a reference and a baseline pipeline. In the context of landmark recognition using the BoF model, a pipeline is understood as the combination of a features detector, a features descriptor, and some approach for visual vocabulary construction.

| Detector | Descriptor | No. of features | Size in Bytes |
|---|---|---|---|
| HARRIS | BRIEF | 3'848.354 | 128 MB |
| HARRIS | BRISK | 4'019.848 | 257 MB |
| DoG | BRIEF | 16'348.741 | 510 MB |
| DoG | SIFT | 17'814.954 | 8.6 GB |
| HESSIAN-AFFINE | SIFT | 12'532.047 | 6.0 GB |
| AGAST | BRIEF | 84'580.664 | 2.6 GB |
| AGAST | BRISK | 89'462.894 | 5.4 GB |
| BRISK | BRIEF | 7'354.484 | 235 MB |
| BRISK | BRISK | 7'515.451 | 469 MB |

**Table 5.1: Number of descriptors extracted from Oxford5K datasets.** It were used different pairings of detectors and descriptors.

The *reference pipeline* represents the state-of-art in landmark recognition using only BoF retrieval and places the highest mark in retrieval performance. It is a reproduction of the BoF part of the system used in [20] as a reference to test their proposal. It uses a Hessian-affine feature detector paired with SIFT for feature description. For images retrieval it uses the BoF model fed by a 1M visual words vocabulary built from all the extracted descriptors using Hierarchical K-Means, together with a linear classifier supported by an inverted index. An interesting proposal for features spatial layout verification is to use Locally Optimized RANSAC for estimating three different perspective transformation models: 3 dof, 4 dof, and 5 dof, in each case it is used a general (6 dof) affine transformation for the iterative step. Such procedure is performed only for the top 1.000 images from the shortlist of candidates.

The *baseline pipeline* is a reduced version of the reference pipeline. For keypoint detection it uses DoG instead of an affine detector. Such change is motivated by the fact that using a Hessian-Affine detector, although very effective to deal with perspective distortion effects, is unaffordable for mobile landmark recognition. For image retrieval it follows the same strategy than the reference pipeline, and for features spatial layout verification it uses RANSAC for estimating an homography between each query and database images in the shortlist of retrieved candidates.

The *tailored pipeline* for using binary features in a BoF model uses for images description a combination of the detectors and descriptors presented in chapter 3. For vocabulary construction it uses either Hierarchical K-Majority or Approximate K-Majority, while for geometric verification uses RANSAC for estimating an homography.

## 5.2 Performance evaluation framework

### 5.2.1 Performance measures

As said in sec. 1.4, landmark recognition is basically a binary classification problem where images in a database are classified with respect to a given query image depending on whether it represents the same landmark or not. To measure the quality of such classifier I employ two popular and widely accepted measures, *precision*, defined as the ratio of retrieved and relevant images to the total number of retrieved, and *recall*, the ratio of the number of retrieved and relevant images to the total number of relevant images in the corpus. In the case of landmark recognition, the relevant images with respect to a given query image are those representing the same landmark.

To characterize the classifier's performance with respect to a single query, I look at the *precision-recall* curve, result of varying the threshold of the number of retrieved images, and the *average precision*, a useful measure for information retrieval systems that return a ranked sequence of documents, defined as the precision value averaged across all values of recall between 0 and 1, i.e. the area under the precision-recall curve:

$$AvgP = \int_0^1 p\left(r\right) dr$$

In practice this integral is approximated by a finite sum over all precision values at every possible threshold value, multiplied by the change in recall:

$$\sum_{k=1}^{N} P\left(k\right) \Delta r\left(k\right)$$

As done in [20], in order to reduce the impact of wiggles in the curve, instead of the precision value at a certain recall level, here I take the average between the current and the previous precision values:

$$\sum_{k=1}^{N} \left(\frac{P\left(k\right) + P\left(k-1\right)}{2}\right) \Delta r\left(k\right)$$

To evaluate the overall system performance and not only with respect to a single query, I use the *mean Average Precision (mAP)* score, a global measure result of computing the arithmetic mean of the average precision values of a set of queries. To complement the *mAP* score, I use a single *precision-recall* curve of the averaged retrieval results of all queries.

### 5.2.2 Evaluation methodology

The plan consists in executing a series of experiments carefully prepared to respond the questions posed at the beginning of the chapter. Each of them is run independently, thus their incomes and outcomes are not correlated. A single experiment consist in the comparison of one or more pipeline instances, each of which is determined by a set of algorithms parameters which vary on each case, see Tab. 5.5, Tab. 5.6 and Tab. 5.8 for a full description of the parameters of the employed algorithms. Running a single pipeline consist in three steps: 1) build the visual vocabulary, 2) create the database (Tab. 5.2 shows the relative size of the utilized databases), 3) perform BoF retrieval, and 4) optionally perform spatial verification.

To render fairer the comparison process between the different pipelines instances, identical cluster initialization is used, i.e. no random seeding. To make it more realistic, the datasets are divided between *training* and *query images*, the former group is used to build the vocabulary and to create the database, and the second one is used to test the pipeline. The training set is further divided between *distracting* and *non-distracting images*, the first is excluded from the database since in the frame of the landmark recognition problem it's senseless to include images not depicting any landmark.

| Dataset | No. of images | Description |
|---|---|---|
| Oxford5K | 790 | All images except the queries and the distractors |
| | 5.007 | All images except the queries |
| | 5.062 | All images |
| Paris6K | 3.399 | All images except the queries and the distractors |
| | 6.337 | All images except the queries |
| | 6.392 | All images |
| Rome10K | 84 | All images except the queries and the distractors |
| | 10.000 | All images except the queries |
| | 10.084 | All images |

**Table 5.2:** Number images per database for each dataset

## 5.3 Results and discussion

I performed a total of four experiments, the first one aimed at evaluating BoF retrieval while the latest three evaluated other aspects of the tailored pipeline. Firstly, I tried the tailored pipeline using both Hierarchical K-Majority and Approximate K-Majority with several combinations of parameters, and compared them in order to: 1) find the best scheme to address binary BoF, and 2) discover which parameters are more sensitive. Secondly, I run the optional geometric verification step over the ranked candidates produced by the best scheme found in the first experiment. Thirdly, I tried different combinations of feature detectors and descriptors to investigate how does their invariance capabilities to different changes in visual appearance influence retrieval performance. To conclude the experiments, I checked the generalizing capabilities of a visual vocabulary trained with binary descriptors. In all cases for storing the database images BoF vectors I used an inverted index with tf-idf word weighting, L1-norm for normalization and L1-distance for comparison. All the experiments, unless stated, were run over the Oxford5K dataset and always using my own implementation of the described algorithms.

### 5.3.1 Reproducing the state-of-art

Prior to experiments execution I proved my implementation of the *reference pipeline* and compared it with the analog results achieved in [20]. There the authors trained the visual vocabulary using all dataset descriptors, then created a database and tested it using the descriptors corresponding to the query regions, i.e. landmarks on the query images. Doing so they achieved a mAP score of 0.43 and improved it up to 0.469 after geometric verification. In this section I address only BoF retrieval while the evaluation of geometric verification effects is done on a later section.

