Sparse Binary Features for Image Classification

Master thesis

Johan Paratte
EPFL EE LTS2 – 2013
johan.paratte@epfl.ch

Supervised by:

Pierre Vandergheynst – EPFL
Signal Processing Lab
http://lts2www.epfl.ch

Alexandre Alahi – Stanford University
Vision Lab
http://vision.stanford.edu/

Abstract

The past decade has seen the rise of cheap and ubiquitous access to cameras, especially by its systematic embedding into mobile phones. In conjunction with the increase of network bandwidth and a large worldwide commitment to online social media, we face today a tremendous amount of multimedia data.

The rate at which the quantity of data increases is sufficiently high to prohibit the idea of the traditional but cumbersome and lengthy process of human-driven data organization and labelling. This induces a need of automatic processing of multimedia data. Image classification is a good example of such a task.

In this work a new method for automatic image classification is proposed. It relies on a compact representation of images using sets of sparse binary features. This work first evaluates the Fast Retina Keypoint binary descriptor and proposes improvements based on an efficient descriptor representation. The efficient representation is created using dimensionality reduction techniques, entropy analysis and decorrelated sampling.

In a second part, the problem of image classification is tackled. The traditional approach uses machine learning algorithms to create classifiers, and some works already propose to use a compact image representation using feature extraction as preprocessing. The second contribution of this work is to show that binary features, while being very compact and low dimensional (compared to traditional representation of images), still provide a very high discriminant power. This is shown using various learning algorithms and binary descriptors.

These years a scheme has been widely used to perform object recognition on images, or equivalently image classification. It is based on the concept of Bag of Visual Words. More precisely, an image is described using an unordered set of visual words, that are generally represented by feature descriptions. The last contribution of this work is to use binary features with a simple Bag of Visual Words classifier. Tests of performance for the image classification are performed on a large database of images.

Contents

Acronyms

ANN	Artificial Neural Network
BRISK	Binary Robust Invariant Scalable Keypoints
BRIEF	Binary Robust Independant Elementary Features
FREAK	Fast Retina Keypoint
GBT	Gradient Boosted Trees
kNN	k Nearest Neighbors
LSH	Local Sensitive Hashing
ORB	Oriented BRIEF
PCA	Principal Component Analysis
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine

Introduction

Today we face an unprecedented data deluge for which we need automatic and intelligent processing. Indeed, as we entered in the era of free and easy to use social media, the amount of data available online becomes overwhelming. This can be explained by the fact that this data can be extremely redundant (i.e. present in many places in different quality or format), poorly indexed and sometimes of questionable quality.

To summarize in one word, the Big Data we have access to is highly disorganized. Given the fact that some portion of it is not associated with semantic information, one cannot rely on metadata and must thus devise data-aware processing.

We can split the organization into two category of tasks : clustering and classification. The first one is needed to automatically create groups of similar objects and the latter is needed to add meaningful semantic information to it. Of course, these two operations should be interconnected as they can highly benefit from one another.

One of the digital data type which occurs the most is visual image-based content such as pictures, videos, scanned documents, and so on. As humans we have still a clear superiority over artificial computer based vision, especially for tasks related to identification, recognition and classification. We are so good at this that we seem to forget how difficult the task is. One fact which makes it especially challenging is that recognition relies on memory and *in fine* on learning.

That is why automatic processing of image-based content is currently still considered as a hard problem and involves many fields of research such as image processing, computer vision, machine learning and large scale databases.

In this work, we address the image classification task, while keeping in mind this Big Data context. We thus propose an efficient method for image classification based on a compact representation of images using sparse binary features.

The first part of this work will remind theoretical foundations needed to develop feature-based image classification methods. In the first section we will review the feature-based image description pipeline and describe different keypoint detectors and descriptor extractors. In the second section several concepts of dimensionality reduction techniques will be presented, as they can be useful to create more compact representation of high-dimensional data. The third section is an overview of machine learning methods both for clustering and classification tasks.

In the second part, we will first propose an improvement of the Fast Retina Keypoint binary feature descriptor. We will evaluate the potential of description of a compact representation of the descriptor using distance preserving dimensionality reduction, entropy analysis and decorrelated sampling. In the last section we will start by evaluating the discriminant power of binary descriptors in the context of image classification and finally use a Bag of Visual Words classifier with sparse binary features to perform image classification on a challenging image dataset.

Part I
Theoretical Foundations

1 Feature-based image description

1.1 Purpose

An image, with the meaning of digital raster image, is a two dimensional array of discrete values, potentially multi-channel. This pixel array representation is useful to store and display images or for general image processing but is not suitable for several kind of processing techniques. One of the reason to prefer an alternative representation is that an image is a very high-dimensional object, i.e. it is a vector in $\boldsymbol{Z}_q^{w \times h \times c}$, where w is the width, h the height, c the number of channels and q the quantization level. A typical example would be a full HD color image with each color channels quantized on 8 bits. The space in which lives the image is huge[1] and thus prevents several kind of processing, especially those very sensitive to dimensionality such as machine learning, as we will see in the following sections.

Even if the space in which an image lives is of high-dimension, it happens that most images come from the real world and thus contain non-distinctive parts (such as patches of uniform color). It becomes then very useful to extract only meaningful parts of the image, in other words to retrieve only a small amount of very discriminative image areas. A common way in computer vision to fulfil this purpose is to find locations containing a lot of information, and then to describe them using their local pixel neighbourhood. Such points are generally called keypoint features (or interest points), and their description is generally referred to as descriptors.

The feature-based image description is thus a collection of features and their respective descriptors. The problem of high-dimensionality of the standard representation is solved since descriptors live in (arbitrarily) low-dimensional spaces. Let us note that in general a feature is not restricted to interest points but can be anything which has a high discriminant power (e.g. edges). In this work we will however only consider keypoint features in our feature-based image representation. Many techniques rely on this alternative representation either for its compact form or for its focus on interest points. Numerous examples can be found in the literature such as use in object retrieval [38][24], 3D stereo reconstruction [50] or image stitching [7][46] to name a few.

1.2 Feature points detection

As stated above, the first requirement of the feature-based image representation is to find discriminative image patches. This process is commonly known as feature detection and has been solved using several approaches. The common goal of the many techniques is to find locations at the center of image patches which have to be easy to track. The patches need to be highly textured (as uniform color is impossible to track), possess a strong contrast (i.e. high image gradient) and at the same time avoid the aperture problem [25][34] [3]. Intuitively, the only kind of feature which respects these constraints is a patch that possess at least a pair of sufficiently orthogonal gradient directions. A typical example is a patch containing a corner, which explains why this process is sometimes called *corner detection.*

[1] in $\boldsymbol{Z}_{256}^{6082560}$

Auto-correlation based detection

A simple way to measure the quality of a keypoint is to compute the autocorrelation associated to its patch, i.e. to compute the correlation between the patch and small shifts of itself :

$$R_{II}(\Delta\mathbf{u}) = \sum_i I(\mathbf{x_i})I(\mathbf{x_i} + \Delta\mathbf{u}) \tag{1}$$

where I is the image patch, $\mathbf{x_i}$ a pixel location and $\Delta\mathbf{u}$ a shift. A good and stable keypoint would yield a strong local minimum.

Using the Taylor Series expansion of the shifted term $I(\mathbf{x_i} + \Delta\mathbf{u}) \approx I(\mathbf{x_i}) + \nabla I(\mathbf{x_i})$, one can derive the following formulation of the autocorrelation[2] :

$$R_{II}(\Delta\mathbf{u}) = \Delta\mathbf{u}^T A \Delta\mathbf{u} \tag{2}$$

with

$$A = w * \nabla I \nabla I^T = w * \begin{bmatrix} \dfrac{\partial^2 I}{\partial x^2} & \dfrac{\partial^2 I}{\partial x\,\partial y} \\ \dfrac{\partial^2 I}{\partial y\,\partial x} & \dfrac{\partial^2 I}{\partial x^2} \end{bmatrix} \tag{3}$$

where w is a weighting factor accounting for the local autocorrelation estimate and $*$ represents the convolution operation.

An eigenvalue analysis of the matrix A gives two eigenvalues and two corresponding eigenvectors defining an ellipse. From the autocorrelation matrix and its eigenvalues $(\lambda_{min}, \lambda_{max})$, various ways of detecting keypoints have been proposed.

In Shi and Tomasi [44] they used the minimum eigenvalue λ_{min}, while in Harris [23] they used the quantity[3] $\lambda_{min}\lambda_{max} - \alpha(\lambda_{min} + \lambda_{max})^2$ to name the two most famous. The general method used by all these auto-correlation based detectors is the same :

1. compute the partial derivatives[4] (generally noted I_x and I_y) using a convolution with derivatives of Gaussians

2. compute the outer product of $[I_x\ I_y]$ to get the autocorrelation matrix A

3. compute a measure based on eigenvalues analysis

4. choose all points where this measure has a value above a given threshold

These detectors, while giving reasonably good results, are quite simple and can be improved by considering scale, rotation and affine invariance. We will give a brief overview of different detectors providing robustness to one or more of these characteristics.

[2]see [47] for a full derivation

[3]this quantity can be efficiently computed as it is simply $\det(A) - \alpha \cdot \mathrm{trace}(A)^2$

[4]$\dfrac{\partial I}{\partial x}$ and $\dfrac{\partial I}{\partial y}$

Scale-invariant detection

The robustness to scale (or scale invariance) is a highly desired property for a keypoint feature detector since the image scale is *a priori* unknown. Indeed, a simple example would be different pictures of the same object with different levels of zoom. In such a case, a large-scale (i.e. low gradient frequency) feature would not be detected at a fine image scale. To overcome this issue, the simplest solution would be to detect features at different scales and accounting for all the detections at each scale. However the adopted solution was a bit different, mainly to improve the efficiency. Indeed, the authors of [33] and [36] choose to select features that were stable in space (in the sense of the previous section) and in scale at the same time, instead of taking all detected features at each scale.

The well known Scale Invariant Feature Transform (SIFT) feature detector uses such a space and scale combination for the detection [33]. The authors proposed to use a pyramid of Difference of Gaussian, and then to search for maxima in both scale and space at the sub-octave level. The location at the sub-pixel level is then computed using a quadratic interpolation.

It is worth noting that the features detected by the SIFT detector are different from the ones detected by auto-correlation based detectors such as Harris detectors. In fact the features of the two kinds of algorithms are rather complementary[5].

Efficient detection

The detectors presented above provide features of very good quality but are computationally intensive and can thus be impossible to use in real-time applications, e.g. for mobile devices or even in very large scale configurations. This is why some research has been done in order to provide very efficient detectors.

One of the most famous detectors, designed to be very efficient, is the detector called Features from Accelerated Segment Test (FAST)[41]. It is a morphological feature detector, meaning it is based on morphological operations rather than convolutional-based operations. It does not require second order derivatives and thus no prior denoising. This difference accounts for a large part in the efficiency gain.

The FAST criterion for corner detection works as follows. A ring of pixels around a given point is considered (i.e. the segment test). If a given number of consecutive pixels in the ring have sufficient difference in intensity than the pixel of the central point, then the latter is considered as a detected keypoint. The authors of FAST proposed a learned sequence of pixel evaluations lowering the number of tests to perform (i.e. accelerated segment test) and thus improving a bit more the efficiency of the detection. Finally a non-maximal suppression step is performed to remove corners which have an adjacent corner with a higher response.

This method was improved by the authors of [35] who proposed the Adaptive and Generative Accelerated Segment Test (AGAST) feature detector. Their work uses the same segment test as the FAST, but the authors proposed to use optimal decision trees to provide a generic accelerated segment test and thus avoid the learning step of FAST.

FAST and AGAST are very efficient detectors which provide corners of good quality. However they lack scale and rotation invariance as the detection is done in space only. To overcome this issue the authors of [31] proposed an extension of the AGAST detector called Binary Robust Invariant Scalable Keypoints (BRISK).

The method used in BRISK relies on scale-space detection. For this purpose, a scale-space pyramid of n octaves and sub-octaves are created by half-sampling. A step of FAST

[5]see for example [47] p.215

detection is done in each octaves and sub-octaves using a same threshold. The following step is a non-maximal suppression in both scale and space. This means that a point is considered as a valid keypoint if (1) it has a greater FAST score than its neighbours in the scale domain in its own layer and (2) it is also greater than the corresponding scores in layers above and below it. Finally, the true scale of the point is determined by quadratic interpolation, and the location at sub-pixel level is defined as the closest maxima in the corresponding scale.

The rotation invariance is not provided by the detector but by the corresponding BRISK descriptor which will be presented in the next section.

1.3 Keypoint description

Once the features are detected, i.e. the keypoints are found, they have to be described. Usually this is done using a local neighbourhood of pixels around the keypoint location. Several characteristics can be desired for a good keypoint descriptor. First, the size (i.e. dimension) of the descriptor need to be bounded to a sufficiently small value, in order to keep the benefits of the feature-based image representation, and to be able to match features efficiently. For matching purposes, the description should be descriptive enough to allow a good discrimination but at the same time robust to variations of the surrounding pixels of the point corresponding to a same object. Robustness can be wanted for in-plane rotation, scale, affine or perspective transformation, compression noise, etc. As we saw in the previous section, scale and rotation information can be provided by the detector.

The keypoint descriptors were traditionally real-valued, as they generally encode normalized difference of gradients (i.e. orientations). We will first look at some of the classical ones and then present more recently developed binary descriptors.

1.3.1 Real-valued descriptors

SIFT

One of the most famous and widely used keypoint descriptor is the SIFT, introduced at the same time as the detector which was briefly described before. The description of a detected keypoint is done using the following method.

- The image in the pyramid whose scale is the closest to the keypoint scale is selected,
- A Gaussian weighting function centered on the keypoint is applied to give more importance to the pixels near the center of the patch,
- The patch is discretized in a square grid of 16 cells.
- A 8-bin histogram of gradients is computed for each cell.
- The 128 (16 x 8) values are normalized and form the SIFT descriptor.

This description is supposed to provide invariance to various parameters such as viewpoint or illumination changes. The final SIFT descriptor is a 128-dimensional real valued vector.

Many extensions of the SIFT descriptor have been proposed such as PCA-SIFT[28] that replaces the Gaussian weighted histogram of gradients by a Principal Component Analysis (PCA) applied to the normalized grid of gradients. Another example is the Color SIFT[53] which uses an opponent color model instead of the standard mono-component grey values intensity model. More recently, a method called RootSIFT was proposed in [4] which essentially does not modify the descriptor but the metric used to compute distances between them. According to the paper, taking the Hellinger distance instead of the usual Euclidean distance improves the matching of the descriptors.

SURF

Another well known algorithm for keypoint description is the Speeded-Up Robust Features (SURF)[6] descriptor[6]. It relies on integral images to approximate convolutions, which provides an appreciable improvement in efficiency (compared to SIFT for example).

The descriptor extraction algorithm of SURF works as follows :

- A weighting Gaussian function centered at the keypoint is applied
- The general orientation is computed using sums of Haar wavelet filter responses.
- The patch is divided to form a square grid (oriented in the direction of the general orientation computed in step (2)). The grid consists of 16 cells (called subregions).
- Orientations inside each cell is computed using Haar wavelet filter responses.
- A vector of 4 different sums of the orientations is computed in each cell.
- The 64 (16 x 4) values are normalized and form the SURF descriptor.

Variants such as SURF-36 (3x3 grid) or SURF-128 (extended set of features) are based on the same method. The SURF descriptor is supposed to provide invariance to illumination, viewpoint and contrast variations. The standard SURF descriptor is a 64-dimensional real valued vector.

1.3.2 Binary descriptors

As we will introduce binary descriptors let us first define the notation. We will denote the finite field of two elements $GF(2)$ with the Hamming distance metric[7] as $\boldsymbol{Z}_2$.

All the descriptors we reviewed above were high-dimensional real-valued vectors. But since floating point values need to be encoded as binary strings of 32 bits[8], the vector corresponding to a descriptor of 128 floating point values is in fact a much higher dimensional binary vector.

A solution to this problem of poor distribution of the information over the usable bits can be to provide a better encoding and rely generally on hashing applied to the real-valued descriptor [2][24][38]. But one might want to use this huge space directly by using binary descriptors.

Other advantages of the binary descriptors are that they can be more compact that traditional descriptors[9] and they are much faster to match. Indeed a matching between binary descriptors makes use of the Hamming distance which can be implemented as a XOR operation followed by a bit count and such operations can be computed extremely efficiently even on low-end or mobile devices.

[6]SURF provides in fact both detector and descriptor
[7]later noted $|\,.\,|_H$
[8]for most single-precision implementations of real value variables
[9]since there is a one-to-one correspondence between the number of dimension and the number of bits required to encode it

BRIEF

The Binary Robust Independent Elementary Features (BRIEF)[9] descriptor is a good example of binary descriptor which possess the qualities we just described. In addition to these, the BRIEF descriptor provides a descriptor extraction algorithm that is way faster than any standard real-valued descriptor.
The description extraction is fairly simple and follows these steps :

- A Gaussian smoothing is applied to the patch.
- The difference of intensity of 512 pairs[10] of pixels is computed and thresholded using a step function. In other words the descriptor is formed of the 512 values computed as the binary sign of the pairs differences.
- The location of the pairs is precomputed and randomly drawn from a Gaussian distribution centered at the keypoint.