For feature detection and extraction I used a pre-built binary[3] implementing a modified version of the Hessian-Affine detector proposed by Krystian Mikolajczyk, then I formatted the extracted descriptors and fed them to my implementation. Since it was not clear which combination of training set and database did the authors use, I tried several combinations.

Tab. 5.3 shows the retrieval results using the full query images rather than the clipped regions. At glance it can be seen that the unrealistic scenario where all the descriptors of the dataset are used to train the vocabulary, presents in all cases the highest retrieval performance. This is because the descriptors of the query images bias the vocabulary construction towards one including words which better describe, not only, the database images but also the query images. Likewise performance increases when the query images are included in the database.

It can be seen as well that including distractor images in the database hurts retrieval performance in a range of 10% to 20%, this is a reasonable outcome since such distractors are difficult ones and may easily be confused with those used in the query set.

Regarding the size of the training set (see Tab. 5.4) we notice that a higher value results in all cases in an increased retrieval performance, moreover using more descriptors than visual words increases the quality of the vocabulary. In other words the ratio of the number of descriptors to the number of words benefits retrieval performance.

[3]A Linux binary of the Hessian-affine detector with SIFT descriptor is publicly available at http://cmp.felk.cvut.cz/∼perdom1/code/index.html

| Training set | Database | mAP | mAP* |
|---|---|---|---|
| All images except the queries and the distractors | All images except the queries and the distractors | 0.4121 | - |
| All images except the queries | All images except the queries and the distractors | 0.4335 | 0.4174 |
| All images | All images except the queries and the distractors | 0.4406 | - |
| | | | |
| All images except the queries and the distractors | All images except the queries | 0.2203 | - |
| All images except the queries | All images except the queries | 0.3384 | 0.3127 |
| All images | All images except the queries | 0.3480 | |
| | | | |
| All images except the queries and the distractors | All images | 0.5795 | - |
| All images except the queries | All images | 0.5894 | 0.5514 |
| All images | All images | 0.6135 | - |

**Table 5.3: Retrieval performance of the reference pipeline combining different training sets, databases, and queries.** All scenarios use Hierarchical K-Means with parameters: depth=6, branch facto$r$=10, max training cycles=10, seeding=RANDOM, fed with SIFT descriptors extracted from Hessian-Affine regions. The first mAP scores column corresponds to evaluate the system using the full query images, while the second column corresponds to evaluate the system using only the query regions.

To achieve a faithful reproduction of the test environment I performed the same exercise than before, but using query regions. Since the mAP scores are consistent among the different vocabularies I chose to use the one corresponding to the training set excluding the queries. As shown in Tab. 5.3 using query regions to test the pipeline decreases retrieval performance, this is probably because descriptors corresponding to surroundings help recognition rather than distract as mentioned in [2].

Considering the results over the *reference pipeline* for the following experiments the vocabulary is built using as training set the descriptors corresponding to all images except the queries and as database all the images excluding the queries and the distractors.

### 5.3.2 Find best performer set of vocabulary parameters

In this experiment, I explore the effect over the retrieval performance in the tailored pipeline when varying the parameters of the clustering algorithms proposed in the

| **Training set** | **No. of descriptors** | **No. of descriptors / No. of words** |
|---|---|---|
| All images except the queries and the distractors | 2’055.919 | 2,056 |
| All images except the queries | 12’384.049 | 12,384 |
| All images | 12’532.047 | 12,532 |

**Table 5.4: Size of the training sets in terms of number descriptors and ratio between the number of descriptors and the number of visual words.** Features were extracted using SIFT from affine regions detected over images from the Oxford5K. The ratios were computed assuming a vocabulary with 1M words.

previous chapter. During the execution of this experiment, despite the set of parameters producing the best retrieval performance, I expect to find the crucial and not crucial parameters, i.e. the ones towards which the result is more sensitive. Such procedure is of course heuristic because of the infeasibility of testing all possible combinations of parameters.

### Hierarchical K-Majority

The behavior of the Hierarchical K-Majority clustering algorithm is influenced by four parameters: depth, branching, training cycles and clusters seeding (see Tab. 5.5 for a full description). I start the assessment with the standard set of Hierarchical K-Means parameters, depth 6, branch factor 10, maximum 10 training cycles and random clusters seeding.

| **Parameter** | **Description** |
|---|---|
| Depth | Zero-based index indicating the depth of the tree starting at the root. |
| Branch factor | Number of children of each node. |
| Maximum number of iterations | Maximum limit in the number of iterations in a single run. |
| Seeding algorithm | Algorithm used for cluster centers initialization, one among random, k-means++, or Gonzalez algorithm. |

**Table 5.5:** Hierarchical K-Majority parameters

Figure 5.4a shows how the algorithm’s performance depends on the size of the vocabulary. Although there is no optimum number of words due to the trade-off between distinctiveness and repeatability mentioned in sec. 1.3, it is observed that generally performance increases with the number of words, and particularly with the depth but not much with the branch factor. The drop in performance after 10M words can be explained by the empirically learned rule that the ratio between the number of training descriptors and the number of words should be kept high.

The shape of the vocabulary has also a positive impact on the retrieval performance but in a much lesser extent, as shown in Figure 5.4b. Although both vocabularies have the same number of words, the flatter one reports a slightly higher performance. Such behavior is consistent up to a 3% of recall corresponding to the top 4 candidates and a precision level of at least 40 %. A reasonable explanation behind this result is that bigger leaf nodes increase repeatability while the hierarchical structure acts as a sort of nearest neighbor index. This result partially motivates using a flat vocabulary instead of a hierarchical one.

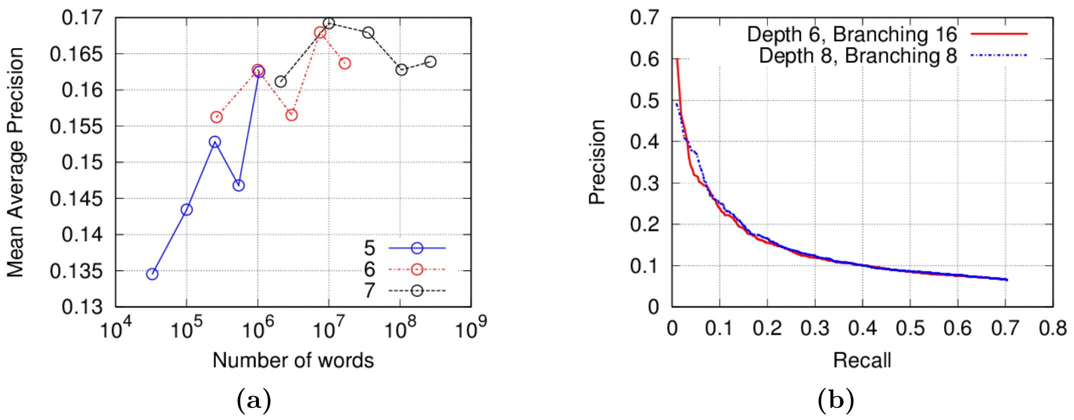


**Figure 5.4: Effect of varying size and shape of a vocabulary built using Hierarchical K-Majority.** (a) Result of increasing the number of words varying the branch factor from 8 to 16 and the depth from 5 to 7. (b) Averaged precision-recall curves of two schemes producing 16M visual words. The solid red line corresponds to a vocabulary with depth 8 and branch factor 8, it achieves a mAP score of 0.1609, while the dash-dotted blue line corresponds to a vocabulary with depth 6 and branch factor 16 which achieves mAP score of 0.1634.