By itself BRIEF is neither scale nor rotation invariant. This is why the authors of [42] proposed an extension of BRIEF providing rotation invariance and more robustness to noise that is called Oriented BRIEF (ORB).

BRISK

Inspired by the efficiency of the BRIEF descriptors, the authors of the BRISK detector proposed an associated binary descriptor. The description is also a binary sign of a differences between pairs of locations. The two main differences with BRIEF is that BRISK is a rotation invariant descriptor and uses a special pattern to sample the pairs instead of a random distribution. The algorithm for the extraction of the descriptor is the following :

1. A concentric sampling pattern is created.
2. A set of long range pairs and a set of short range pairs are created.
3. The global orientation is estimated using the set of long range pairs.
4. The sampling pattern is rotated in the direction of the computed orientation.
5. In order to avoid noise, the sampling pattern defines Gaussians centered at each point, so that the differences of the pairs are in fact differences of Gaussians
6. The descriptor is constructed using a determinitic set of 512 short range pairs

FREAK

Extending the works of [31] on the BRISK descriptor, the authors of [1] proposed a descriptor called Fast Retina Keypoint (FREAK) to continue on the way of bitstream containing Differences of Gaussians. They proposed a new sampling pattern inspired by the human retina that has two main differences with the one proposed in BRISK. The Gaussians are called receptive fields in reference to the human retina.

The first contribution is an allocation of the receptive fields using a concentric distribution and of size exponentially increasing with the distance to the keypoint. The second contribution is to choose a pattern that creates overlaps between the different receptive fields.

[10]256 pairs for BRIEF-32

The sampling pattern given in the original implementation of the FREAK descriptor contains 43 receptive fields, which leads to 903 possible pairs. For efficiency purposes and to avoid having too much correlation between the pairs, the authors propose a method to select a subset of 512 pairs from the 903.

This process is referred to as pairs selection and we will develop it more in the second part of this work. It is also a good example of a concept called dimensionality reduction, which is the subject addressed in the next section.

2 Dimensionality reduction

2.1 Purpose

Dimensionality reduction (also called dimension reduction) is a very general subject and can be used in many fields such as numerical analysis, machine learning, data mining and databases. It is very useful in particular for all the domains that suffer from the so-called *curse of dimensionality*. This term refers to the problems occurring as the volume of a space increases very fast in function of its dimension, resulting with a sparse repartition of the data. The problem is that statistical methods are often not suitable to process sparse data. In other words, as the dimension grows, the amount of data needed to avoid sparsity (and thus ensure valid statistics) grows exponentially. The sparse repartition of the data also makes data association and organization very inefficient. Dimensionality reduction is thus often used to tackle this problem.

Another argument for dimensionality reduction is the fact that in many cases, some dimension can contain redundant information or even no information (e.g. white noise). In this case, dimensionality reduction can even improve the quality of the data. A very well known example of dimensionality reduction is the Principal Component Analysis (PCA) which is an orthogonal linear transformation that uses the eigenvectors of the covariance matrix to select a new basis which maximizes the variance on each direction (also called component). PCA is widely used and has many implementations.

2.2 Distance preserving dimensionality reduction

In a general setup, only a subset of the metrics or properties of the data can be preserved in the process of dimensionality reduction. If a given metric is the only thing we need to measure in the high-dimensional space, then we need a projection to a low-dimensional space which preserves only this metric. Distance preserving dimensionality reduction is a good example of this concept.

2.2.1 Random Projection in $\mathbb{R}$

Random projection is a technique which is used to project points from a high-dimensional Euclidean space to a low-dimensional subspace whose basis is chosen randomly. More precisely, let us have a set of n-dimensional points that we would like to project to a k-dimensional subspace, with $k < n$. The projection can be represented by $n \times k$ orthonormal projection matrix. The projection is thus said to be orthogonal. If we have a vector u in a n-dimensional space, let us call its projection in a k-dimensional space v, then the projected vector will be :

$$v = \sqrt{\frac{n}{k}} R^T u \tag{4}$$

where R is the random projection matrix.

The columns of R form a basis of the k-dimensional subspace, and are chosen randomly. In other words this is a simple projection with a rescaling, which is needed in order to have $\mathbb{E}[\|v\|^2] = \|u\|^2$.

The interesting property of random projection is that it is a distance preserving dimensionality reduction. It is due to the Johnson-Lindenstrauss lemma which states in essence that a collection of high-dimensional points can be embedded into a much lower dimensional space without changing relative distances between them.

Theorem 1. *(Johnson-Lindenstrauss lemma) Let $\epsilon \in (0, \frac{1}{2})$. Let $Q \subset \mathbb{R}^n$ be a set of n points and $k = \frac{20 \log n}{\epsilon^2}$. Then there exists a Lipschitz mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ such that :*

$$(1 - \varepsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \varepsilon)\|u - v\|^2$$

for all u, $v \in Q$

The initial proof of this theorem can be found in the original paper [27]. A simpler proof was given later by the authors of [12] who also proved several extensions of the original lemma. Essentially they proved a similar result where the entries of the random projection matrix were independent samples of a normal distribution $N(0, 1)$. This is formalized in the following theorem.

Theorem 2. *(Norm preservation - Gaussian) Let $u \in \mathbb{R}^n$. Assume that the entries of the projection matrix $R \subset \mathbb{R}^{k \times n}$ are sampled independently from $N(0, 1)$. Let $v = \frac{1}{\sqrt{k}} R^T u$. Then,*

$$\begin{aligned}&(1)\, \mathbb{E}\left[\|v\|^2\right] = \|u\|^2 \\ &(2)\, \mathbb{P}\left[|\|v\|^2 - \|u\|^2| \geq \varepsilon\|u\|^2\right] \leq 2e^{-(\varepsilon^2 - \varepsilon^3)\frac{k}{4}}\end{aligned}$$

As emphasized by the authors, the distribution does not need to be Gaussian, and other distributions with certain properties[11] are also valid for the projection matrix creation (e.g. by sampling a uniform random variable $U(-1, 1)$). It is also worth noting that as a corollary of the previous theorems, inner products are also preserved under random projection. For the proof see [12].

2.2.2 Random Projection in $\boldsymbol{Z}_2$

As we saw in the previous section, some objects like binary descriptors are not in $\mathbb{R}^n$ but in $\boldsymbol{Z}_2^n$, with the Hamming distance for metric instead of the Euclidean distances. Then it is interesting to know if the random projection preserves Hamming distances of a projection from $\boldsymbol{Z}_2^n$ to $\boldsymbol{Z}_2^k$, since we already saw that it was preserving Euclidean distance from $\mathbb{R}^n$ to $\mathbb{R}^k$. Note that since all binary vectors and matrices are in the finite field of two elements ($GF(2)$), all operations are done modulo 2.

The process of random projection, given a vector $u \in \boldsymbol{Z}_2^n$, will yield a vector $v \in \boldsymbol{Z}_2^k$ such that :

$$v = R^T u$$

where R is the random projection matrix whose entries are independently picked as 1 with probability p and 0 with probability $1 - p$.

Since the Johnson-Lindenstrauss lemma cannot be applied directly in $\boldsymbol{Z}_2$, it has an analogous formulation, formalized in the following theorem.

[11]for details see [12]

Theorem 3. *(Random projection in $\mathbf{Z}_2$) Let $0 \le \epsilon \le \frac{1}{2}$ and $1 \le l \le n$. Let each entry of a $n \times k$ matrix R be chosen independantly to be 1 with probability $p = \dfrac{\epsilon^2}{l}$ and 0 otherwise. Let u,v be two points in $\mathbf{Z}_2^n$ and u', v' be obtained as*

$$u' = R^T u,\, v' = R^T v$$

There is a constant C such that with probability at least $1 - 2e^{-C\epsilon^4 k}$,

(a) *If* $|u-v|_H < \dfrac{l}{4}$, *then* $|u'-v'|_H < (1+\epsilon)kp\dfrac{l}{4}$

(b) *If* $\dfrac{l}{4} \le |u-v|_H < \dfrac{l}{2\epsilon}$, *then* $|u-v|_H < (1-\epsilon)kp \le \dfrac{|u'-v'|_H}{|u-v|_H} \le (1+\epsilon)kp$

(c) *If* $|u-v|_H > \dfrac{l}{2\epsilon}$, *then* $|u'-v'|_H > (1-\epsilon)kp\dfrac{l}{4}$

As we can see, this lemma is weaker than the original Johnson-Lindenstrauss lemma for the Euclidean case. Indeed, within a certain interval, distances are preserved within an ϵ-margin, but all smaller distances are bounded above, and bigger distances bounded below. In other words, small distances can be shrunked and big distances can be stretched.

2.2.3 Locality Sensitive Hashing

Locality-sensitive hashing (LSH) is another method of dimensionality reduction. It is aimed at hashing high-dimensional data in a way that preserves similarity with some probability. The technique was first introduced in [26] and improved in [21] with application to sublinear-time similarity searching. The concept is formalized with the following definition.

Definition 1. *(LSH Families) Assume that S is a metric space with distance function $d(.,.)$ and let $B(q,r) = \{p : d(q,p) \le r \text{ with } q,p \in S\}$ be the set of elements in S within distance r of q. A family $\mathcal{H}$ of functions $h : S \to U$ is said to be (r_1, r_2, p_1, p_2)-sensitive for the metric $d(.,.)$ if for h randomly chosen among $\mathcal{H}$ and for any $q,p \in S$*

$$(1)\ \textit{If}\ p \in B(q,r_1),\ \textit{then}\ \mathbb{P}_{\mathcal{H}}\left[h(q) = h(p)\right] \ge p_1$$
$$(2)\ \textit{If}\ p \notin B(q,r_2),\ \textit{then}\ \mathbb{P}_{\mathcal{H}}\left[h(q) = h(p)\right] \le p_2$$

In addition, a local-sensitive family is said to be useful when the following inequalities are satisfied : $p_1 > p_2$ and $r_1 < r_2$.

It is interesting to look at the special case where the distance function is the Hamming distance, i.e. $d(.,.) = d(.,.)_H$. In that case, vectors are in the Hamming hypercube. Then one can formulate the following proposition :

Proposition 1. *(Bit sampling with Hamming distance) Assume that S is the Hamming n-dimensional hypercube with the Hamming distance function $d(.,.) = d(.,.)_H$. Let $p = (p_1, p_2, \ldots, p_n) \in S$ and let $\mathcal{H}$ be the family of bit sampling functions, i.e. $\mathcal{H} = \{h_i : h_i(p) = p_i \forall i\}$. Then for any r, ϵ the family $\mathcal{H}$ is $(r, r(1+\epsilon), 1 - \frac{r}{n}, 1 - \frac{r(1+\epsilon)}{n})$-sensitive.*

The proof of this proposition can be found in [26].

2.3 Heuristic methods

For some problems which are hard to solve, direct and globally optimal methods may be either hard to find or hard to implement. In such cases, heuristic methods may be valid alternatives. They tend to find suboptimal solutions but are generally fast, or present a different kind of trade-off.

2.3.1 Greedy dimensionality reduction

Greedy algorithms are very simple and common kind of algorithm using heuristics. By definition, an algorithm is said to be greedy when it follows this general heuristic : each time where there is a choice to be made, the locally optimal option is chosen.

A greedy algorithm for distance preserving dimensionality reduction could be designed this way. Let us assume that the initial dimension is n and the target dimension is $k < n$. Instead of finding the best distance preserving dimensionality reduction from n to k, which would be globally optimal, we can use the following method : split the reduction into $n - k$ steps, and at each step choose the locally optimal solution. In other words, we start by reducing the dimension from n to $n - 1$, and then from $n - 1$ to $n - 2$, and so on so forth until k is reached.

2.3.2 Genetic Algorithms

Genetic Algorithms belong to the class of artificial evolutionary systems, and more precisely to evolutionary algorithms[17]. Evolutionary algorithms were inspired by the principles of natural evolution and molecular genetics in order to tackle hard optimization problems, find best class of programs or design circuits. These principles can be summarized as (1) population survival, (2) creation of diversity, (3) application of selection, and (4) genetic characteristics transmission.

Genetic algorithms are based on the following principles. Different solutions of a problem form a population. An individual is a specific solution to a problem and is called the phenotype and its genetic representation the genotype. The process of selection is the concept which makes optimization possible and can be summarized this way : the different phenotypes of the individuals are evaluated and given a quantitative value (generally referred to as fitness score), then under the process of reproduction, the genotypes corresponding to high fitness scores phenotypes are copied many more times than the one corresponding to low fitness scores phenotypes. The binary encoding of the individuals is one of the specificity of genetic algorithms. The general form of a genetic algorithm is iterative and follows these steps:

1. **Initialization** : a population of a certain size is created randomly (and if possible uniformly spread across the solution space).
2. **Evaluation** : The population is evaluated using a fitness function.
3. **Selection** : The individuals with the lowest fitness score are removed.
4. **Reproduction** : The remaining individuals are duplicated so as to refill the pool.
5. **Mutation and crossover** : Randomly selected individuals of the pool undergo modifications via genetic operators such as mutation and crossover .

The stages from (2) to (5) are repeated until a sufficiently good solution is found.

The selection phase is what makes this method heuristically based, whereas the mutation and crossover processes allow the method to step out of local optima. Genetic algorithms have proven to be very valid methods for hard problems such as the travelling salesman problem [29]. Genetic algorithms are well suited especially for the case of dimensionality reduction using sub-sampling. For example if we want to minimize a cost by projecting high-dimensional points from space of dimension n to dimension $k < n$. We have to search for the best sub-sampling of k coordinate among n, which generates a search space L of size $\binom{n}{k}$. Then we can use solutions in L as the population, with a binary encoding, and the cost as the fitness function.

3 Machine Learning

3.1 Purpose

Machine learning can be defined as the field of study at the intersection of computer science and statistics, aimed at creating programs and algorithms that can be said to learn. The fundamental aspect of machine learning is that the tasks are solved by the analysis of datasets. Machine learning can be separated into several different categories such as supervised learning, unsupervised learning, reinforcement learning, etc.

Machine learning techniques have been able to solve tasks which are extremely difficult or even impossible to solve using ordinary algorithms. Examples are numerous : face detection, spam filtering, speech recognition and brain computer interfaces to name a few.

3.2 Unsupervised learning

In the taxonomy of machine learning algorithms, we can find unsupervised learning among the main classes. Basically, the algorithm needs to learn a model of the input data without feedback about correctness of the model or previously known or desired output. Clustering techniques are good examples of unsupervised learning, also known as unsupervised classification analysis.

3.2.1 *k*-means

The method of *k*-means clustering (or simply *k*-means) takes unlabeled input data and gives as output a k-partition of the data. The k partitions, here called clusters, are disjoint and non-hierarchical. Thus, each data sample is labeled as belonging to one of the clusters. The partitioning tries to minimize the within cluster sum-of-squares objective function :

$$J(\mu_1, \ldots, \mu_k) = \sum_{j=1}^{k} \sum_{i \in C_j} |x_i - \mu_j|^2 \tag{5}$$

where $C_i, i = 1 \ldots k$ are the partitions (i.e. disjoints sets of data points) and $\mu_i, i = 1 \ldots k$ the centroids corresponding to each cluster C_i.

The *k*-means method is highly dependent of the initialization of the clusters and has no guarantee to converge to the global minimum of the objective J. The standard algorithm was first proposed by Lloyd [32] and has also been proposed by Forgy [18], which is why the algorithm is generally referred to as Lloyd's algorithm (or Lloyd-Forgy algorithm).

k-means algorithm (Lloyd-Forgy)

- **Initialization :** choose k centers by setting their centroids $\mu_1, \ldots, \mu_k$ to random values.

- **Assignment step :** assign each data sample to the closest cluster :

$$C_i^{(t)} = \left\{x_p : \left\|x_p - \mu_i^{(t)}\right\| \leq \left\|x_p - \mu_j^{(t)}\right\| \; \forall \; 1 \leq j \leq k\right\} \tag{6}$$

  where x_p is the sample to assign.

- **Update step :** compute the new centroid for each cluster from the current assignations :

$$\mu_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x_j \in C_i^{(t)}} x_j \tag{7}$$

- Alternate between assignment and update step until there is no more change in assignations.

Let us emphasize that this algorithm will converge, i.e. find a minimum, but it can be a local minimum of J. It is also very sensitive to the initialization step. Furthermore, the number of clusters k has to be given to the algorithm, so if the number of clusters if unknown *a priori* a different strategy has to be used.

Since the algorithm is quite fast, it is possible to tackle these problems by running the algorithm several times. Then a good way to compare different clustering solutions is to use the objective function J. For the initialization problem, one can run the algorithm multiple times, each time with different initial centers, and choose the instance which exhibits the lowest value of J. To find the good number of clusters, a simple strategy is to start with one cluster, apply the previous method to mitigate the initialization bias, increment the number of clusters and iterate until J is minimal (or after its decrease between iteration n and $n+1$ is under a certain threshold).

Another common problem of the k-means method is that it assumes that the covariance of the data is normalized, as it is in fact a special case of Expectation-Minimization (EM) using uncorrelated data. If it is not the case, one can compute the covariance matrix

$$\Sigma = \mathbb{E}\left[(X-\mu)(X-\mu)^T\right] \tag{8}$$

where X is the matrix of data samples and $\mu = \mathbb{E}[X]$.

And then normalize the data using the inverse of the covariance matrix

$$X' = X\Sigma^{-1/2} \tag{9}$$

For the initialization problem a new algorithm k-means++ has been proposed[5]. The method provides a distribution of initial cluster centers which allows the algorithm to converge quickly to a clustering with small error (with respect to the optimal clustering).

3.2.2 k-means in $\boldsymbol{Z}_2$

The k-means method presented in the previous section works in Euclidean spaces but need small modifications to be applied in $\boldsymbol{Z}_2$. In the assignment step, data samples are assigned to their closest cluster using Euclidean distance. For this step, it is sufficient to replace the metric by the Hamming distance. In the update step, the centroid is computed using the geometric mean of all points currently associated to the cluster. The closest equivalent to the geometric mean in $\boldsymbol{Z}_2$ is to take the component-wise median, which is equivalent to maximum voting for each component.

3.2.3 Hierarchical clustering

In k-means we saw that the algorithm was designed when the number k of clusters was known *a priori*, even if we discussed a method for unknown k using repetitions. Also, k-means provides a flat clustering, with no hierarchical information. Hierarchical clustering is a method which tackle these two issues.

Hierarchical clustering can be of two different types, it can be either agglomerative or divisive. In agglomerative clustering, the algorithm starts with each data sample as a single cluster and then iteratively merge clusters to form bigger clusters. In divisive clustering, the algorithm starts with all data considered as a big cluster and iteratively divides the cluster into smaller clusters. In both cases, the process continues until a certain number of clusters is found, or until an extremum of an objective function is reached.

To be able to select which subset of points will be merged or split a new proximity measure between subsets has to be introduced. This is generally called the linkage metric. Several kinds of linkage exists which can have a big impact on performance or quality of the clusters.

The most common linkages are :

- **Single Linkage** : The distance between two clusters C_i and C_j is the distance between the closest pair of points.

$$d(C_i, C_j) = \min_{n,m}\{d(p_{i,n}, p_{j,m})\} \text{ with } n \in [1, \ldots, |C_i|] \text{ and } m \in [1, \ldots, |C_j|] \quad (10)$$

- **Complete Linkage** : The distance between two clusters C_i and C_j is the distance between the farthest pair of points.

$$d(C_i, C_j) = \max_{n,m}\{d(p_{i,n}, p_{j,m})\} \text{ with } n \in [1, \ldots, |C_i|] \text{ and } m \in [1, \ldots, |C_j|] \quad (11)$$

- **Average Linkage** : The distance between two clusters C_i and C_j is the average distance between all pairs of points.

$$d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{n=1}^{|C_i|} \sum_{m=n}^{|C_j|} d(p_{i,n}, p_{j,m}) \quad (12)$$

The agglomerative algorithm works as follows :

1. **Initialization** : create as many clusters as the number of data points, and assign one point per cluster.
2. **Agglomeration step** : find the closest clusters according to the linkage metric and merge them taking either the mean or the median to define the new cluster center.
3. Iterate the agglomeration step until some criterion is fulfilled e.g. some minimal number of clusters is found or the maximal distance between clusters has reached a threshold.

And the divisive algorithm works as follows :

1. **Initialization** : create one cluster containing all points.
2. **Division step** : choose the cluster having the farthest pair of point or the maximum variance and split it into two clusters.
3. Iterate the division step until some criterion is fulfilled e.g. some maximal number of clusters is found or the within cluster maximal distance has reached a threshold.

The result of these algorithms is a hierarchical clustering which forms a tree diagram called a dendrogram. This hierarchical structure provides a way to examine the clustering at different scales.

3.3 Supervised learning

Another main type of machine learning algorithms is supervised learning. In short, a supervised learning algorithm will try to learn a model of the input data using a feedback about the correctness of the model or desired output. Classification tasks are very often solved using supervised learning algorithm.

3.3.1 Support Vector Machine

Support Vector Machines (SVMs) are very famous algorithms used for classification and regression problems. They belong to the class of sparse kernel machines and are maximum margin classifiers[12], meaning that they minimize the classification error and maximize the geometric margin at the same time.

In its original form [54], the SVM is a binary decision machine, meaning it supports only two classes and cannot provide posterior probabilities. The method was first proposed as a linear classifier, but was then extended to handle non-linearly separable problems with the use of kernel functions.

As other so-called kernel methods, SVM handle non-lineraly separable problems by projecting points in a higher dimensional space using a mapping $\phi : L \to S$ where L is the original space and S the higher-dimensional space. The goal is that by applying $x' = \phi(x)$ we get a linearly separable problem in L.

However, finding ϕ can be very difficult. To solve this issue, kernel methods rely on a principle called the kernel trick. It relies on the observation that the only operation that we want to perform in the high-dimensional space is computing inner-products. We thus only need to find a function allowing to compute inner-products in L without defining the mapping ϕ explicitly. This function $k(.,.)$ is called the kernel function and must respect a few properties.

One of the most important is that the kernel functions need to respect Mercer's condition. That is,

$$\iint k(x,y)g(x)g(y)\,dxdy \geq 0. \tag{13}$$

for all square integrable functions $g(x)$.

Also most kernel functions assume that the high dimensional space L is a Reproducing Kernel Hilbert Space. This is often referred to as the reproducing property of the kernel. Simply put, it states that the evaluation of the inner product of two functions in L yields a function in L.

The most used kernels are :

- Polynomial kernels : $k(\mathbf{x_1}, \mathbf{x_2}) = (\gamma \mathbf{x_1}^T \mathbf{x_2} + c)^p$, with $\gamma > 0$, $c \in \mathbb{R}$ and $p \in \mathbb{N}$.
- Hyperbolic Tangent kernels : $k(\mathbf{x_1}, x_2) = \tanh(\gamma \mathbf{x_1}^T \mathbf{x_2} + c)$, with $\gamma > 0$ and $c \in \mathbb{R}$.
- Gaussian Radial Basis Function kernels : $k(\mathbf{x_1}, \mathbf{x_2}) = e^{(-\gamma \|\mathbf{x_1} - \mathbf{x_2}\|^2)}$, with $\gamma > 0$.

Many variations of SVMs exist and a good review can be found in [8].

[12]Current implementations use soft margins [11].

3.3.2 k-Nearest Neighbours

The k-Nearest Neighbours (kNN) method is a very simple method generally used for classification. The learning step is in fact virtually inexistant since it consists only of keeping the list of training samples and their associated labels in memory.

The classification of a new sample is then done by applying a majority voting of its k nearest neighbours in the feature space. In other words, the label of the new sample is computed as the label having the highest occurance in its neighbors.

Although simple, the kNN can provide very good results in classification tasks. It has however several drawbacks. The first one is that, counter to most of the learning algorithms, the learning step is very quick, but the classification of new samples can be time consuming. Indeed, the search of nearest neighbors in high-dimensional space has a relatively high complexity, in function of the number of elements in the space.

As we see, the kNN method does not scale well, and becomes impossible to use for very large scale datasets both in terms of storage of the training set and in terms of classification query time. The complexity of the nearest neighbor search can be mitigated using special datastructures such as k-d trees.

The method can also suffer from a non-uniform distribution in the number of training samples per category. Indeed, if some category is over represented with respect to the others, it implies a bias in the majority voting.

3.3.3 Gradient Boosted Trees

Gradient Boosted Trees (GBT) is a method which applies gradient boosting to simple decision trees ; it was introduced in [19] and improved in [20]. It is mainly designed to solve regression problems, but can solve classification problems too using specifically designed loss functions.

The method of gradient boosting is a generalization of standard boosting technique to arbitrary loss functions. As with boosting, the goal is to combine weak learners together to create a strong learner. It is an iterative method which will create a model of improving precision. The refinements of the models are done with an iterative re-weighting of the loss function (similarly as the re-weighting of incorrectly classified samples in standard boosting).

Based on this principle the GBT algorithm tries to solve a solution to a regression problem of input set T and of loss function $\mathcal{L}(y, f)$ by constructing a regression model $f(x)$ iteratively. Each iteration is greedy and creates a single regression tree computed using the gradient of the loss function computed for the current state of the model. After n iterations, the model is :

$$f(x) = f_0 + \nu \sum_{i=1}^{n} t_i(x) \tag{14}$$

where $t_i(x)$ is the result of the regression tree i and $\nu \in [0, 1]$ the shrinkage parameter.

For regression problems, simple loss functions can be used, such as squared or absolute loss (i.e. $\mathcal{L}(y, f) = \frac{1}{2}(y - f(x))^2$ or $\mathcal{L}(y, f) = |y - f(x)|$). But since the gradient boosting works for any loss function one can also use functions which may allow to solve classification.

Indeed, let us assume that we have a k-class classification problem to solve. Now, let us associate loss function f_i to each class c_i, then we can compute the general loss function as :

$$\mathcal{L}(y, f_1, \ldots, f_k) = -\sum_{i=1}^{k} \mathbb{1}(y = k) \ln \left(\frac{\mathbb{E}[f_k(x)]}{\sum_{i=1}^{k} \mathbb{E}[f_i(x)]} \right) \tag{15}$$

Using the GBT algorithm with such a loss function, one can do classification by computing the predicted class $\hat{k}$ of a sample x as:

$$\hat{k} = \max_i f_i(x) \ , \forall i \in [1, \ldots, k] \tag{16}$$

Soon after the first method was proposed, its author proposed a small improvement called Stochastic Gradient Boosting that was designed to avoid overfitting. In essence the modification consisted in taking a random subset of the training data to train the weak regression tree instead of the full training set.

3.3.4 Bag of Words model

The term Bag of Words (BoW) comes from linguistics and refers to a way of representing language as a bag of words. More specifically it was first used in natural language processing to give a simple representation of language.

The general method to use a BoW model is to first construct a vocabulary. It can be constructed with different goals, but should essentially be discriminative. It can be constructed by agglomeration of many texts. It should obviously not contain very common words (such as "it", "and", etc.) and not be too large. Then the BoW representation associated to a vocabulary V of a text t is a normalized sparse histogram of words in V constructed using the words in t, i.e. a vector of occurrences.

The model in itself is not useful but is used as a preprocessing step for various algorithms. A simple example is the Spam filtering by Bayesian filtering. The method works as follows :

1. Construct two BoW ; one created from spam emails and the other from normal emails.
2. Train a naive Bayesian classifier using these two classes of samples
3. Classify a new sample x by first computing its two BoW representation and then let the classifier predict its class.

Bag of Visual Words

The BoW model has been extended out of language processing and have proved useful especially in computer vision. In this context, this model is sometimes called Bag of Visual Words (BoVW). The BoVW model has been used in object retrieval [38], mobile visual search [51], scene categorization [16] and content based image indexing and retrieval [40].

The general method to create a BoVW model is derived from the initial BoW model. That is, first a vocabulary needs to be constructed. This is generally done using feature-based representation of images. The general method to create a vocabulary (also called codebook) of size n is the following :

- Detect and extract features from a set of images.
- Apply a clustering method to find k representatives in the feature space.
- The set containing the k representatives is defined as the vocabulary.

Recently, hierarchical clustering has been used instead of flat clustering and have been shown to yield to good performance. In this context, the codebook is sometimes called a Vocabulary Tree.

The BoVW representation of an image I given a vocabulary V is then created using the following steps :

- Detect and extract features in I using the same detectors and descriptors than the one used to create the vocabulary.
- Find the visual words (also called codewords) corresponding to each feature by assigning its closest codeword in V using a nearest neighbour search.
- Create a sparse histogram of the codewords in I with respect to the codebook V.
- The normalized sparse histogram is the BoVW representation of I.

As for text-based BoW, this representation is not really useful in itself but is used as preprocessing before using learning algorithms. In fact this method could also be considered as dimensionality reduction, as it projects k high-dimensional vectors (i.e. descriptors) to a unique vector of occurrence of fixed size $|V|$.

Bag of Hash Bits

A variation of the BoVW called Bag of Hash Bits (BoHB) model has been proposed in [24]. In this work it is proposed to use LSH functions to map descriptors directly to words (or in this case to hash bits). This methods has two advantages over the BoVW method. First, there is no need to create a vocabulary which has a high computational cost if the desired size of the codebook is big and also if the training set contains many samples. The second advantage is to simplify the mapping between a descriptor and its corresponding codeword, avoiding the nearest neighbour search in the codebook.

Part II
Contributions and Applications

4 FREAK pairs selection

4.1 Dimensionality reduction

4.1.1 Motivation

The FREAK descriptor, as we saw previously, computes values based on difference of intensity between several receptive fields. There are thus $\frac{n(n-1)}{2}$ possible pairs[13] for n receptive fields. Even for a few receptive fields, since there is $O(n^2)$ corresponding pairs, the vector of the descriptor can be considered as living in a high-dimensional space.

This kind of binary descriptor is designed to be fast to compute, even on mobile devices, which is why they are represented in compact integer format. Also they are most of the time used in feature-based image representations. In this setup, several tasks will need fast nearest neighbour search (e.g. panorama stitching, visual search) and other will imply the use of machine learning algorithms (e.g. image classification) which are very sensitive to the dimensionality of the data. There is thus a need to reduce the dimensionality while preserving the information contained in the descriptor.

A descriptor can be used in two ways, either by directly using the information contained in the vector, or to match it to other descriptors. Matching is clearly the most frequent use of descriptors and is based on comparisons by distance. A descriptor q is said to match a descriptor $p \in T$, where T is a set containing multiple descriptors, if $d(q,p) < d(q,p') \; \forall p' \in T, p' \neq p$, with $d(.,.)$ is a distance between descriptors, and more precisely the Hamming distance in the case of binary descriptors.

This is then a perfect case to apply distance preserving dimensionality reduction which would allow to solve descriptors matching in the low dimensional space almost as well as in the high dimensional space.

4.1.2 Distance preserving dimensionality reduction

In section 2 of this work we saw multiple techniques to reduce the dimensionality while preserving good pairwise distances. To be able to evaluate the quality of the dimensionality reduction, let us define the objective function as the mean of the error between the distance in the low-dimensional space and the distance in the original high-dimensional space.

Definition 2. *(l^p projection error metric) Let us have $S \subset \mathbf{Z}_2^n$ a set of binary descriptors and $L \subset \mathbf{Z}_2^k$ the set corresponding to the projection of all elements from S to L using some function $f : S \to L$. Assume that $k < n$. Then the l^p projection error metric is defined as*

$$E_p(f) = \frac{2}{|S|(|S|-1)} \sum_{u,v \in S, u \neq v} \left\| \frac{n}{k} |f(u) - f(v)|_H - |u-v|_H \right\|_p \tag{17}$$

where $\|.\|_p$ is the l^p-norm and $|.|_H$ the Hamming distance.

Formulated as an optimization problem, the dimensionality reduction task consists in finding the best projection function $f^\star$ minimizing the projection error $f^\star = \min_f E_p(f)$ with $f : S \to L$. To simplify the problem, one can restrict f to the class of linear functions, i.e. $f(u) = R^T u$ where R is the $n \times k$ projection matrix.

Ideally, f should even consist only of pair sampling[14] (i.e. pair selection) and thus avoid linear combination of pairs. This way, the description extraction phase can be done

[13] $\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$

[14] more formally, the projection matrix R has exactly one non-zero value per column.

quicker since there is no need to compute the values corresponding to the non-selected pairs neither to project via matrix multiplication. This extra constraint tends to discard random projection techniques, as there is no possibility to control the construction of the projection matrix, whose entries are selected randomly.

This process of dimensionality reduction was applied to the problem of FREAK pairs selection. In the current version of FREAK, a subset of 512 default best pairs from the 903 possible pairs have been selected using pairwise variance maximization and decorrelated selection.

The goal of this first contribution is to find the 512 pairs which best approximate distance between full descriptors (i.e. 903 dimensional descriptors). The objective functions to minimize were either the l^1 projection error metric (E_1) or the l^2 projection error metric (E_2). These functions have been evaluated on a set of more than 10^6 descriptors coming from random images[15].

Evaluation set

To avoid evaluating the error on the set D_p containing all the $O(10^{12})$ pairs of descriptors, a small subset $S_p \subset D_p$ of all descriptor pairs is created. But instead of taking the element of the subset at random, the goal is to devise a way of creating a subset with a good entropy of descriptors. The proposed method to construct the subset is the following :

Given a set of descriptors D, an initially empty subset $S \subset D$ and a target size m.

- Select a descriptor d_i randomly by picking $i \in U[0, |D|]$.
- Let $S' = S \cup \{d_i\}$. If $H(S') \geq H(S) - \dfrac{\epsilon}{|S|}$ then add d_i to S.
- Repeat the procedure until $|S| = \lfloor \dfrac{\sqrt{1+8m}+1}{2} \rfloor$[16].

where $U[0, |D|]$ is the discrete uniform random variable and $H(S)$ is the entropy of the subset computed as

$$H(S) = \sum_{i=0}^{n} H_b(S_i) = -\sum_{i=0}^{n} \sum_{j \in \{0,1\}} \mathbb{P}[S_i = j] \log_2(\mathbb{P}[S_i = j]) \tag{18}$$

with S_i is the vector formed by all the i-th element of all descriptors in S, and

$$\mathbb{P}[S_n = j] = \frac{1}{|S_n|} \sum_{i=0}^{|S_n|} \mathbb{1}(S_n[i] = j) \tag{19}$$

where $\mathbb{1}$ is the indicator function.