The next examined parameter is the maximum number of training cycles. From Fig. 5.5 it can be observed that a maximum of 30 iterations performs better than a smaller or larger value. K-Majority is known to solve ties randomly and it might end up in unstable results. Running few iterations might not be enough to achieve clusters of good quality, while running more than 30 iterations might over do the vocabulary and hence hurt performance.

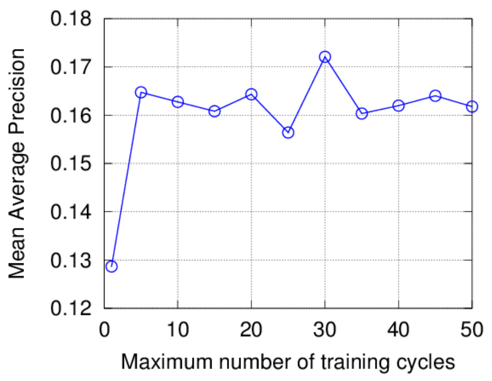


**Figure 5.5: Effects of the unsupervised Hierarchical K-Majority vocabulary training on retrieval performance.** Result of increasing the maximum number of training cycles.

The last parameter I examine is the seeding algorithm. Fig. 5.6 shows how does the algorithm behaves when changing the seeding algorithm. Among all *Gonzalez* presents the worst overall performance while *K-Means++* slightly outperforms *Random* up to a recall level of 3 % after which their performance becomes pretty similar.

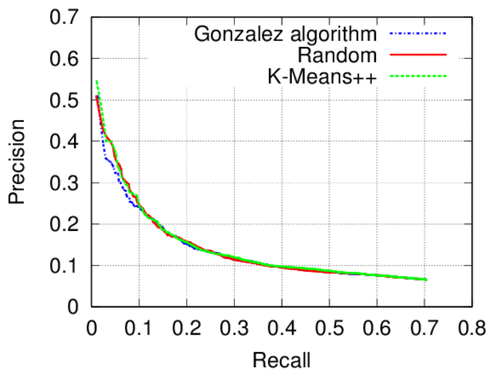


**Figure 5.6: Effect of using different seeding algorithms to bootstrap the vocabulary training process.** Averaged precision-recall curves result of changing the seeding algorithm used to build a 1M visual words vocabulary. The corresponding mAP scores are 0.1652 for K-Means++, 0.1624 for Random, and 0.1591 for Gonzalez algorithm.

Both *Gonzalez* and *K-Means++* are greedy algorithms based on the idea of spreading out the initial clusters, although they operate similarly, they have a fundamental difference in the way of choosing new centers. In the *Gonzalez* algorithm the first cluster center is chosen randomly then iteratively it adds in the farthest point from the ones chosen so far, until $k$ centers have been chosen. *K-Means++* operates in the same manner but it randomly choses as new center a data point among the remaining ones, using a weighted probability distribution where a point $x$ is chosen with probability proportional to $D(x)^2$.

Introducing some randomization in the centers selection process helps to increase retrieval performance, possibly because it helps breaking the curse of dimensionality[4]. This would explain why both *Random* and *K-Means++* achieve a similar performance though the second one theoretically provides centers of better quality. It's worth noticing that *K-Means++* has a bigger cost than *Random* in terms of memory and execution time, specially in cases of a big volume of data.

Fig. 5.7 shows the averaged precision-recall curves of the reference and baseline pipelines in comparison with those of the tailored pipeline using as clustering approach the two best Hierarchical K-Majority schemes found during parameter assessment. The gap in performance between the reference and baseline corresponds to a difference in mAP of around 10%, it obeys to the change in the feature detector and supports the idea of having a baseline which doesn't use affine keypoints to make more fair the comparison process against the tailored pipeline.

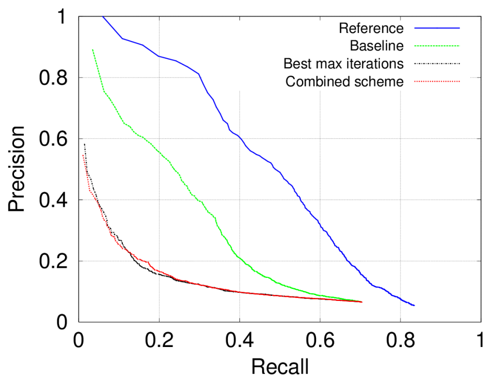


**Figure 5.7: Comparison between the best Hierarchical K-Majority schemes and the reference and baseline pipelines.** The red dotted line corresponds to the scheme combining the parameters which individually best perform., i.e. a 10M visual words vocabulary built with depth 7 and branch factor 10 trained with a maximum of 30 cycles and K-Means++ seeding. The black dash-dotted line corresponds to the best performer scheme found during the test of maximum number of training cycles, i.e a 1M visual words vocabulary with depth 6 and branch factor 10 trained with a maximum of 30 cycles and Random seeding. The corresponding mAP scores are 0.1678 for the combined scheme, 0.1721 for the best max iterations scheme, 0.4335 for the reference pipeline, and 0.3546 for the baseline pipeline.

Looking at the schemes using binary features it is evident that both exhibit a much lower retrieval performance than any of the schemes using SIFT descriptors. Some

[4]The interested reader might search for “Using Randomization to Break the Curse of Dimensionality” by John Rust

relevant reasons to explain this result are:

- **Noisy descriptors:** in Hierarchical K-Majority like in Hierarchical K-Means the variability and noise in the descriptor vectors is not a big issue since it helps moving them between different quantization cells and partially address the loss due to quantization.
- **Loss due to quantization:** it's one of the main reasons why the bag-of-words model often fails, nevertheless it affects equally all the tested pipelines and hence doesn't pose a challenge for the purpose of this assessment.
- **Low discriminative power:** although BRIEF is more efficient to compute than SIFT, it encodes less information and hence it has a lower discriminative power.
- **Influence of the feature detector:** in [4] the authors conclude that using BRIEF in conjunction with SURF negates part of its considerable speed advantage while pairing it with some fast detector like CenSurE improves descriptor's matching performance. In [18] the authors go beyond and claim that the single most important factor governing image's matching performance using local features is the number of patches sampled from the training images and interest operators such as those used by SURF or SIFT detectors cannot provide enough patches. This might indicate that the reason why the scheme combining DoG with BRIEF has so low retrieval performance, and pairing it with a more adequate detector might lead to some improvement.