Using this method the set of descriptors S has a good entropy, which means that S is a subset of D that has a good variety of descriptors. The subset of pairs S_p is then created by exhaustive enumeration of the pairs in the set of descriptors S and is guaranteed to have a size very close to the target size m.

[15] creative commons images from flickr.com

[16] it comes from the fact that an exhaustive enumeration of all non identical pairs of S gives a set of size $m = \frac{|S|(|S|-1)}{2}$

Genetic algorithm

As one of the methods used to solve the dimensionality reduction problem we implemented a genetic algorithm. Indeed, if the problem to solve is to find a good pair selection from the set of all pairs, we have a combinatorial number of possibilities, more specifically $\binom{903}{512} \approx 5.32 \cdot 10^{266}$ different possible pair selection.

In the theoretical part we saw the generic approach of genetic algorithms applied to dimensionality reduction by fixed sub-sampling. More specifically, the goal of this implementation is to find the best subset of 512 pairs from the 903 available.

At first, a genetic representation is needed. A simple yet effective one consists in defining the genotype as a binary vector in $\boldsymbol{Z}_2^{903}$ where each 1 means that the corresponding pair index is sampled, with a Hamming weight of 512. Unfortunately this representation is problematic since crossover and mutation operators can yield genotypes of different Hamming weight, and would need an artificial refining.

A simple solution based on the previous one is to define the genotype as a list of 512 integers corresponding to pairs indices. In this case the phenotype is defined as the projection function corresponding to the sub-sampling of the pairs indices encoded by the genotype.

The fitness function can be easily chosen to be $-E_p$, as we want to minimize E_p and the selection process tends to maximize the fitness function.

Having defined both the genetic representation and the fitness function, we can describe the genetic algorithm :

1. **Initialization** : the population is filled with individuals picked uniformly at random from the solution space, until the pool has size N. Also an evaluation set S_p is created using the method described above.

2. **Evalutation** : each individual g_i in the pool with corresponding phenotype f_i is given a quantitative score s_i using $s_i = -E_p(f_i)$.

3. **Selection** : all the K best individuals (i.e. having the highest fitness score) are selected with probability p_{bs}, and all the remaining ones are selected with probability p_{rs}.

4. **Reproduction** : selected individuals are duplicated randomly in order to get back a pool of size N. Individuals with a higher fitness score have a higher probability of being duplicated.

5. **Crossover** : All individuals issued of the reproduction process undergo a random number of crossovers n_{co} determined by a Poisson distribution of parameter λ_{co}.

6. **Mutation** : A number $n_{mutants}$ of mutants are chosen in the pool. The number $n_{mutants}$ is picked randomly according to a Poisson distribution of parameter $\lambda_{mutants}$. The mutants are altered in $n_{mutations}$ positions, where the number is determined by a Poisson distribution of parameter $\lambda_{mutations}$.

The steps (2) to (6) are repeated iteratively a number of time n_{iter}.

The parameters used for this algorithm were the following : $N = 500$, $K = 0.3 \cdot N$, $p_{bs} = 0.95$, $p_{rs} = 0.2$, $\lambda_{co} = 5$, $\lambda_{mutants} = 0.05 \cdot N$, $\lambda_{mutations} = 16$ and $n_{iter} = 3000$. The sampling of the Poisson distributions has been implemented using the inverse transform method.

Greedy dimensionality reduction

To find a subset of pairs index, a simple and very efficient greedy dimensionality reduction algorithm can be designed. As described in the theoretical part, such a method will decompose the dimensionality reduction from n to k into $n-k$ steps. At each step, the goal is to find the subset of pairs which minimizes the E_p metric.

The proposed algorithm follows this sequence of steps :

1. Initialize the subset of pairs P_c as the full set of pairs of size n. The goal is to iteratively remove pairs to have $|P_c| = k$ with $k < n$.
2. Enumerate all possible subsets $P_n \subset P_c$ of size $|P_c| - 1$, and compute the E_p metric for each one.
3. Set the new P_c as the P_n which had the minimal value of E_p. If $|P_c| = k$ exit ; else go back to step (2).

This algorithm will always converge in $\dfrac{n(n+1) - k(k+1)}{2}$ evaluations of the metric E_p.

Random projection

As we said above, random projection methods are not guaranteed to yield pairs sampling projections, since they provide linear projection in $\boldsymbol{Z}_2$. But since the random projection could be better than pairs sampling methods, it is worth a deeper evaluation.

As we already saw, the theorem is weaker than its analogous formulation in $\mathbb{R}$ and we must thus define a range in which the distances are preserved. For the theorem to be applicable we need to choose the parameters l and ϵ. This choice is crucial as it determines the range where the projection is distance preserving.

The range is defined as $r = [\frac{l}{4}; \frac{l}{2\epsilon}]$ and thus its size can be computed as

$$s_r = \frac{l}{2\epsilon} - \frac{l}{4} \tag{20}$$

But we also want to have the constraint $kp = 1$ so that distances are not distorted. Thus :

$$kp = k\frac{\epsilon^2}{l} = 1 \iff k = \frac{l}{\epsilon^2} \tag{21}$$

Then to find the biggest range we need to solve the following optimization problem. Maximize s_r with $k = \dfrac{l}{\epsilon^2}$, $\epsilon \in [0, \frac{1}{2}]$ and $l \in [1, n]$.

The solution to this problem with $n = 903$ and $k = 512$ is $\epsilon = 0.5$ and $l = 128$, which produces a range $r = [32, 128]$ of size $s_r = 96$.

The first remark is that the range r is applied to Hamming distances in $\boldsymbol{Z}_2^{903}$, so the real range of distances is $[0, 903]$. This implies that the widest range where distances are preserved by random projection covers actually only $\frac{96}{903} \approx 10\%$ of all possible distances.

This problem could be mitigated if we knew a priori that most distances were in this specific range, but we have no prior distribution of the descriptors or their distances. So we can only conclude that due to the weakness of random projection in $\boldsymbol{Z}_2$, we cannot use it for this specific application.

This does not imply that random projection in $\boldsymbol{Z}_2$ is not useful in general. In fact, it has been proved to yield interesting results for Approximate Nearest Neighbours search. The authors of [55] mitigated the problem due to the short range of non-distorted projections by using multiple projections. Although interesting, this would not be applicable to our problem as storing only one matrix is already not desirable.

Results

We can see in Figure 1 the error measured using E_1 and E_2 metric of the two subsets of pairs computed using dimensionality reduction methods, a random set of pairs and the default subset of pairs of FREAK. First, we can see that dimensionality reduction techniques work yielding pairs with the smallest error. Then we can see that the default pairs have a very large error even compared to random pairs. This question is addressed in the next section.

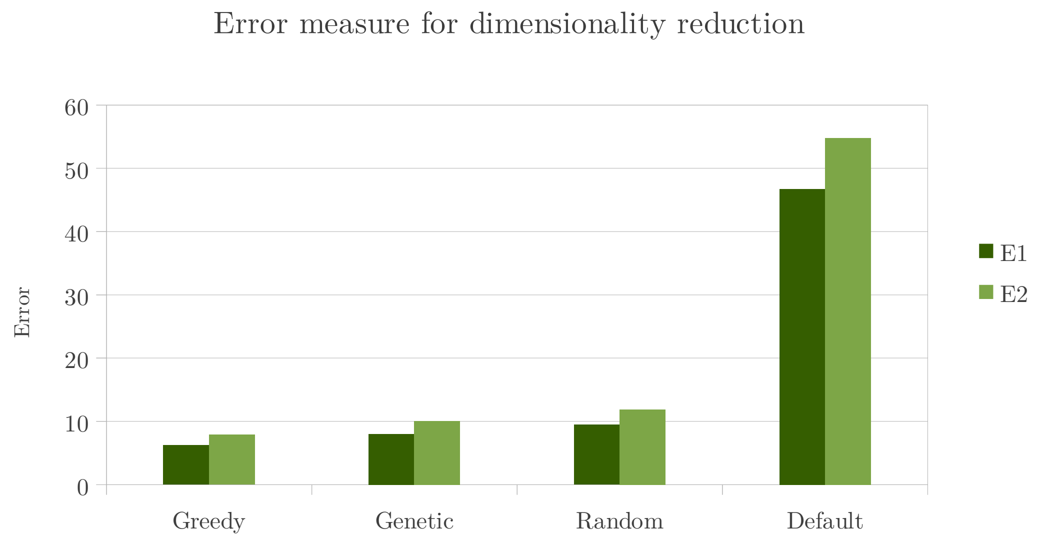

Figure 1: This graph shows the errors in E_1 and E_2 metrics of subsets of 512 pairs obtained using three dimensionality reduction techniques and using the default pairs. The labels are the different methods and the default pairs.

4.1.3 Correlation bias

As we saw, the proposed methods of distance preserving dimensionality reduction are effective, and yield subset of pairs with low distortion with respect to the full set of pairs. Also, the results show that the default pairs for FREAK perform very poorly in both E_1, E_2 metrics. In fact they are even worse than random.

The explanation comes from the fact that the metrics used for dimensionality reduction make an implicit hypothesis which is not valid in practice. Indeed, the proposed metrics E_p assume that all the information contained in the high-dimensional vector is relevant. Unfortunately, this is not the case for the specific problem of FREAK pairs selection.

As it is already stated in the original paper, "many of the pairs might not be useful to efficiently describe an image"[1]. Indeed, if we look at the FREAK pattern we can note its strong symmetry which implies a potentially high correlation between some pairs. The correlation is in fact the justification of the pairs selection in the work of [1]. This is why the authors propose a method which produces a decorrelated set of high-variance pairs.

Let us make the claim stronger, the correlation between pairs is not only a source of inefficiency, i.e. not contributing positively to the descriptors, but can have a negative contribution. Indeed, there is a possibility that the power of description of the descriptors is not linearly related to the number of pairs used to create them. For example, it may be better to get a specific structured set rather than the full set.

If so, then some pairs can be considered as noise and must be discarded from the descriptors. To be more specific, this is not necessarily some pairs which always perform bad, but pairs relative to a subset of pairs. From now on, let us refer to this phenomenon as the correlation bias.

Since we made a stronger claim about the correlation between pairs, let us devise an experiment to show it more clearly. A standard measure used to assess the quality of a descriptor is the recall vs. precision test. And the authors of [37] have proposed a framework, widely used, to perform this kind of descriptors performance evaluation.

Let us then use this practical evaluation to see if it gives some support to the claim. The selection of 512 pairs coming from the pairs sampling dimensionality reduction techniques presented above, along with original default pairs, random pairs and full 903 pairs are compared using a recall vs. 1−precision measure.

Results

As we can observe in Figure 2 that all the pairs performing well in the E_p metric yield similar recall precision curves, but they are also similar to the random pairs, which is not a very good point. Also, the default pairs, while having a very bad result in the E_p metric, outperforms quite systematically the three other sets of pairs.

These results seem to confirm the problems linked to the correlation bias. In the next section we will try to find a solution which solves the problem of the correlation bias, while keeping in mind an optimization using distance preserving dimensionality reduction.

4.1.4 Pairs selection

Since the dimensionality reduction using E_p metric seem to provide a low practical interest in itself for the problem of FREAK pairs selection, we can see how it can be combined with other heuristics to make it really useful in practice.

As a starting point let us first recall the pairs selection method proposed in [1] and formalize it a bit. From the paper and the current implementation we can summarize the algorithm :

1. Gather a matrix D containing full descriptors of keypoints detected on a dataset of images.
2. On each column D_i of D (i.e. pair-wise) compute the linear deviation from the maximum mean (i.e. $s_i = 0.5 - \mathbb{E}[D_i]$)
3. Reorder the columns of D so as to sort them in decreasing s_i order.
4. Then select the pairs starting with the first column and enumerating in order, adding them if they do not exceed a given threshold of correlation with the already selected pairs.

In order to give a higher level view, we can split the algorithm into two distinct stages :

- **Scoring** : compute a score for each pair and order them accordingly.
- **Sampling** : run through the pairs in scoring order so as to avoid the correlation bias.

The next contribution of this work is to elaborate on this more general method. The goal is to have both a good scoring and sampling schemes.

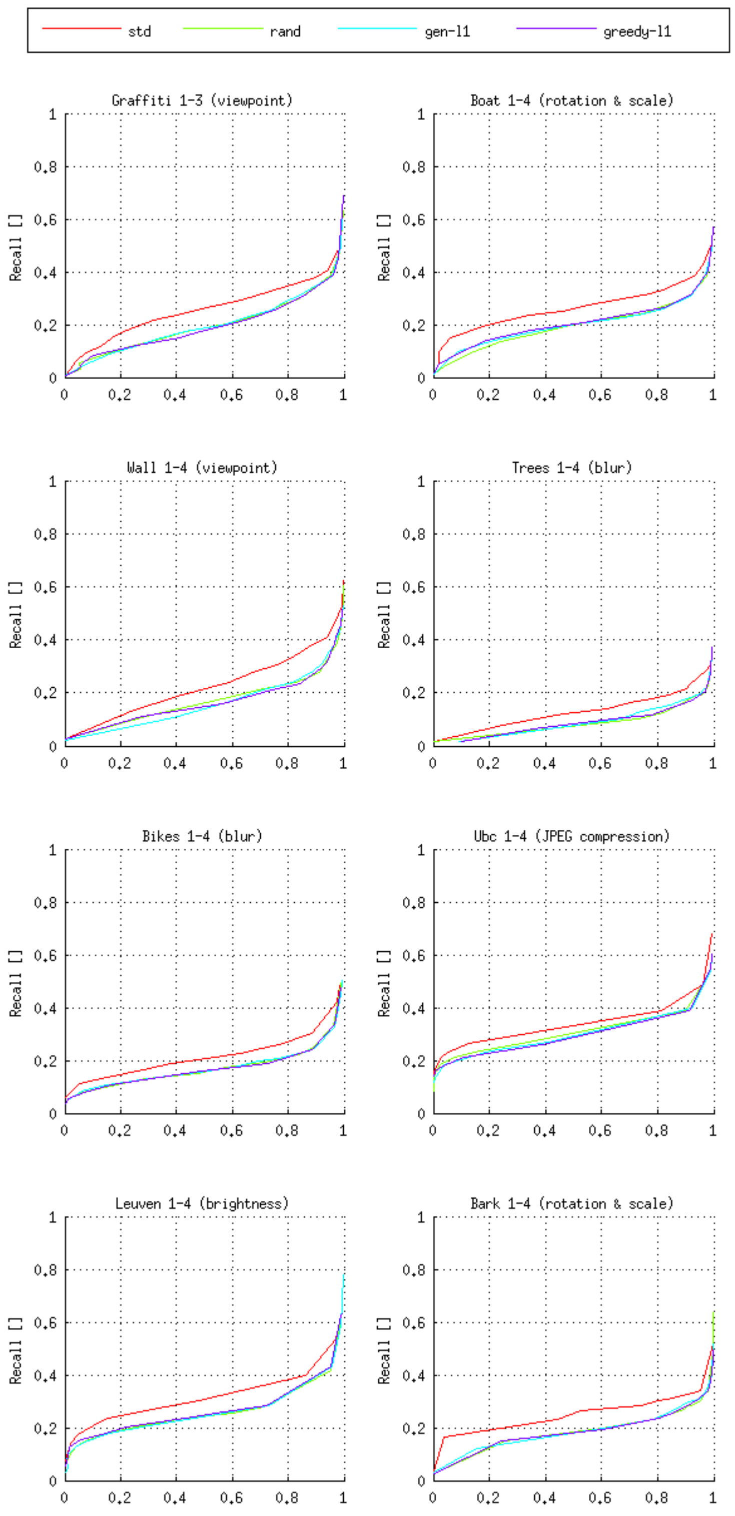

Figure 2: These plots show the resulting recall precision curves obtained using the evaluation framework provided by Mikolajczyk and Schmid [37] using pairs coming from dimensionality reduction techniques, random and default pairs.

4.1.5 Scoring

A scoring scheme can be defined as a value associated to each individual pair of receptive field. More formally, let us give the following definition. The full-size freak descriptor corresponds to a vector $d \in \mathbb{Z}_2^{903}$. A score is thus a vector $s \in \mathbb{R}^{903}$. Let us define that the higher the score is, the better the pair is evaluated.

The first idea is to reuse some methods of the dimensionality reduction part to compute scores. However, most of them are only designed to search a specific combinatorial space, and cannot yield a pairwise score. The genetic algorithm falls in this category, and the random projection method does not evaluate the pairs.

The greedy method however can be adapted to provide a score. Indeed one can avoid to stop the greedy reduction when the target dimension is reached and let it go through all pairs. The score is then defined in the following manner : set the E_p norm of the current iteration as the score corresponding to the removed pair. Such a score thus corresponds to the tendency to preserve high-dimensional distances.

Instead of looking at distance preserving pairs, one can look at simple statistical criteria to compute scores. We thus look at pairs having a strong discriminant power.

Variance

In [1], the proposed criterion is based on variance estimation. The method was described above and need a matrix of full descriptors D based on detection and description of keypoints in a dataset of images. As it is often the case with statistical methods, one must be especially careful about the creation of the dataset. Indeed, the claim made in the paper is that a discriminant feature is associated to high-variance.

Entropy

Instead of computing a mean-based estimate of the variance, one can use the entropy. The scoring is very similar as the previous one, but the quantity computed is different. If D is a matrix with one descriptor per raw, and D_n the n-th column of D, then the score of the pair n is

$$s_n = H_b(D_n) = \sum_{i \in \{0,1\}} \mathbb{P}[D_n = i] \log_2(\mathbb{P}[D_n = i]) \tag{22}$$

where $\mathbb{P}[D_n = k]$ is computed as before.

Note that the same comment as the one made about the variance apply about the design of the matrix D.

4.1.6 Sampling

The sampling scheme we propose is a refined version of the original sampling described in [1] and reminded above. The proposition is to avoid using a fixed threshold ρ_{max} given as a parameter but to view it as a parameter to optimize.

Indeed, the goal is to get pairs as few correlated as possible. We can thus try to find the minimal value of ρ_{max} for which the resulting set of pairs is of sufficient size. Given a matrix D of full size descriptors (one per raw), S a scoring scheme for the pairs, let us define the correlation between two rows D_n, D_k of D as :

$$\rho_{n,k} = \frac{\mathrm{Cov}[D_n, D_k]}{\sqrt{Var[D_n]}\sqrt{Var[D_k]}} = \frac{\sum_{i=0}^{|D_n|}(D_n(i) - \mathbb{E}[D_n])(D_k(i) - \mathbb{E}[D_k])}{\sqrt{\sum_{i=0}^{|D_n|}(D_n(i) - \mathbb{E}[D_n])^2}\sqrt{\sum_{i=0}^{|D_k|}(D_k(i) - \mathbb{E}[D_k])^2}}$$

where $\mathbb{E}[D_n] = \frac{1}{|D_n|}\sum_{i=0}^{|D_n|} D_n(i)$.

Let us now define P as the set of all pairs indexes, $P_s \subset P$ as the set of selected pairs indexes and $P_r = P \setminus P_s$ as the set or remaining pairs indexes. The optimization problem we have is thus :

Find the set P_s such that ρ_{max} is minimized with

$$\rho_{max} = \max|\rho_{n,k}|,\ \forall n, k \in P_s \tag{23}$$

and the following constraints :

1. $|P_s| = 512$
2. the pairs are added iteratively in the order given by S.

The algorithm proposed to solve this problem is iterative and is described below.

```
Data: A set $S$ of ordered pairs indices and a parameter $\epsilon$.
$\rho_{max} = 0.1$ and $\rho_{step} = 0.1$ ;
exit = false ;
while not exit do
    $S_c = S$ ;
    $P_s = \{\}$ ;
    while $|P_s| < 512$ and $S_c$ not empty do
        select the first pair index $s$ in $S_c$ ;
        remove $s$ from $S_c$ ;
        if $\rho_{n,s} < \rho_{max}, \forall n \in P_s$ then
            add $s$ to $P_s$ ;
        end
        if $|P_s| = 512$ then
            if $\rho_{step} > \epsilon$ then
                $\rho_{step} = \frac{\rho_{step}}{2}$ ;
                $\rho_{max} = \rho_{max} - \rho_{step}$ ;
            else
                exit = true ;
            end
            break ;
        end
        if $S_c$ is empty then
            $\rho_{max} = \rho_{max} + \rho_{step}$ ;
        end
    end
end
```

Algorithm 1: Optimally decorrelated sampling

A typical value for ϵ is 10^{-5}.

4.1.7 Results

Many sets of pairs have been generated for testing. The creation always used the following steps :

1. Compute one or more scores and order the pairs indices accordingly
2. Use the algorithm described above to get an optimally decorrelated set of pairs.

The scoring is based on three different methods, i.e. variance (var), entropy (ent) and greedy dimensionality reduction (dr). There is thus seven possible scoring schemes : 1-scores (var, ent, dr), 2-scores (var+ent, var+dr, ent+dr) and 3-score (var+ent+dr).

In Figure 3, the results using different pairs resulting from scoring including variance scoring are represented. We can see that all variance based curves are approximately the same (the var+ent being the best of them). They overall perform better than default pairs except for one case (Wall 1-4 viewpoint) where they have a worst result and two (Trees 1-4 blur and Bark 1-4 rotation and scale) where the yield similar performances.

In Figure 4, the results using different pairs resulting from scoring including entropy scoring are represented. In this case we can see that the dr+ent method has a different behaviour than the other ones. Indeed, dr+ent is either clearly better or slightly worst than the other entropy based pairs (from which var+ent is the best). One of the entropy based method is systematically better than the default pairs, either var+ent or dr+ent.

In Figure 5, the results using different pairs resulting from scoring including dimensionality reduction scoring are represented. There is not much more to say about the different curves present as dr alone does not reveal much, and the other cases have already be covered.

In Figure 6, we can see the results coming from all scoring consisting of only 1 method. We can observe a similar trend already seen above. Indeed, all 1-score pairs perform very similarly (with var being consistently the best). And default pairs are globally worse, perform best on Wall 1-4 and equally well on Trees 1-4 and Bark 1-4.

In Figure 7, the last of the series, we can observe all scoring consisting of 2 methods. Again, the best performance are achieved using either var+ent or dr+ent, with a slightly best average performance of the dr+ent method.

To summarize, the proposed method for pairs selection works since it provides pairs which outperform default pairs. We can also see that combination of two scoring methods is better than only one method and also than all of them together.

We can also assess the validity of all scoring methods, especially the entropy based which combines very well with other methods.

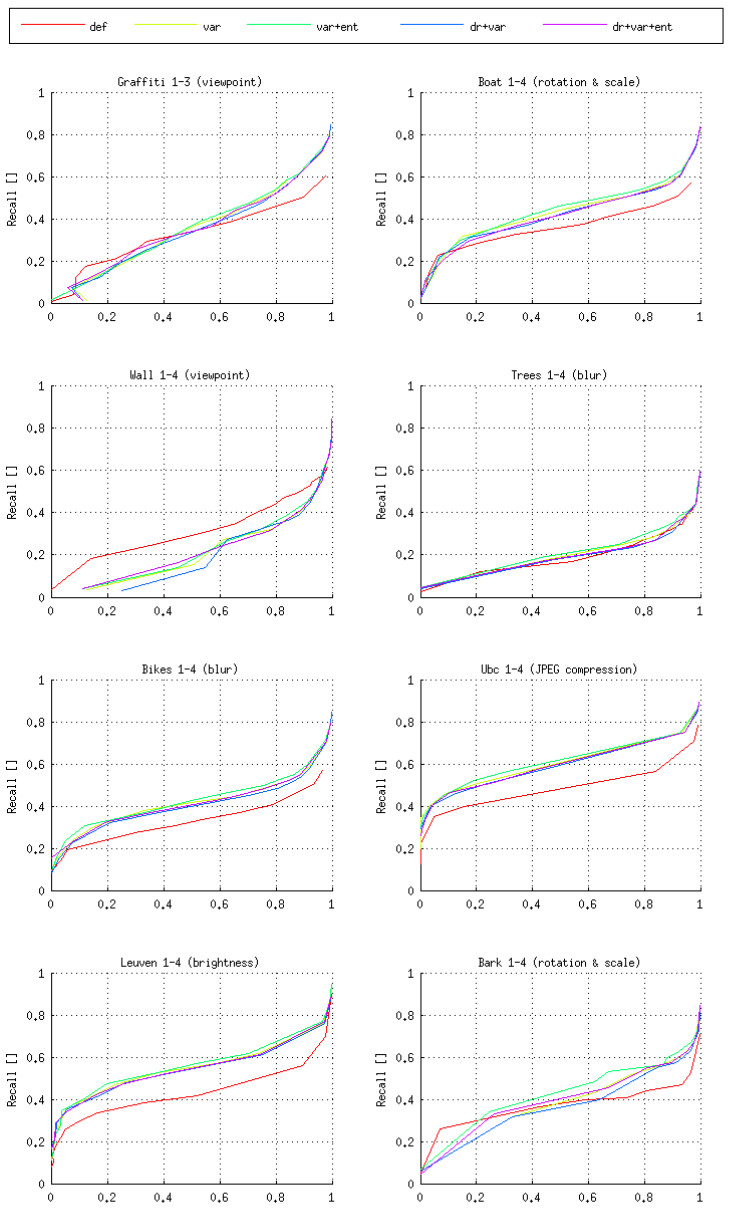

Figure 3: These plots show the resulting recall precision curves obtained using the evaluation framework provided by Mikolajczyk and Schmid [37] with pairs coming from variance-based scoring techniques and the default pairs.

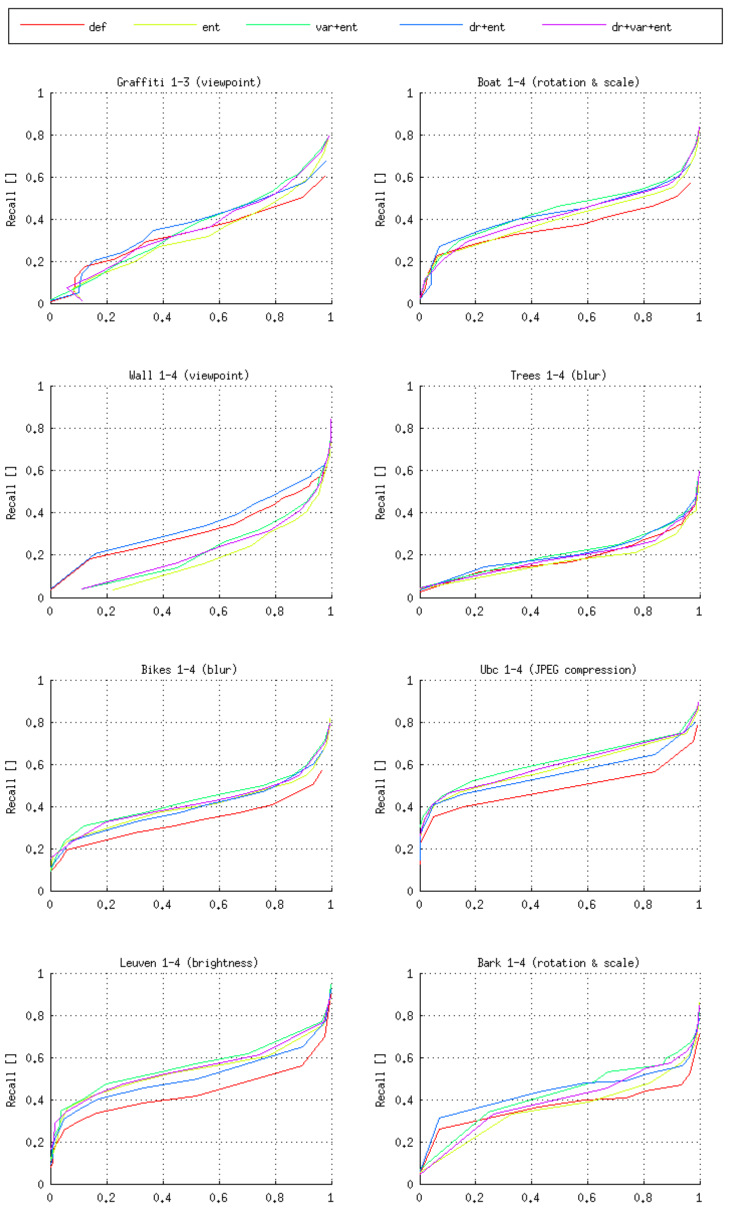

Figure 4: These plots show the resulting recall precision curves obtained using the evaluation framework provided by Mikolajczyk and Schmid [37] with pairs coming from entropy-based scoring techniques and the default pairs.

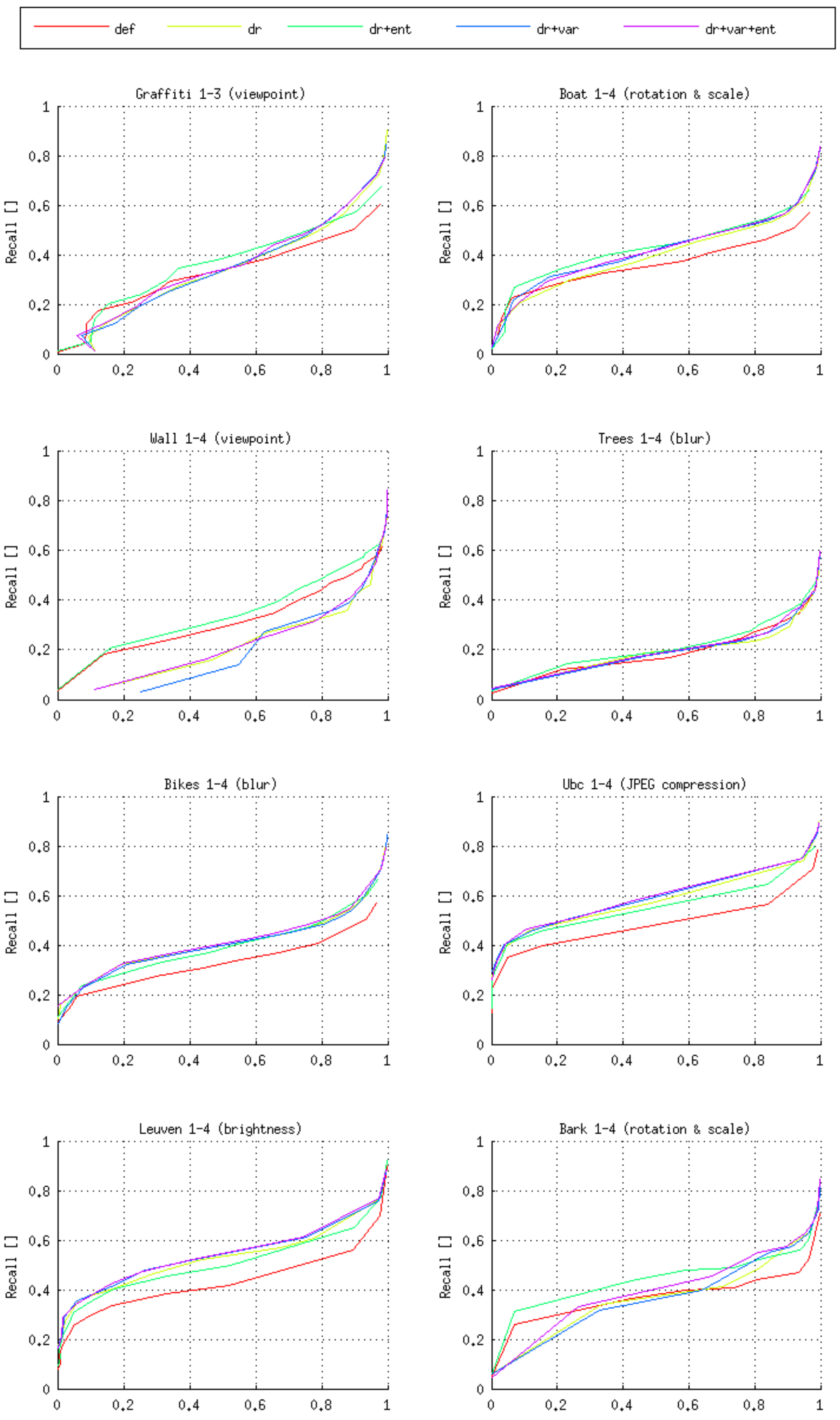

Figure 5: These plots show the resulting recall precision curves obtained using the evaluation framework provided by Mikolajczyk and Schmid [37] with pairs coming from dimensionality reduction based scoring techniques and the default pairs.