Between both schemes using binary features the combined one shows the lowest performance, indicating that the best individually performing parameters don't fit very well altogether. The parameters of both schemes are the same except for the *tree depth* and the *seeding algorithm*, it was already mentioned that a higher depth increases retrieval performance, but how about the seeding algorithm? Using *Random* seeding the required number of training cycles to achieve a stable result is higher while using *K-Means++* it is required a smaller number of training cycles to achieve a result of good quality, this indicates that the *seeding algorithm* should be carefully chosen with respect to the maximum number of *training cycles*.

The above experiments show that the best parameters to choose depend on the size and characteristics of the dataset, however for the future experiments I consider that the best scheme uses the following parameters: depth 7, branch factor 10, maximum 30 training cycles and random seeding.

### Approximate K-Majority

The behavior of this algorithm it's influenced by the following parameters: the number of clusters, the maximum number of training cycles, the seeding algorithm, and the Hierarchical Clustering Trees parameters (see Tab. 5.6 for a full description).

| Parameter | Description |
|---|---|
| Number of words | Final number of clusters or quantization cells. |
| Maximum number of iterations | Upper limit in the number of iterations. |
| Seeding algorithm | Algorithm used for cluster centers initialization, one among random, k-means++, or Gonzalez algorithm. |
| HCT - Number of trees | Number of trees built. |
| HCT - Branch factor | Number of children of each node of a tree. |
| HCT - Maximum leaf size | Maximum number of points to examine before stopping search algorithm. |
| HCT - Number of checks | Maximum number of leaf nodes to visit. |

**Table 5.6:** Approximate K-Majority parameters

Starting from the findings in [16], I addressed the first test which consisted in finding a fair set of the HCT algorithm's parameters. Particularly I concentrated in the number of trees, the branch factor, and the maximum leaf size.

According to the authors the optimum number of trees depends on the desired search precision[5], for less than 70% two parallel trees is enough and for precisions above 85% a higher number of trees between 4 and 8 gives better results. They notice however that in practice the optimum number of trees depends also on the available memory and constraints on the tree build time. For the problem at hand the main concern is having a low quantization time thus using two trees in parallel is enough.

The authors also claim that higher branch factors perform better for search precisions above 80% with little gain for values above 16, conversely for lower search precisions very large branching factors perform worse and it is associated a higher tree build time.

As for the maximum leaf size the authors argue that a value of 150 performs better than a smaller values, which result in deeper trees with small leafs, or a larger ones, which result in flatter and more superficial ones with large leaves. The authors explain such result by the fact that for small leaf sizes the overhead of traversing the tree to examine more leaves is greater than the cost of comparing the query feature to all the features in a larger leaf, while for very large leaf sizes the cost of linearly examining all the features in the leaves ends up being greater than the cost of traversing the tree.

In the basis of these findings I came out with a set of combinations of HCT algorithm's parameters, they are shown in Tab. 5.7. As the tests with the different combinations were being executed I noticed that the construction of the vocabularies was requiring too much time.

Investigating the root cause of the high time required for vocabularies construc-

[5]Percentage of exact neighbors found in the total neighbors returned

tion, I found that the time required to quantize a single BRIEF descriptor using a vocabulary built with Approximate K-Majority[6] was around 0.1 ms in comparison with 0.01 ms required by a vocabulary built with Hierarchical K-Majority[7]. This means that using Approximate K-means to quantize a typical query image which has an average of 3.000 BRIEF descriptors would require 0.3 seconds. With a few descriptors this might be a negligible number, but for a higher number of descriptors it turns into a main issue specially because in landmark and location recognition having low retrieval times is a main concern.

| Number of trees | Branch factor | Maximum leaf size | Avg index building time | Vocabulary build time | mAP |
|---|---|---|---|---|---|
| 2 | 8 | 100 | 6 sec | 22 hours | 0.1565 |
| 2 | 8 | 150 | 6 sec | 20 hours | 0.1580 |
| 2 | 16 | 100 | 9 sec | 1 day 23 hours | 0.1561 |
| 2 | 16 | 150 | 9 sec | 1 day 14 hours | 0.1580 |
| 2 | 32 | 100 | 14 sec | ~3 days | 0.1632 |
| 2 | 32 | 150 | 13 sec | ~3 days | 0.1599 |

**Table 5.7:** Effect over retrieval performance when varying Approximate K-Majority ANN method parameters

Although building the HCT indices took few time, building the full vocabularies required too much time because repeatedly searching for nearest neighbors within a big quantity of descriptors its an expensive task.

Finally, indexing a large amount of data using an HCT index requires that used trees are neither too deep, since there is the risk that it becomes very expensive traversing them, nor too flat, or it could end up in a unaffordable linear search. To build such trees it is necessary to find a fair combination of branch factor and maximum leaf size because they control the flatness and depth of the trees. Nevertheless in practice such combination is infeasible, possibly changing the termination criterion of the trees' construction process to one that allows to control their shape could improve the results.

Since there is no set of parameters which provides low quantization times even sacrificing search precision, I concluded that Hierarchical Clustering Trees algorithm is not suitable to address nearest neighbor search in the context of the K-Majority algorithm and thus I didn't considered it anymore for future tests.

[6]The vocabulary was trained with 1M clusters, maximum 10 iterations, random clusters seeding, 2 parallel Hierarchical Clustering Trees with branch factor 8 and a maximum leaf size of 150.

[7]The vocabulary was trained with depth 6, branch factor 10, maximum 10 training cycles and random clusters seeding.

### 5.3.3 Geometric consistency check with binary features

In this experiment I compare the improvement achieved by running the optional geometric consistency check over the tailored and reference pipelines, using Hierarchical K-Majority best scheme as clustering approach for the former one. The objective is analyzing how the different parameters influence the algorithm behavior (see Tab. 5.8 for a full description of the parameters) and find a reasonable set of values that helps to increase retrieval precision.

| **Parameter** | **Description** |
|---|---|
| Top keypoints | Number of higher response keypoints to keep. The filtered in key-points are used to generate the putative matches used as input for the Homography estimation process. |
| SIFT ratio threshold | Ratio of the distances of the first and second nearest neighbors descriptors of the candidate image to the descriptors of the query image. Applying this ratio helps to increase descriptor's matching robustness by discarding false matches. |
| Distance threshold | Analog to the SIFT ratio threshold but for the case of binary features. |
| Minimum matches | Minimum number of matches required to estimate an Homography. |
| RANSAC re-projection threshold | Inlier threshold value used by the RANSAC procedure. The parameter value is the maximum allowed distance between the observed and computed point projections to consider it an inlier |

**Table 5.8:** Geometric consistency check parameters

The first parameter I explore is the number of top keypoints, Fig. 5.8 shows how does the geometric verification algorithm behaves when varying the number of filtered in keypoints. Using only the keypoints with higher response, i.e. a small number of top keypoints, might filter out some true positives which are very descriptive while including false positives such as keypoints result of shadow casting. In the reference pipeline using more keypoints increases performance but hurts retrieval time since more features need to be considered during Homography estimation. On the other hand in the tailored pipeline beyond some limit using more keypoints doesn't benefit retrieval performance, this happens because in that case the set of putative matches are low quality and hence constraining RANSAC to work on top of the higher response keypoints doesn't help to provide a better quality model.

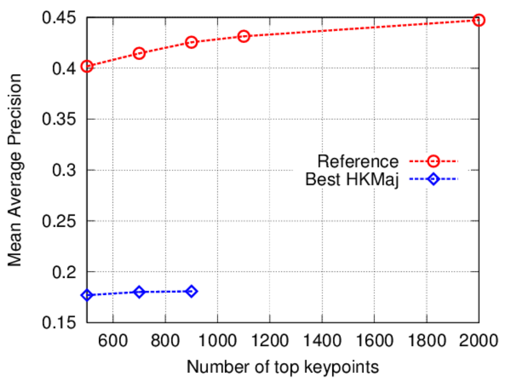


**Figure 5.8: Effect over retrieval performance of varying the top number of keypoints.**

The next parameter I investigate is the SIFT ratio threshold, this parameter acts only upon real-valued descriptors which can be compared using Euclidean-distance, i.e. only upon the reference pipeline. In Fig. 5.9a can be seen that a smaller threshold results in sufficiently different descriptors and hence a set of putative matches of better quality and generally a higher retrieval performance.

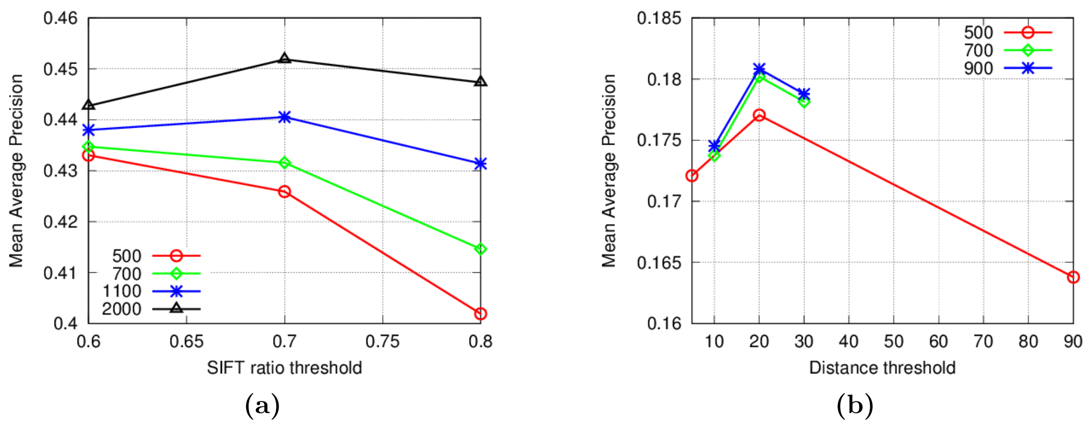


**Figure 5.9: Effect over retrieval performance of varying ratio and distance threshold.** (a) Retrieval performance of the reference pipeline keeping different numbers of top keypoints. (b) Retrieval performance of the tailored pipeline using the best Hierarchical K-Majority scheme and varying the distance threshold from 500 to 900.

Analog to the ratio threshold there is the distance threshold which acts only upon binary features, it's effect upon the tailored pipeline using the best Hierarchical K-Majority scheme is shown in Fig. 5.9b. A distance threshold of 20 tends to produce a better retrieval performance, a smaller or higher value worsens the retrieval performance. This result is not surprising since BRIEF descriptor's length is 256, using a higher value wouldn't discard as many wrong matches while a smaller one results more discriminative.

The third parameter I observe is the minimum number of matches, Fig. 5.10 shows the retrieval performance for both pipelines for a wide range of values. In both cases it is not a crucial parameter for geometric verification and a higher value tends to worsen the retrieval performance. This result can be explained because the quality of the estimated Homography found by RANSAC depends on the quality rather than the quantity of the input putative matches.

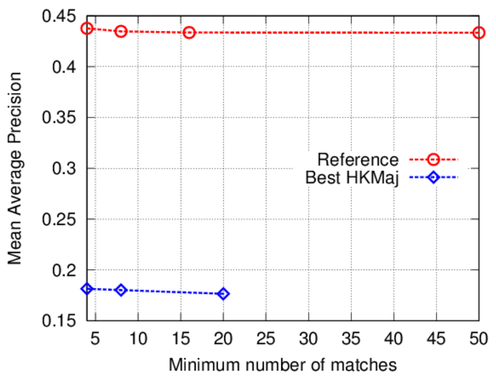


**Figure 5.10: Effect over retrieval performance of varying the minimum number of matches.**

The last evaluated parameter is the RANSAC re-projection threshold, in Fig. 5.11 can be seen that in none of the pipelines the geometric verification algorithm is sensitive towards it. In the case of the reference pipeline it can be explained because the affine variance is captured by the Hessian-Affine detector and hence there is no need to rely on the re-projection threshold to find a better quality model. In the case of the tailored pipeline neither the detector nor the descriptor capture the affine variance of the images and hence we see a small improvement when using 10 pixels over using only 3 pixels. It is worth recalling that in both cases the quality of the homography estimation depends more on the quality of the putative matches.

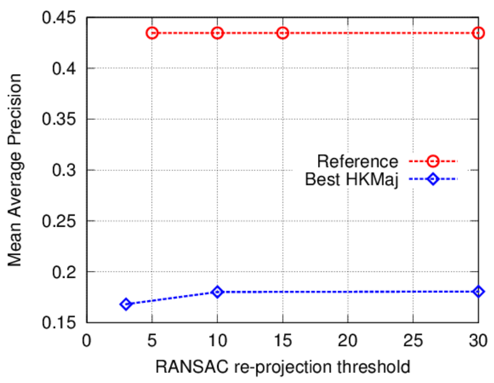


**Figure 5.11: Effect over retrieval performance of varying RANSAC threshold.**

Considering the above tests, the best combination of parameters for the reference pipeline results from keeping the top 2000 keypoints, applying a SIFT ratio threshold of 0.7, requiring to have at least 8 putative matches and using a RANSAC re-projection threshold of 10 pixels. For the tailored pipeline the best combination results from keeping the top 700 keypoints, applying a distance threshold of 20 pixels, requiring to have at least 4 putative matches and using a RANSAC re-projection threshold of 10 pixels. Fig. 5.12 shows the averaged precision-recall curves before and after applying geometric verification for the reference and tailored pipelines using on either case the best set of parameters.

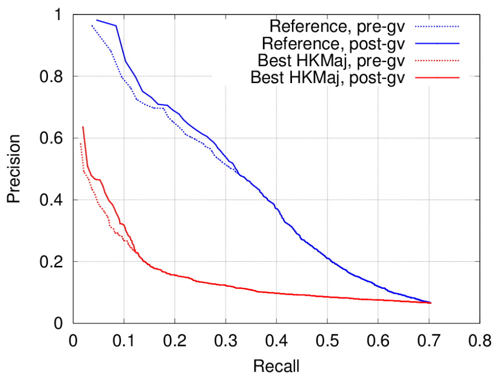


**Figure 5.12: Improvement achieved by running geometric verification.** The pair of red lines correspond to the tailored pipeline using the best Hierarchical K-Majority scheme and the blue lines correspond to the reference pipeline. The dashed lines show the results before geometric verification. The corresponding mAP scores improvements after geometric verification are from 0.4335 to 0.4519 for the reference pipeline and from 0.1721 to 0.1815 for the tailored pipeline.