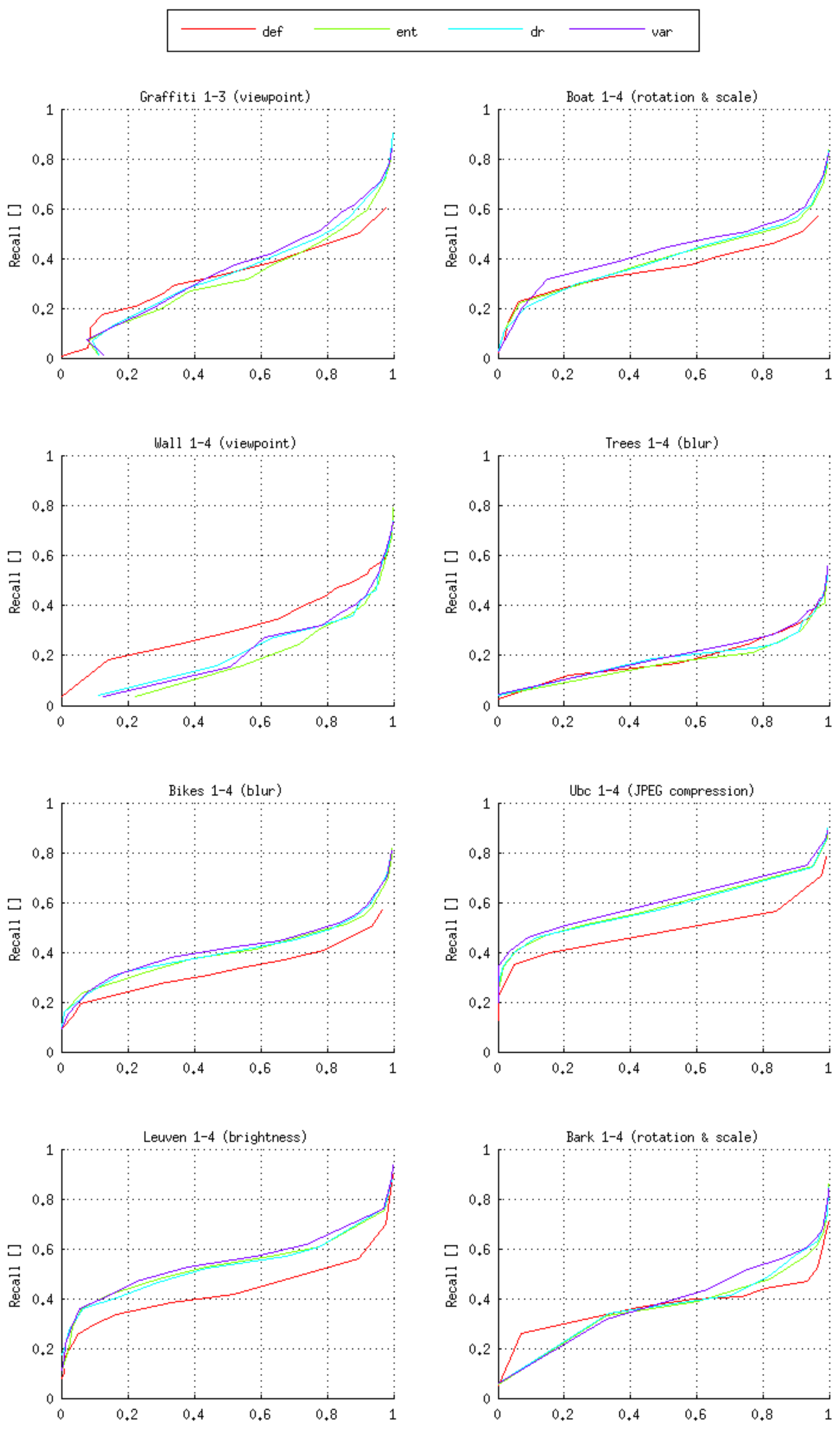

Figure 6: These plots show the resulting recall precision curves obtained using the evaluation framework provided by Mikolajczyk and Schmid [37] with pairs coming from only one scoring technique and the default pairs.

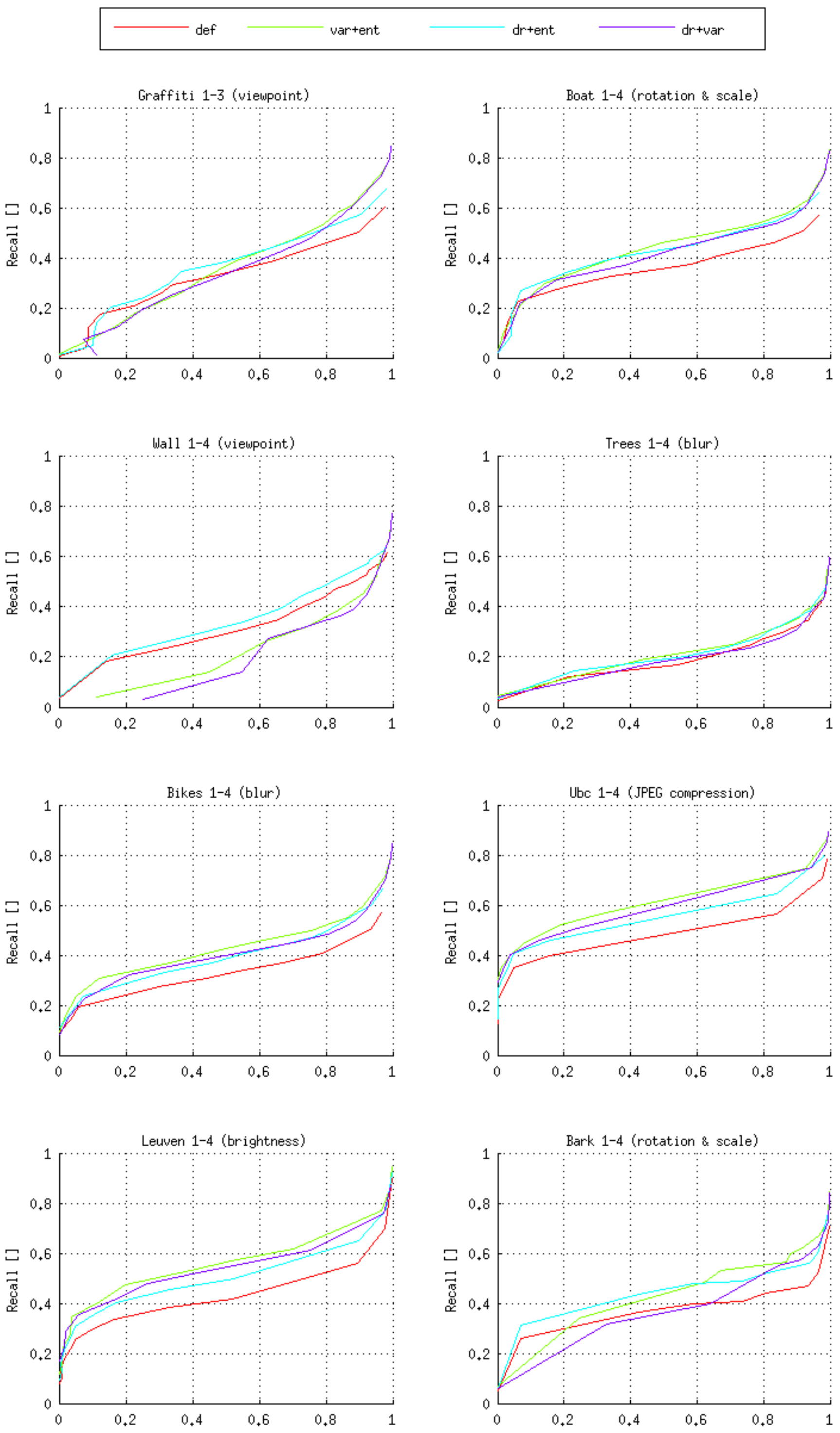

Figure 7: These plots show the resulting recall precision curves obtained using the evaluation framework provided by Mikolajczyk and Schmid [37] with pairs coming from the combination of two scoring techniques and the default pairs.

5 Image Classification

A typical application which can benefit of feature-based image description is image classification. The task of image classification, or object recognition, is a very important one and has a huge potential. The problem is however extremely difficult to solve.

Many datasets have been created as incentives to research in this particular area. Among many can we cite the Caltech-101[15] and Caltech-256[22] which are sets of 9104 (resp. 30608) non pre-conditioned images with 101 (resp. 256) object categories, the NORB dataset [30] containing 48600 images of 5 categories of toys taken under controlled and varying illumination and viewpoint conditions, or more recently the ImageNet database containing currently more than 14 millions images[13].

The authors of ImageNet also organized a challenge called Large Scale Visual Recognition Challenge, which is an opportunity for researchers to compare their work. A similar well-known challenge is the Pascal Visual Object Classes Challenge[14]. Both challenges had a 2012 edition.

As we can see, image classification is still a very current research topic in both machine learning and computer vision areas. We will not review here all the different techniques used to tackle this problem, but we can however make a distinction between two general approaches. Let us summarize the task of classification as

1. Find suitable preprocessing and apply it to the images.

2. Use a machine learning algorithm to classify the data.

Then the first approach will focus on the preprocessing step, while the other will focus on the design of the learning algorithm. In fact, the focus generally depends on the field of specialization of the researchers. So computer vision specialists will try to find very clever preprocessing techniques and then use relatively simple learning, while machine learning researchers will maybe apply simpler preprocessing and focusing more on very sophisticated learning methods.

The preprocessing step very important for the learning stage to perform well, especially for very high-dimensional data such as images as we know that most learning algorithm are very sensitive to the dimension of the input. This is why the feature-based image description is a very interesting model and has been used in many works of image classification such as [48][52][45][43].

It is worth noting that until now, and to the best of our knowledge, almost all results in image classification or visual search use the most famous real-valued descriptors such as SIFT and SURF[17]. The final contribution of this work is to show that binary descriptors can be very valuable alternatives to real-valued descriptors even for challenging tasks such as image classification[18].

5.1 Image classification with one descriptor

5.1.1 Setup

As a first application of binary descriptors used for image classification it can be interesting to measure their descriptive power. The tests have been done using the NORB database[30][19], briefly introduced above. The database contains five different categories

[17]we can refer to [51][24] [39] [38] [4]

[18]Another very different reason to find alternatives to SIFT and SURF is the fact that both are patented and thus not free of use.

[19]more precisely the "small" dataset

of objects. The images are in grey scales, taken with different lighting and different viewing conditions. Samples images from the dataset are shown in Figure 8 The images are dispatched between a training and a testing set, each containing 24300 images.

The goal is then to train a classifier using the training set and then to test its performance on the testing set. The general method of image classification has been applied so there is a preprocessing stage followed by a learning stage. In order to be able to measure the ability of the descriptors to describe an image sufficiently well for classification a "ground truth" measure has been defined.

5.1.2 Preprocessing stage

Ground truth

To be able to measure the capacity of the descriptors to extract meaningful information about the images we need to compare it to the full images. Since all the images have the same size, 96 by 96 pixels, which is not too big, it is possible to feed the learning algorithms directly with a high-dimensional vector representing the pixel values. Actually, this is one of the reason to choose the NORB dataset to perform this experiment, since otherwise a scale normalization and a possibly mandatory resizing would be needed.

In this raw form, the images are linearized row by row into a vector $x \in \boldsymbol{R}^{9216}$. A simple and standard preprocessing is then applied to x in order to normalize it (statistically speaking) :

$$x_n = \frac{x - \mu_x}{\sigma_x} \tag{24}$$

with $\mu_x = \mathbb{E}[x]$ and $\sigma_x = \sqrt{\mathbb{E}[x - \mu_x]}$.

The ground truth is thus the result of the learning and testing using this normalized vector form as input.

Descriptors representation

In order to provide a test framework unbiased by the detection, all the are given a unique keypoint located at the center of the images, whose scale diameter is the same as the image patch size. Each image is thus represented by only one descriptor.

The descriptors evaluated and their parameters are :

- the FREAK descriptor. A 512-bit (64 bytes integer) descriptor, using the default pairs, with a pattern scale of 6.2 and 3 octaves.

- the BRISK descriptor. A 512-bit (64 bytes integer) descriptor with a pattern scale of 0.5 and 3 octaves.

- the ORB descriptor. A 256-bit (32 bytes integer) descriptor with a pattern scale of 1.2 and the default parameters.

The parameters were chosen empirically as yielding the best performance.

The preprocessing step in the descriptor representation consist only of the extraction of the descriptor. No further preprocessing were done as it could have biased the results.

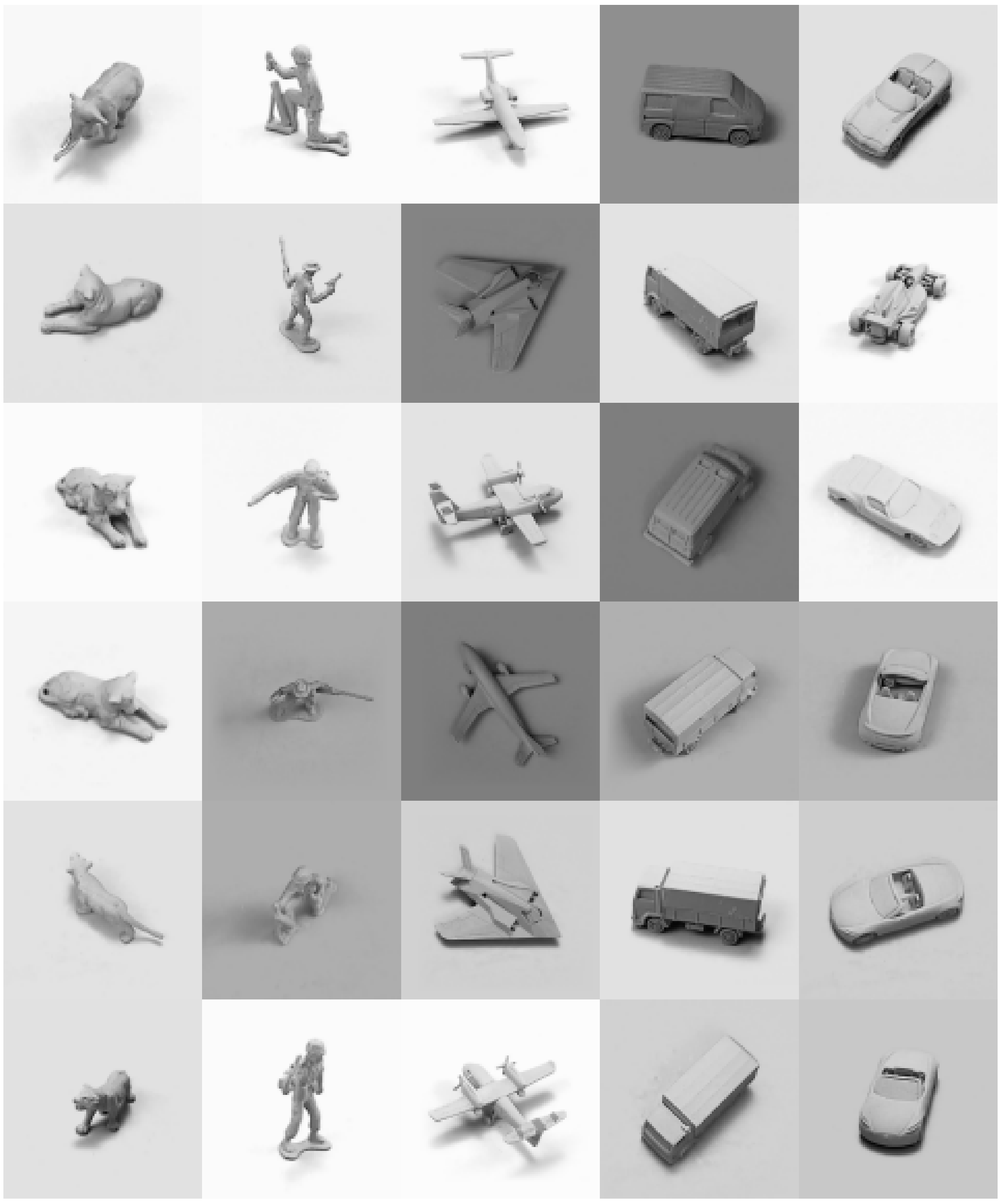

Figure 8: This is a small sample of NORB image database. Each column is associated to a sample of 6 images coming from one category. They are (from left to right) 0 - animal, 1 - human, 3 - plane, 4 - truck and 5 - car.

5.1.3 Learning stage

The training of the classifier is done in the following way. The set of images used for training is created by picking at random in the training database until the target number of images gathered. If all examples are used, a random shuffling is applied to the set to avoid any bias. As it is a supervised classifier training, the labels according to each image is read at the same time. The labels are categorical (i.e. number from 0 to 4). The learning stage was performed using different algorithms.

The algorithms tested and their parameters are :

- a multi-class C-SVM algorithm using a Radial Basis Function (RBF) kernel with parameter $\gamma = 20$, $C = 7$ and termination criterion of $iter_{max} = 1000$ and $\epsilon = 10^{-6}$.
- a k-NN classification algorithm with parameter $max_k = 3$.
- a GBT algorithm with a Deviance Loss function, 300 boosting iterations a shrinkage of 0.1, a subsample portion of 0.9 and max depth of 10.

These parameters were chosen as offering good classification performance while keeping a reasonable bound on the training time.

5.1.4 Testing stage

The performance of the different descriptors and learning algorithms were evaluated via classification on the testing database. The metric chosen to evaluate performance is the proportion of correct classification over the total number of testing images. Note that the training and testing sets are disjoint.

5.1.5 Results

As it can be seen on Figure 9, the descriptors yield very good performances. We can see that the ground truth measure has the best performance, but this was expected. From these results, we can assess than binary descriptors, while being very compact and designed for speed, still provide a very high discriminant power.

These results also show that descriptors form a good association with usual machine learning algorithms for the task of image classification as they provide good classification performance while greatly reducing the training time. Indeed a look at Figure 11 shows that descriptors have a training time systematically two orders of magnitude below the ground truth training time, independently of the learning algorithm.