### 5.3.4 Compare different detector/descriptor pairings

As noticed in [11] pairing different detectors and descriptors serves to better address certain changes in visual appearance. Here the objective is to individually evaluate the dataset transformation effects, in particular scale and rotation. Tab. 5.9 lists the candidate detectors and descriptors together with it's invariance capabilites.

| Detector | Rotation | Scale |
|---|---|---|
| FAST | No | No |
| AGAST | No | No |
| BRISK | No | Yes |
| ORB | Yes | Yes |

| Descriptor | Rotation |
|---|---|
| BRIEF | No |
| D-BRIEF | No |
| ORB | Yes |
| BRISK | Yes |

**Table 5.9:** Summary of the invariance capabilities of binary feature detectors and descriptors

The scale information is provided by the detector and the orientation computation is isolated to the description phase. Among the candidate detectors I discarded ORB because it provides orientation information as well as FAST because AGAST and BRISK are improved versions of it. Regarding feature detectors I chose BRIEF and BRISK as examples of non-rotation and rotation invariant feature descriptors. The final combinations used in this experiment are shown in Tab. 5.10.

| Detector | Descriptor | Evaluated effects | mAP |
|---|---|---|---|
| BRIEF | AGAST | Planar geometry | 0.1972 |
| BRIEF | BRISK | Pure scale | 0.2016 |
| BRISK | AGAST | Pure rotation | 0.1590 |
| BRISK | BRISK | Rotation + Scale | 0.1589 |

**Table 5.10:** Retrieval performance for different combinations of detectors and descriptors used to evaluate the transformation effects

As seen in Tab. 5.1 the number of features produced by the combination AGAST + BRIEF is around 5 times more than the ones produced by DoG + BRIEF. Clustering such a big quantity of data wasn't technically possible and hence I opted for training the vocabulary using HARRIS keypoints which share the same invariance properties than AGAST, and testing it over a database of BRIEF descriptors extracted from AGAST keypoints. In the case of BRISK this problem isn't verified because it produces less keypoints since the scale pyramid in conjunction with non-maximal suppression, filters out non-scale invariant keypoints.

Fig. 5.13 shows the results after running the tailored pipeline using as input the descriptors extracted by each of the detector/descriptor combinations listed in Tab. 5.10.

The precision-recall curves corresponding to the combinations considering rotation invariance have among all the lowest performance. As concluded in [11], this happens because a binary descriptor suffers in performance when it takes into account a transform not present in the data, such as rotation in the case of upright images. The best performance is achieved by the scheme considering only pure scale transformations, this is an expected result since such transformation is present all over the dataset. Surprisingly the scheme considering only planar geometry transformations achieves a retrieval performance competitive with that considering only pure scale transformations.

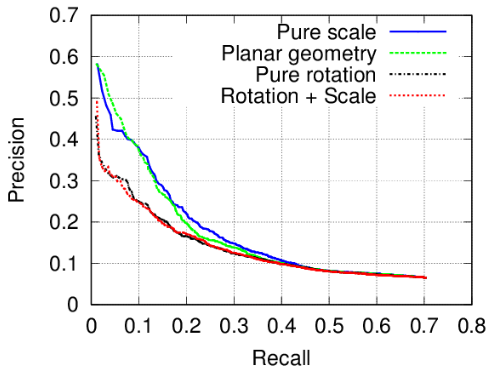
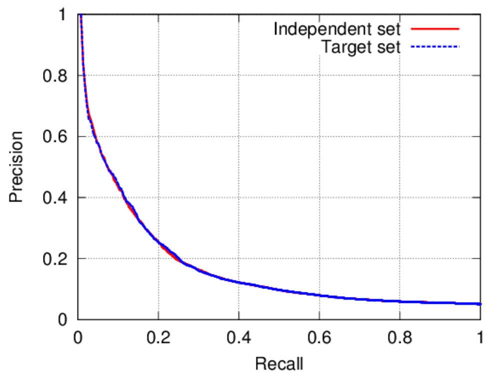


**Figure 5.13: Averaged precision-recall curves for several schemes considering different transformation effects.** In all cases the vocabularies were trained using Hierarchical K-Majority best scheme.

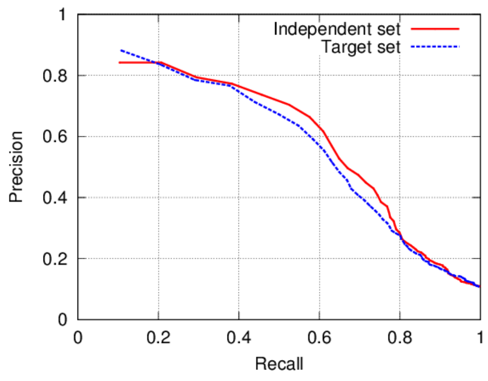
## 5.3.5 Generalization capabilities

A remarkable characteristic of visual vocabularies produced using either Hierarchical K-Means or Approximate K-Means, is their capability to be trained over a set of images independent than the one used for database construction and still be able to achieve a good retrieval performance.

To prove if this holds as well for visual vocabularies trained over binary descriptors using Hierarchical K-Majority, I trained a visual vocabulary over the Oxford5K dataset and tested it over the Paris6K and Rome10K datasets, results are shown in Fig. 5.14a and 5.14b. On the Paris6K dataset the retrieval performance for both vocabularies is unchanged whilst in the Rome10K dataset the vocabulary trained over the independent dataset outperforms the one trained over the target dataset, thus vocabularies produced with Hierarchical K-Majority are agnostic to the dataset used for training. Although in previous experiments it has been demonstrated that retrieval performance depends largely on the quantity of keypoints here it is observed that it also depends on their quality. The images of the Oxford5K dataset have a higher pixel resolution than those from Rome10K and hence the keypoints detected on them are of better quality.



(a)



(b)

**Figure 5.14: Averaged precision-recall curves for the tailored pipeline using vocabularies trained over different datasets.** (a) Results over the Paris6K dataset. The vocabulary trained over an independent dataset scored a mAP of 0.2150 while the vocabulary trained over the target dataset scored 0.2174. (b) Results over the Rome10K dataset. The vocabulary trained over an independent dataset scored a mAP of 0.6813 while the vocabulary trained over the target dataset scored 0.670604. All vocabularies were trained using Hierarchical K-Majority best scheme, taking as input BRIEF descriptors extracted from HARRIS keypoints.

As final experiment I setup a test to see how would my approach behave in an hypothetical system undertaking complex visual analysis tasks with constrains of bandwidth and processing power. The experiment consisted simply in training a small visual vocabulary over binary descriptors and compare it's retrieval performance in terms of mAP score to that achieved in a similar unpublished work (the same were the Rome10K dataset was introduced). On the system used as reference in this experiment, the authors achieved a mAP score of almost 50% using 1.000 Bytes to encode each query BoF vector produced by inter-frame encoding the BoF vectors of the query video frames and combining them by means of median rank aggregation.