From Figures 10 and 11 we can give a quick evaluation of the learning algorithms chosen for this test. The first thing to observe is that GBT and SVM have almost exactly the same performances. Good performances were expected from SVM as it is known to give good results for this kind of classification tasks but it appears that GBT is a very valid alternative. Furthermore, GBT training time is almost one order of magnitude faster than SVM training time.

The kNN algorithm, while yielding reasonably good performances is consistently outperformed by SVM and GBT. Of course, the training time is very fast (from two to three order of magnitude faster than the other two), by design of the algorithm. Given its poor scalability to very large databases and its average performance, we can discard the kNN algorithm from the possible candidates of large scale image classification.

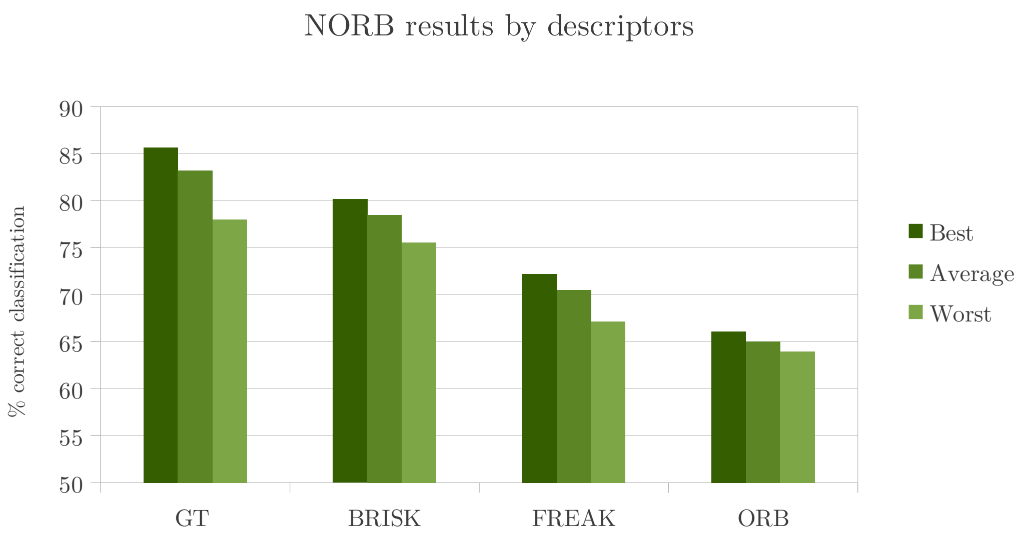

Figure 9: This graph presents the best, average and worst performance of each data representation. The labels are the following : GT is for Ground Truth and the others labels are the descriptors acronyms. The performance is measured as the percentage of correct classifications done in the testing set.

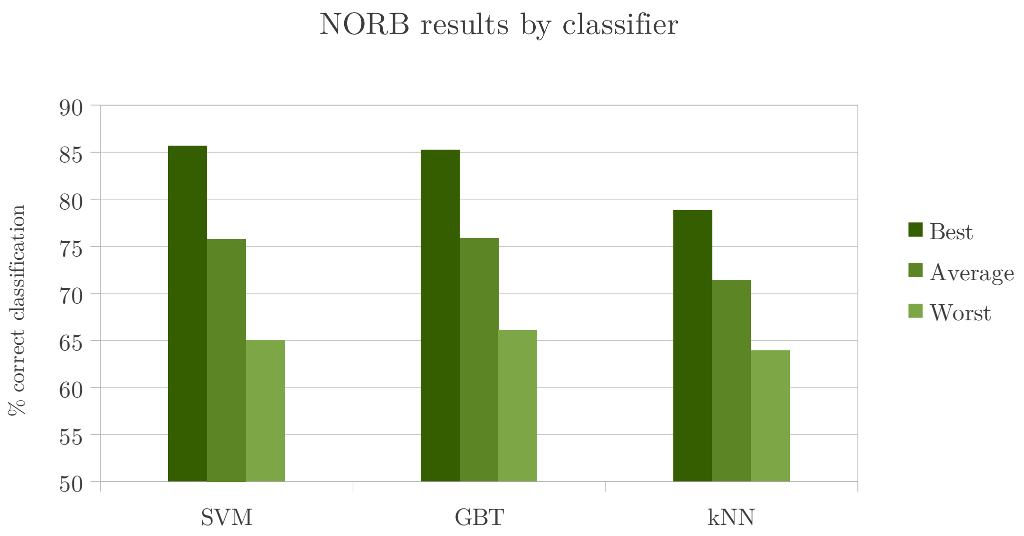

Figure 10: This graph presents the best, average and worst performance of each learning algorithm. The labels are the different classifiers. The performance is measured as the percentage of correct classifications done in the testing set.

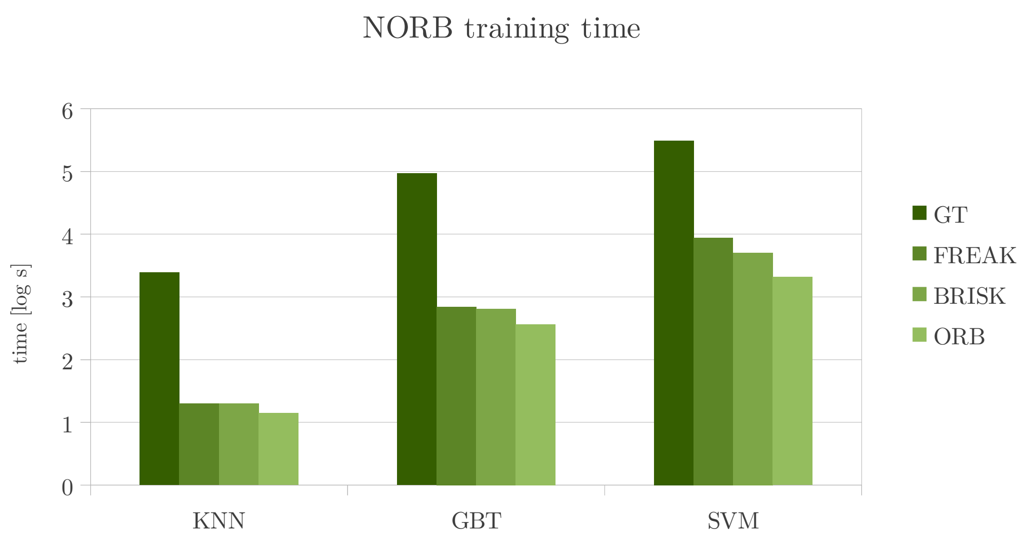

Figure 11: This graph presents the time needed to train the different learning algorithms in function of the data representation. The time axis uses a logarithmic scale, i.e. the unit of time is $\log_{10} s$.

5.2 Bag of Visual Words classification

5.2.1 Setup

A commonly used framework for large scale image classification and image-based retrieval uses a feature-based representation of images mapped to visual words and finally uses Vocabulary Trees (VT) to classify images. This basic method is generally improved using Geometry Verification (GV) and various re-ranking methods.

As we already said, most methods following this pipeline use real-valued descriptors, generally SIFT or SURF in the feature extraction part. The goal is to show that binary descriptors can be a valid alternative and that the compressed form we proposed in this work can potentially be much more efficient by skipping the mapping to visual words, since the compressed features can already be used as words.

The object classification task was performed on the Caltech-256 dataset which is a challenging dataset of relatively large scale. And since the best scores on this dataset are currently around 50% of global correct classification [49], this tends to assess the difficulty of the task.

Indeed if we compare to the NORB dataset, where each category had nearly 5000 images for training, the number of training images by category in the Caltech-256 is around 60. The training must be able to generalize on very few examples. Also the visual similarity between objects of a same category are sometimes not obvious, as it can be seen on Figure 12.

The goal of this last experiment is to train a classifier using a Bag of Visual Word and then evaluate its performance on the Caltech-256 dataset. Mostly for practical reasons, no re-ranking or geometric verification was implemented. In order to provide a baseline of this simple implementation, a classification using SIFT descriptors was performed. Then another classification using FREAK as a choice of binary descriptor has been done.

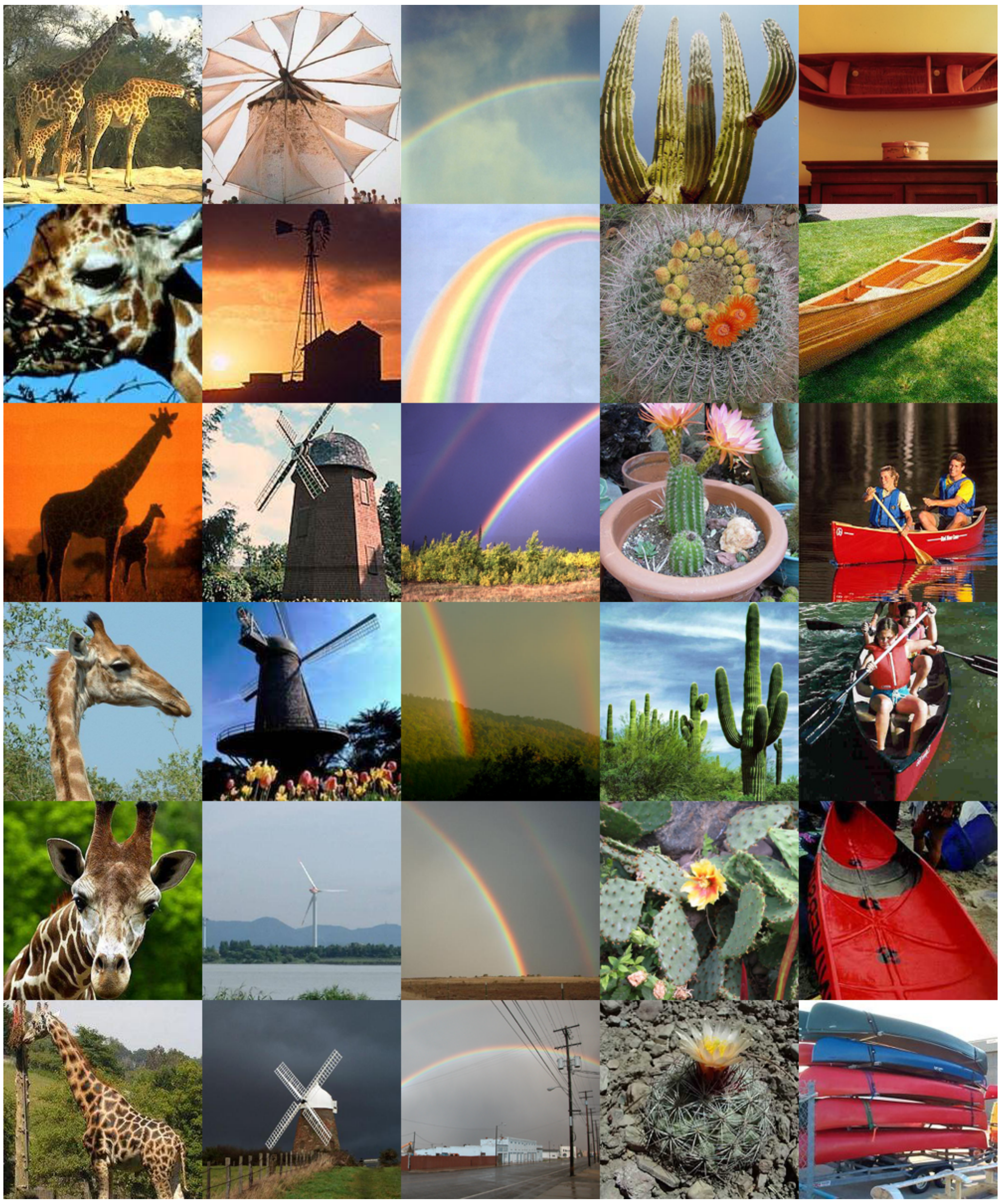

Figure 12: This is a small sample of the Caltech-256 image database. Each column is associated to a samples of 6 images coming from one category. They are (from left to right) 084.giraffe, 245.windmill, 170.rainbow, 025.cactus and 030.canoe.

5.2.2 Algorithm

The method implemented follows the general implementation of a BoVW classification algorithm. It can be summarized as the following sequence :

1. Train a vocabulary V of n visual words, by clustering a set of descriptors into n clusters and taking each cluster center as a visual word.

2. Run through all training images, detect and extract features. Then map the extracted descriptors to their closest visual words in V. Finally make an histogram per image of the n visual words and normalize it.

3. Train a classifier using histograms of visual words as input.

For the baseline the detector was BRISK, the descriptor SIFT, the clustering method k-means in $\mathbb{R}$ using k-means ++ initialization and the classifier à multiclass C-SVM.

For the naive binary approach the detector was BRISK, the descriptor FREAK, the clustering method k-means in $\boldsymbol{Z}_2$ and the classifier a multiclass C-SVM.

5.2.3 Results

As we it can be seen in Figure 13, the proposed method for classification is working. Indeed, the results are well above random classification (which would be $< 1\%$) and increase with the number of training images. Of course, one cannot hope to reach high performances with such a simple method, but a performance above 10% is already respectable.

But the main goal of this experiment was to see if binary descriptors were able to provide similar performance than standard real-valued descriptors such as SIFT. On this matter, the results are outstanding since the FREAK-based classification consistently outperforms the SIFT-based one.

Let us emphasize that each point on the graph correspond to tests done with cross-validation. This means that each result is the average of 10 runs, for which independently randomly sampled training and testing sets were created. There is thus a very low probability of statistical bias.

Figures 14 and 15 represents the confusion matrices corresponding to FREAK-based classification for 5 and respectively 35 training samples per category. Let us recall that a confusion matrix is defined as a matrix C for which an entry $c_{i,j}$ is the number of times a sample of class i is classified as being of class j. A perfect classification would thus yield an scaled identity matrix.

Confusion matrices help to determine the classification distribution. They can also be used to determine the global performance by computing the average of the diagonal. We can for example see in Figure 14 the confusion matrix of a low performance result (i.e. $< 4\%$). We can see a diagonal, but it is not very clear, and we can see many vertical lines. One is especially bright, and it means that this specific class had the tendency to capture all classifications.

The Figure 15 shows the classification for a better result (i.e. $> 10\%$). In this one, the diagonal is much more easy to notice and there are still vertical lines, but they do not dominate in any way.

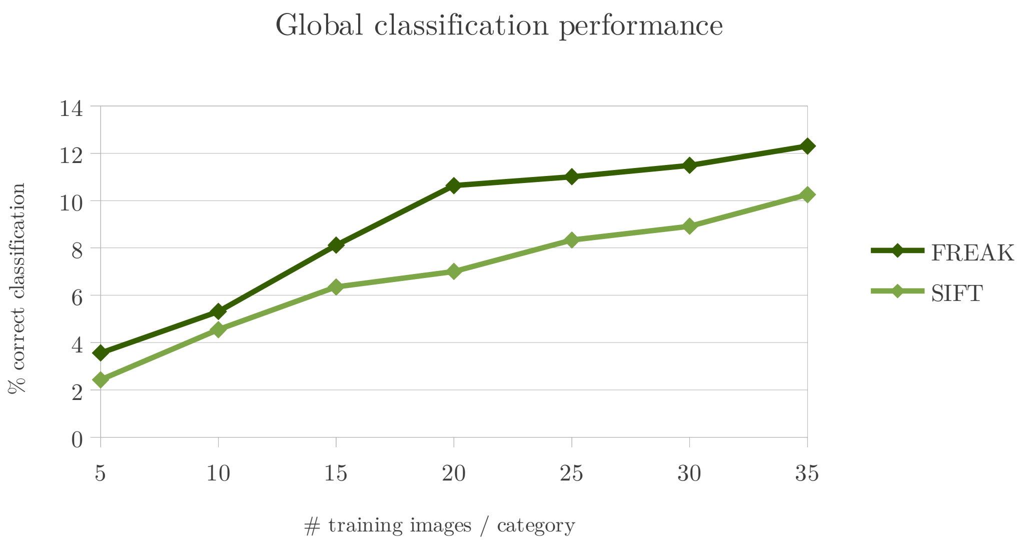

Figure 13: This graph presents the global performance of the BoVW classification in the Caltech-256 database. The performance is given for both FREAK and SIFT based methods. The x-axis is the number of training examples by category and the y-axis the percentage of correct classification done in the testing set. The number of samples used for testing is 25 by category.

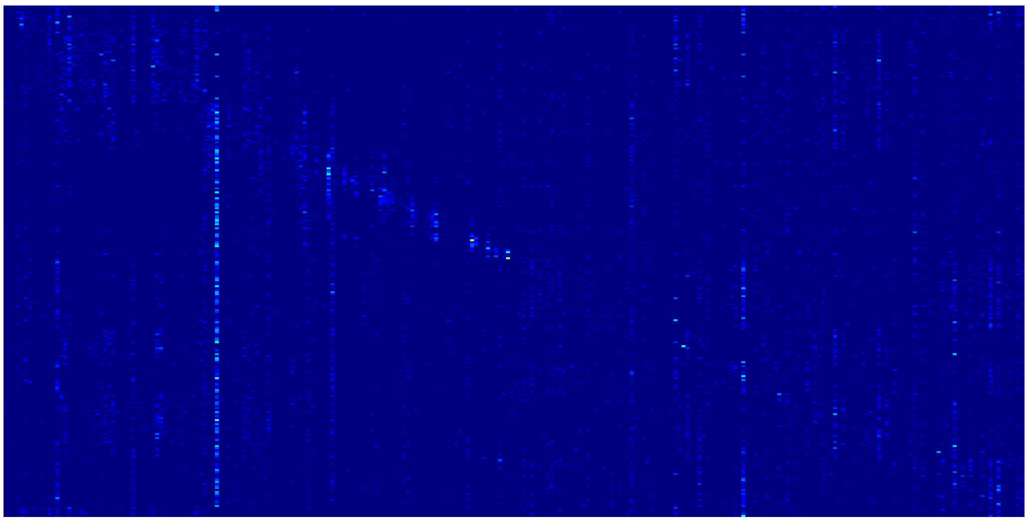

Figure 14: This figures represents the 256 × 256 confusion matrix generated after a 10 runs cross-validation of the FREAK-based classification with 5 training samples by category. The matrix is displayed in jet-color.

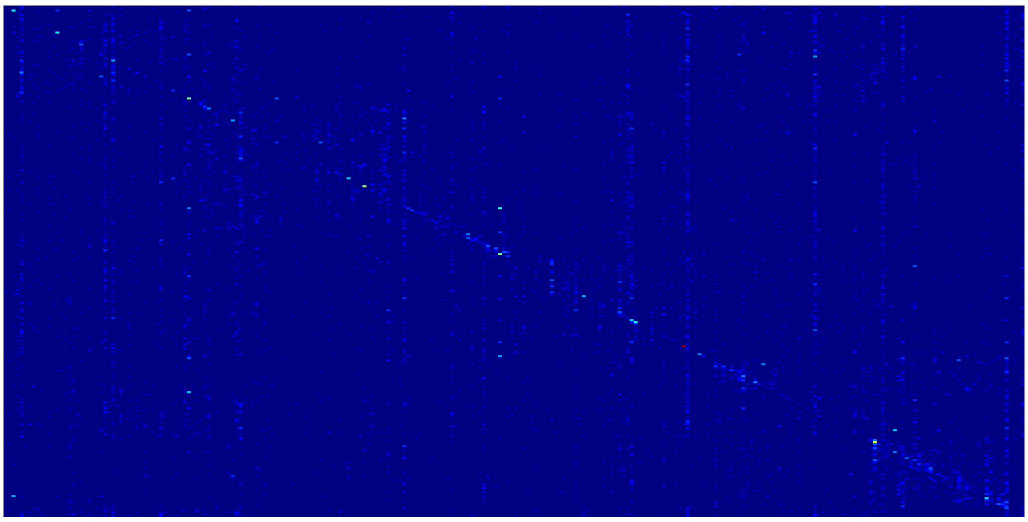

Figure 15: This figures represents the 256 × 256 confusion matrix generated after a 10 runs cross-validation of the FREAK-based classification with 35 training samples by category. The matrix is displayed in jet-color.

Conclusion

In conclusion let us review the proposed evaluations and contributions. We first proposed to formulate the FREAK pairs selection as an optimization of distance preserving dimensionality reduction. We saw that this formulation was incomplete to generate a good subset of pairs due to the phenomenon which we called the correlation bias. We thus devised a general method for pairs selection based on scoring and decorrelated sampling. The performance achieved by this technique outperformed the current implementation of FREAK.

This work on FREAK pairs selection shows how important the context is when designing algorithms with direct real-world applications. Indeed, the correlation bias is not obvious to infer in an abstract and general framework.

On the other hand, we saw that very theoretical concepts such as entropy analysis and distance preserving dimensionality reduction can yield very good results in empirical tests. While promising, the random projection technique has been shown as not suitable for this problematic. Indeed, the weaker version of the Johnson Lindenstrauss lemma in $\boldsymbol{Z}_2$ makes it not applicable to pairs selection.

In a second time, we used binary descriptors to solve the problem of image classification. The experiment on the NORB dataset showed that binary descriptors, while simple and compact, are very discriminative and are close in performance to much higher dimensional representation of images. They especially provide a boost in training time of two to three orders of magnitude which is fundamental for very large scale experiments.

The last experiment was designed to assess the validity of binary descriptors in Bag of Visual Words classification, which had always been performed using classical real-valued descriptors. The results showed that binary descriptors have their role to play in image classification since they perform even better than traditional descriptors. Of course, these preliminary results should be confirmed and built upon in future research.

Future work

Many aspect of this work can lead to extensions, but here are some interesting aspects that would benefit of more research.

FREAK

Pattern extension

As other binary descriptors, FREAK provide very good performance for a small cost and can be extended. Since we saw that patterns can be very correlated, a possible work would be to create a pattern yielding more dimensions and then apply the same kind of pairs selection to obtain a descriptor with even more discriminant power. A color model could also be associated to the intensity based values.

Compression

As we exposed before, one of the goal of designing a binary descriptor is to provide a more compact representation than the traditional real-valued descriptors. An other advantage is the integer representation, along with the Hamming distance, both having a very beneficial impact for the embedded and mobile computations.

However, even this representation is not sufficiently compact in a context like mobile visual search for example. Indeed, the pipeline for mobile visual search generally imply the use of a server and thus the transmission of the features or the storage of a database on the mobile side. This is why some authors chose to design compressed descriptors, for example Compressed Histograms of Gradients (CHoGs)[10].

We also saw that Bag of Hashs models made use of hashed features to create a vocabulary. In this way, instead of clustering complete descriptors as in the Bag of Visual Word model, and thus having to match query descriptors to their closest visual words, a very interesting future work would be to compress the descriptor directly to a vocabulary word.

This impose some quality on the compressed feature which has to project the descriptors in a way which create a good tesselation of the descriptors distribution. Ideally, the space of compressed features should have a high entropy, or in other words, each compressed descriptor should correspond to an equal number of original descriptors. Maximum entropy encoding techniques could provide very valuable techniques to create such a compressed feature.

Image Classification

Bag of Visual Words

Image classification using BoVW has been extensively done these recent years, using real-valued descriptors. This work is the first towards a reproduction of the pipeline using binary descriptors, but some experiments should be designed to assess the validity of binary descriptors based BoVW classification. Especially, a state of the art BoVW with Geometric Verification and re-ranking should be implemented.

Machine learning in $\boldsymbol{Z}_2$

We saw in this work that the k-means algorithm could be easily extended in $\boldsymbol{Z}_2$. But many algorithms such as SVMs always assume that samples are in Euclidean spaces and use norms and inner-products accordingly. An interesting future work would be to extend SVMs and other classification algorithms in $\boldsymbol{Z}_2$ as they could be much more efficient.

Deep learning

As we saw, image classification is very challenging. A way to explore would be to combine different classification techniques in a hierarchical manner to provide a better performance. Boosting and bagging techniques are generally used to group very weak and simple learners, but could be also very valid meta-algorithms applied to already complex algorithms.

Technical remarks

The different methods proposed in this work have been implemented using Open Source libraries and Software, and the code can be made available upon request. The implementation was done in C++ using the Qt framework[20] as the main frame. The OpenCV library[21] was used for almost all vision and machine learning algorithms.

Since almost all techniques introduced in this project are computationally intensive, parallelization was implemented on every critical component. It was achieved via multi-core implementation (using the Intel Threading Building Blocks libraries[22]) at the CPU level and CUDA[23] implementation at the GPU level.

A list of hardware and software used for implementation is presented at page 59.

To give an idea, k-means clustering more than 4 millions of points in 2000 clusters would take more than 120 hours using a single thread program. The parallelization is thus not only desirable but it becomes mandatory for such large scale computation.

The Caltech-256 database is relatively large scale since it contains more than 30000 images. But it has also been chosen so that all operations can be performed in RAM. Really huge image collection containing millions of images imply the use of databases which need to be sufficiently fast.

It is thus crucial to do research while keeping in mind this Big Data context, and this is why compact representation such as binary descriptors are really valuable.

[20]`http://qt.digia.com/`
[21]`http://opencv.org/`
[22]`http://threadingbuildingblocks.org/`
[23]`http://www.nvidia.com/object/cuda_home_new.html/`

Acknowledgements

Working on this project has been very exciting all along. It is very enjoyable to work on real world problematic. Also the balance between a theoretical research part and its corresponding implementation allows to understand what are the practical issues and compromises to be taken when implementing very nice theoretical concepts and at the same time to empirically assess its validity.

As it can be seen in the future work section, this project has led to many possible follow-ups and the subject is even more interesting now than at the beginning of the project.

I would first like to thank Prof. Pierre Vandergheynst for making this project possible and for his availability, as our discussions have had a great impact by generating new ideas and been very helpful to clarify some concepts.

I would also like to thank Alexandre Alahi for its inputs about FREAK and image classification and also Kirell Benzi for some helpful discussions.

I would also like to acknowledge Matthias Brändli for its very helpful LaTeXtemplate.

And finally a special thanks to the Open Source community which helps making open research stay in the game.

Equipment and software

Hardware

In this project the following development platforms were used :

Desktop Computer i7 3770K A

- CPU : Intel Core i7 3770K, 4 cores (8 threads) @ 4.5 GHz
- Memory : 32GB RAM @ 1600 MHz
- GPU : Asus GeForce GTX 680, 1536 CUDA cores.
- Base system : Ubuntu 12.04 LTS 64bit

Desktop Computer i7 3770K B

- CPU : Intel Core i7 3770K, 4 cores (8 threads) @ 4.8 GHz
- Memory : 32GB RAM @ 1600 MHz
- GPU : Asus GeForce GTX 660, 1344 CUDA cores.
- Base system : Ubuntu 12.04 LTS 64bit

Software

The following software and libraries were used :

- GCC 4.6.3 with glibc 2.15
- QtCreator 2.4.1 with Valgrind 3.7.0
- Qt libraries 4.8.0
- Intel TBB library 2.0
- OpenCV library 2.4.3-148-g7f542e3
- CUDA libraries 3.0

References

[1] Alahi, A., Ortiz, R., and Vandergheynst, P. FREAK: Fast Retina Keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition* (2012). CVPR 2012 Open Source Award Winner.

[2] Ambai, M., and Yoshida, Y. Card: Compact and real-time descriptors. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (nov. 2011), pp. 97 –104.

[3] Anandan, P. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision 2* (1989), 283–310.

[4] Arandjelović, R., and Zisserman, A. Three things everyone should know to improve object retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition* (2012).

[5] Arthur, D., and Vassilvitskii, S. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 2007), SODA '07, Society for Industrial and Applied Mathematics, pp. 1027–1035.

[6] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. Speeded-up robust features (surf). *Comput. Vis. Image Underst. 110*, 3 (June 2008), 346–359.

[7] Brown, M., and Lowe, D. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision 74* (2007), 59–73.

[8] Burges, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery 2* (1998), 121–167.

[9] Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., and Fua, P. BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 7 (2012), 1281–1298.

[10] Chandrasekhar, V., Takacs, G., Chen, D., Tsai, S., Grzeszczuk, R., and Girod, B. Chog: Compressed histogram of gradients a low bit-rate feature descriptor. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (june 2009), pp. 2504 –2511.

[11] Cortes, C., and Vapnik, V. Support-vector networks. In *Machine Learning* (1995), pp. 273–297.

[12] Dasgupta, S., and Gupta, A. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures Algorithms 22*, 1 (2003), 60–65.

[13] Deng, J., Li, K., Do, M., Su, H., and Fei-Fei, L. Construction and Analysis of a Large Scale Image Ontology. Vision Sciences Society.

[14] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision 88*, 2 (June 2010), 303–338.

[15] Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories.

[16] Fei-Fei, L., and Perona, P. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (june 2005), vol. 2, pp. 524 – 531 vol. 2.

[17] Floreano, D., and Mattiussi, C. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies.* The MIT Press, 2008.

[18] Forgy, E. W. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics 21* (1965), 768–769.

[19] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29* (2000), 1189–1232.

[20] Friedman, J. H. Stochastic gradient boosting. *Comput. Stat. Data Anal. 38*, 4 (Feb. 2002), 367–378.

[21] Gionis, A., Indyk, P., and Motwani, R. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1999), VLDB '99, Morgan Kaufmann Publishers Inc., pp. 518–529.

[22] Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. Tech. Rep. 7694, California Institute of Technology, 2007.

[23] Harris, C., and Stephens, M. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference* (1988), pp. 147–151.

[24] He, J., Feng, J., Liu, X., Cheng, T., Lin, T.-H., Chung, H., and Chang, S.-F. Mobile product search with bag of hash bits and boundary reranking. In *CVPR'12* (2012), pp. 3005–3012.

[25] Horn, B. K., and Schunck, B. G. Determining optical flow. *Artificial Intelligence 17*, 1–3 (1981), 185 – 203.

[26] Indyk, P., and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing* (New York, NY, USA, 1998), STOC '98, ACM, pp. 604–613.

[27] Johnson, W. B., and Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, vol. 26 of *Contemp. Math.* Amer. Math. Soc., Providence, RI, 1984, pp. 189–206.

[28] Ke, Y., and Sukthankar, R. Pca-sift: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition* (Washington, DC, USA, 2004), CVPR'04, IEEE Computer Society, pp. 506–513.

[29] Larrañaga, P., Kuijpers, C., Murga, R., Inza, I., and Dizdarevic, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review 13* (1999), 129–170.

[30] LeCun, Y., Huang, F. J., and Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR (2)'04* (2004), pp. 97–104.

[31] Leutenegger, S., Chli, M., and Siegwart, R. Brisk: Binary robust invariant scalable keypoints. In *ICCV'11* (2011), pp. 2548–2555.

[32] Lloyd, S. Least squares quantization in pcm. *Information Theory, IEEE Transactions on 28*, 2 (mar 1982), 129 – 137.

[33] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov. 2004), 91–110.

[34] Lucas, B. D., and Kanade, T. An iterative image registration technique with an application to stereo vision (ijcai). In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)* (April 1981), pp. 674–679.

[35] Mair, E., Hager, G., Burschka, D., Suppa, M., and Hirzinger, G. Adaptive and generic corner detection based on the accelerated segment test. In *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., vol. 6312 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 183–196.

[36] Mikolajczyk, K., and Schmid, C. Scale & affine invariant interest point detectors. *Int. J. Comput. Vision 60*, 1 (Oct. 2004), 63–86.

[37] Mikolajczyk, K., and Schmid, C. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 27*, 10 (oct. 2005), 1615 –1630.

[38] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2007).

[39] Philbin, J., and Zisserman, A. Object mining using a matching graph on very large image collections. In *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing* (Washington, DC, USA, 2008), ICVGIP '08, IEEE Computer Society, pp. 738–745.

[40] Qiu, G. Indexing chromatic and achromatic patterns for content-based colour image retrieval. *Pattern Recognition 35*, 8 (2002), 1675–1686.

[41] Rosten, E., and Drummond, T. Machine learning for high-speed corner detection. In *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 430–443.

[42] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. R. Orb: An efficient alternative to sift or surf. In *ICCV'11* (2011), pp. 2564–2571.

[43] Schindler, G., Brown, M., and Szeliski, R. City-scale location recognition. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR07)* (Minneapolis, June 2007).

[44] Shi, J., and Tomasi, C. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on* (jun 1994), pp. 593 –600.

[45] Sivic, J., and Zisserman, A. Video google: a text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on* (oct. 2003), pp. 1470 –1477 vol.2.

[46] Szeliski, R. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis. 2*, 1 (Jan. 2006), 1–104.

[47] Szeliski, R. *Computer Vision: Algorithms and Applications*, 1st ed. Springer-Verlag New York, Inc., New York, NY, USA, 2010.

[48] Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W.-C., Bismpigiannis, T., Grzeszczuk, R., Pulli, K., and Girod, B. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceedings of the 1st ACM SIGMM International Conference on Multimedia Information Retrieval, MIR 2008, Vancouver, British Columbia, Canada, October 30-31, 2008* (2008), M. S. Lew, A. D. Bimbo, and E. M. Bakker, Eds., ACM, pp. 427–434.

[49] Todorovic, S., and Ahuja, N. Learning subcategory relevances for category recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (june 2008), pp. 1 –8.

[50] Tola, E., Lepetit, V., and Fua, P. Daisy: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 5 (May 2010), 815–830.

[51] Tsai, S., Chen, D., Takacs, G., Chandrasekhar, V., Vedantham, R., Grzeszczuk, R., and Girod, B. Fast geometric re-ranking for image-based retrieval. In *Image Processing (ICIP), 2010 17th IEEE International Conference on* (sept. 2010), pp. 1029 –1032.

[52] Tsai, S. S., Chen, D., Singh, J. P., and Girod, B. Rate-efficient, real-time cd cover recognition on a camera-phone. In *Proceedings of the 16th ACM international conference on Multimedia* (New York, NY, USA, 2008), MM '08, ACM, pp. 1023–1024.

[53] van de Sande, K., Gevers, T., and Snoek, C. Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 32*, 9 (sept. 2010), 1582 –1596.

[54] Vapnik, V., and Lerner, A. Pattern Recognition using Generalized Portrait Method. *Automation and Remote Control 24* (1963).

[55] Vempala, S. S. *The Random Projection Method.* American Mathematical Society, 2004.