Fig. 5.15 shows the results of the experiment, it confirms previous findings indicating that a flatter vocabulary helps to increase retrieval performance; a noteworthy result is that up to a recall level of 10% the precision is at least 45%. Finally, considering that I am using no aggregation scheme but simply taking the arithmetic mean of the individual average precision results of the query videos frames, thus neglecting the relation between them, the direct comparison between the achieved mAP scores does not seem very fair. Using some aggregation scheme such as median rank aggregation might improve the retrieval performance on the Rome10K dataset, however such topics are out of the scope of this work.

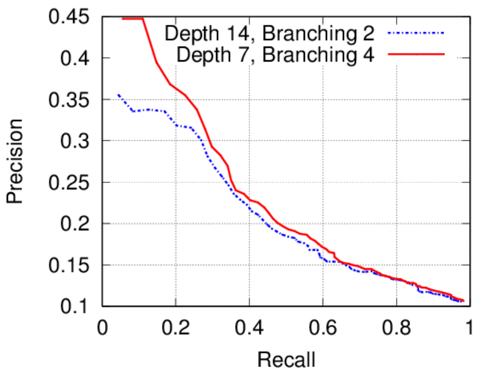


**Figure 5.15: Averaged precision-recall curves for two 16K words vocabularies trained from BRISK descriptors extracted from BRISK keypoints.** The blue dash-dotted line corresponds to a rather deep vocabulary, and the red solid line corresponds to a flatter one. The achieved mAP scores are 0.3055 and 0.3296 correspondingly.

# Conclusions

The need for an alternative or complementary positioning mechanism for scenarios where GPS readings are unreliable or unavailable, motivated the use of images as a visual fingerprint for places recognition. The challenges introduced to the problem of location recognition in a mobile environment claim more adequate techniques. To deal with such challenges this work set out to investigate the use of binary local images features to address the problem of large-scale landmark recognition in the frame of the BoF model.

## Restatement of aims

The main objectives of the present study were to determine how representative could be vocabularies built starting from binary features in the landmark recognition task and to identify which factors govern retrieval performance in a system using such vocabularies. To accomplish these goals it were analyzed many issues about clustering binary data and it were proposed two approaches to build visual vocabularies starting from binary descriptors. Grounded on this ideas it was implemented a proof-of-concept system to validate them in the following aspects: BoF retrieval, ability to produce models of the features spatial distribution, and generalization capabilities.

## Empirical findings

The first major finding of this investigation was that vocabularies built from binary descriptors using Hierarchical K-Majority are less representative than their real-valued counterpart built from SIFT descriptors. A denser description of the images helps to some extent to achieve more representative vocabularies, overcoming binary features' low descriptive power in comparison with SIFT. Another strategy found to increase representativity is –taking the right considerations– to use different feature detectors for training and query images, e.g. Hessian-Affine detector combined with BRIEF descriptor for training images description and FAST detector paired with BRIEF descriptor for query images description. Lastly keeping high the ratio of the number of descriptors to the number of words increases the quality of the vocabulary and thus benefits retrieval performance.

The second major finding was that in pure content-based landmark recognition pipelines as the ones used in this work, i.e. without context information, the most influential factor governing retrieval performance is the utilized detector and descriptor for dataset images description. Concretely a rough analysis of the transformation effects present in the dataset serves to chose a fair detector/descriptor combination which helps boosting retrieval performance. Likewise, using a detector capable of increasing the quantity of keypoints[8] and thus the number of descriptors allows to: 1) address the fast detectors lack of invariance to perspective distortion effects, 2) estimate better models of the features spatial distribution, and 3) increase the representativity of the vocabulary and the retrieval performance. In the case of binary features the use of fast detectors, the descriptors' compactness, and the possibility to efficiently compare them, enables using more keypoints.

## Theoretical implications

The foundations behind clustering binary data streams need to be revisited in order to better understand the dynamics behind generating higher quality clusters in less time and how they can fit into the BoF model. Taken together the findings of this study suggest that although K-Majority is the most reasonable option it is rather not an optimal one because it produces clusters of dubious quality.

Regarding image description the evidence from this study suggested that: 1) salient keypoints aren't optimal for landmark images retrieval, confirming the findings of [18], and 2) retrieval performance can be outperformed with a denser description. An implication of this is that features corresponding to surroundings do help recognition as pointed by [2].

## Significance of the findings

The present study makes several noteworthy contributions to the existing base of knowledge in visual recognition. First, this is one of the earliest works in proposing and exhaustively evaluating a method for visual places recognition using BoF vectors obtained from binary features. Second, this work presented thorough review of the issues related to clustering binary data. Third, to promote further research in the area I published at GitHub[9] the source code of my implementation. Among the released code I contributed to the OpenCV community the wrappers of the authors' implementations of the AGAST feature detector and D-BRIEF features descriptor

[8]Something impossible with salient point detectors such as DoG but possible with fast detectors such as AGAST.

[9]https://github.com/gantzer89/VLRPipeline

into the OpenCV feature extraction framework. And fourthly, the methods presented here and the lessons learned are applicable to other areas such as visual robot localization.

## Limitation of the study

The findings derived from this study are subject to some limitations that must be considered for proper interpretation. The major limitation is the low retrieval performance achieved by the Hierarchical K-Majority approach. Another limitation it's related with the generalizability of the findings because although the evidence indicates that flatter vocabularies show higher retrieval performance, it wasn't possible to build flat vocabularies of enough size. Similarly the generalizability it's compromised because the experiments that led to them were performed only over datasets composed by images crawled from the Web. On the other hand the validity of the findings lack contrast because besides K-Majority no alternative clustering approaches were explored.

## Future work

Aiming at increasing retrieval performance and overcoming some of the described limitations, in this section are discussed some potentially promising directions for future work.

The current implementation of this system is capable of clustering descriptors in the order of millions, extracted from a few thousands of images, nevertheless datasets at the city-scale contain many more[10]. To deal with large-scale training sets it is necessary to store database images descriptors in a sufficiently fast database engine. Such condition would require also a significant reduction in the vocabulary construction time for which I propose to explore the possibility of parallel processing.

In this work it weren't explored many alternatives for clustering binary data but only discussed the issues around it. Future trials should investigate other approaches such as the ones described in [19].

Considering the results of the tests regarding flat vocabularies, it is strongly recommended further research into its use. To do so it would be necessary to investigate some alternative algorithms for ANN search such as LSH or HCT, for the latter however it would be required a better understanding of the issue identified in the previous chapter.

[10]An example is the San Francisco landmarks dataset, firstly used in [5]. It consists of 1.7 million perspective images extracted from approximately 150K panoramic images, and 803 cell phone query images.

To complement the analysis in terms of retrieval performance it would be of great help to assess the improvement in terms of query retrieval times of pipelines built on top of binary features to that of pipelines built on top of SIFT descriptors.

Even if the achieved retrieval performance is lower than the performance achieved by state-of-art techniques, it would be interesting to use the lessons learned here to address the location recognition problem. To do so it would be necessary to use a dataset with associated geographic information using as input query videos in conjunction with some scheme for aggregating frames information.

## Final words

The possibility of describing images at low computational cost in an efficient manner enables performing complex visual analysis tasks in devices with low resources. Bridging this gap between state-of-art visual recognition methods and computers with low processing and storage capabilities sets this work as an innovative contribution to the base of knowledge in visual recognition.

Despite the numerous limitations of this study, the findings achieved remain useful and promising. It is expected that such findings as well as the released code have a positive impact on the research on visual recognition and encourage future developments.

# Acknowledgments

This thesis is an effort in which, directly or indirectly, several people have participated either reading, reviewing, translating, having patience with me, giving me encouragement, or simply accompanying me in moments of crisis or sporadic happiness.

First of all I want to thank to my thesis supervisor for his advice, knowledge and guidance during the whole development of my work.

Then I would also like to individually express my deep gratitude to my whole family. To my lovely mother for her unequivocal supportive words and for having always trusted without any warranty in the path that I have chosen. To my father who since the beginning supported my decision to make this master's program and always gave me his unconditional support, without him the realization of this thesis would have been impossible. And to my twin sister whose admiration and brave spirit have encouraged me to continue in the moments of weakness by coloring my life.

I must also thank my boss, Federico, who without even asking, lent me a hand when I needed it most.

I would also like to thank my friends (both those who have already gone and those who are still with me) who for these three years have accompanied me on the adventure that has been the culmination of my graduate studies, who unconditionally understood my constant absences and bad times, who fed me and helped me in countless ways, but especially for the spiritual strength they gave me to finish this work.

At this point I want to make a special mention to my great high school friend David Duque, who when alive instilled in me his love for academia and who with his unique personality showed me that dedication and perseverance always triumph.

To all of you, my most sincere and visceral thanks.

# Bibliography

[1] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918, 2012.

[2] Georges Baatz, Kevin Köser, David Chen, Radek Grzeszczuk, and Marc Pollefeys. Leveraging 3d city models for rotation invariant place-of-interest recognition. *International Journal of Computer Vision*, 96:315–334, 2012.

[3] Priyadarshi Bhattacharya and Marina Gavrilova. A survey of landmark recognition using the bag-of-words framework. In Dimitri Plemenos and Georgios Miaoulis, editors, *Intelligent Computer Graphics 2012*, volume 441 of *Studies in Computational Intelligence*, pages 243–263. Springer Berlin Heidelberg, 2013.

[4] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 778–792. Springer Berlin Heidelberg, 2010.

[5] David M. Chen, Georges Baatz, Kevin Köser, Sam S. Tsai, Ramakrishna Vedantham, Timo Pylvänäinen, Kimmo Roimela, Xin Chen, Jeff Bach, Marc Pollefeys, Bernd Girod, and Radek Grzeszczuk. City-scale landmark identification on mobile devices. In *CVPR*, pages 737–744. IEEE, 2011.

[6] Tao Chen, Kim-Hui Yap, and L-P Chau. Integrated content and context analysis for mobile landmark recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(10):1476–1486, Oct. 2011.

[7] D. Gálvez-López and J.D. Tardós. Bags of binary words for fast place recognition in image sequences. *Robotics, IEEE Transactions on*, 28(5):1188–1197, 2012.

[8] Costantino Grana, Daniele Borghesani, Marco Manfredi, and Rita Cucchiara. A fast approach for integrating orb descriptors in the bag of words model. *Proc. SPIE*, 8667:866709–866709–8, 2013.

[9] Qiang Hao, Rui Cai, Zhiwei Li, Lei Zhang, Yanwei Pang, and Feng Wu. 3d visual phrases for landmark recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3594–3601, June 2012.

[10] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[11] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Comparative evaluation of binary features. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, Lecture Notes in Computer Science, pages 759–773. Springer Berlin Heidelberg, 2012.

[12] Pierre Legendre and EugeneD. Gallagher. Ecologically meaningful transformations for ordination of species data. *Oecologia*, 129(2):271–280, 2001.

[13] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555, 2011.

[14] Heng Liu, Tao Mei, Jiebo Luo, Houqiang Li, and Shipeng Li. Finding perfect rendezvous on the go: accurate mobile visual localization and its applications to routing. In *Proceedings of the 20th ACM international conference on Multimedia*, MM '12, pages 9–18, New York, NY, USA, 2012. ACM.

[15] Elmar Mair, GregoryD. Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6312 of *Lecture Notes in Computer Science*, pages 183–196. Springer Berlin Heidelberg, 2010.

[16] M. Muja and D.G. Lowe. Fast matching of binary features. In *Computer and Robot Vision (CRV), 2012 Ninth Conference on*, pages 404–410, 2012.

[17] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *IN CVPR*, pages 2161–2168, 2006.

[18] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *In Proc. ECCV*, pages 490–503. Springer, 2006.

[19] Carlos Ordonez. Clustering binary data streams with k-means. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, DMKD '03, pages 12–19, New York, NY, USA, 2003. ACM.

[20] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[21] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

[22] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11:2487–2531, December 2010.

[23] Paul L Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291–307, 1999.

[24] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515 Vol. 2, 2005.

[25] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin Heidelberg, 2006.

[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571, 2011.

[27] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Towards fast image-based localization on a city-scale. In *Proceedings of the 15th international conference on Theoretical Foundations of Computer Vision: outdoor and large-scale real-world scene analysis*, pages 191–211, Berlin, Heidelberg, 2012. Springer-Verlag.

[28] Grant Schindler, Matthew Brown, and Richard Szeliski. City-scale location recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–7. IEEE, 2007.

[29] G. Schroth, A. Al-Nuaimi, R. Huitl, F. Schweiger, and E. Steinbach. Rapid image retrieval for mobile location recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2320–2323, 2011.

[30] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, and E. Steinbach. Mobile visual location recognition. *Signal Processing Magazine, IEEE*, 28(4):77–89, 2011.

[31] J. Sivic and A. 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 vol.2, Oct 2003.

[32] Tomasz Trzcinski and Vincent Lepetit. Efficient discriminative projections for compact binary descriptors. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, volume 7572 of *Lecture Notes in Computer Science*, pages 228–242. Springer Berlin Heidelberg, 2012.

# Nomenclature

| | |
|---|---|
| AGAST | Adaptive Generic Accelerated Segment Test |
| AKM | Approximate K-Means |
| ANN | Approximate Nearest Neighbors |
| BoF | Bag-of-Features |
| BRIEF | Binary Robust Independent Elementary Features |
| BRISK | Binary Robust Invariant Scalable Keypoint |
| CBIR | Content-based Information Retrieval |
| DOF | Degrees of freedom |
| DoG | Difference-of-Gaussians |
| HARRIS | Harris corner detector |
| HCT | Hierarchical Clustering Trees |
| HKM | Hierarchical K-Means |
| KMAJ | K-Majority |
| KNN | K-Nearest neighbors |
| LSH | Local-sensitive hashing |
| RANSAC | RANdom SAmple Consensus |
| SIFT | Scale Invariant Feature Transform |