

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Detecting Nature in Pictures

Tiago Miguel Pereira Andrade



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Jorge Alves da Silva

12th July 2013

Detecting Nature in Pictures

Tiago Miguel Pereira Andrade

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Rui Carlos Camacho de Sousa Ferreira da Silva (Associate Professor)

External Examiner: José Manuel de Castro Torres (Associate Professor)

Supervisor: Jorge Alves da Silva (Assistant Professor)

12th July 2013

Abstract

With the advent of large-scale image sharing on the internet, on websites such as Flickr and Panoramio, combined with the ability to share the geographical location of those pictures, there are now large sources of data ready to be mined for useful information.

Using this data to automatically create a map of man-made and natural areas of our planet, taking advantage of these readily available sources of information, would provide additional knowledge to decision-makers responsible for world-conservation.

The problem of determining the degree of naturalness of an image, which is required to create such a map based on pictures, can be generalized as a scene classification task, which is an active area of research in the computer vision and image processing fields.

The analysis of scene classification literature revealed an underlying pattern for classification: the extraction of relevant features, followed by their transformation into an intermediate representation, having an image classifier as the final step.

Experiments were performed to better understand the applicability of each of the identified scene classification techniques to perform the distinction between man-made and natural images. Their advantages and limitations, such as their computational costs, are detailed.

By carefully selecting the techniques and their parameters it was possible to build a classifier that is capable of distinguishing between natural and man-made scenery with high accuracy and that can also process a large amount of pictures within a reasonable time frame.

Resumo

Com o advento da partilha de imagens em grande escala, em portais como o Flickr e Panoramio, e combinado com a capacidade de partilhar a localização geográfica dessas mesmas imagens, existem agora grandes fontes de dados prontas a serem processadas para a extração de informação útil.

A utilização destes dados para a criação de um mapa das áreas naturais e de origem humana do nosso planeta, tirando partido destas fontes de dados prontamente disponíveis, pode fornecer conhecimento adicional aos decisores políticos responsáveis pela conservação do planeta.

O problema de determinar o grau de naturalidade de uma imagem, pré-condição para a criação de tal mapa, pode ser generalizado como um problema de classificação de paisagens, que é uma área ativa de investigação nas áreas de visão por computador e processamento de imagem.

A análise da literatura sobre classificação de paisagens revela um padrão subjacente para classificação: a extração de características relevantes, seguido da sua transformação numa representação intermédia, tendo um classificador como o passo final.

Foram executadas experiências para melhor compreender a aplicabilidade de cada uma das técnicas identificadas para a classificação de paisagens quando aplicadas à tarefa de distinguir entre imagens naturais e de origem humana. As suas vantagens e limitações, como os seus requisitos computacionais, são detalhados.

Com uma escolha cuidada das técnicas e respetivos parâmetros foi possível construir um classificador capaz de distinguir entre paisagens naturais e de origem humana com elevada precisão, mas também capaz de processar uma grande quantidade de imagens num espaço de tempo razoável.

Acknowledgements

A very big thank you to my supervisor, Prof. Jorge Silva, for readily answering all of my queries and reviewing my work, even when my requests for feedback came dangerously close to the delivery deadlines. The in-depth feedback and suggestions were invaluable and significantly improved the quality of my work.

I would also like thank everyone at UNEP-WCMC for providing me with a motivating and very pleasant work environment, and for giving me the freedom to organize and progress with my work as I deemed fit. I also appreciate all the well-thought-out questions, suggestions and feedback that I received during the presentations I made there.

And a special thanks to everyone both here in Cambridge and back in Portugal who boosted my morale and pushed me to exceed myself whenever I struggled to make progress during the development of this dissertation. Thank you for your support and friendship.

Tiago Andrade

Contents

1	Introduction	1
1.1	Context and Motivation	1
1.1.1	Protected Planet	1
1.1.2	Mapping Natural Areas	2
1.2	Problem Definition	2
1.3	Main Goals	3
1.4	Document Structure	3
2	Scene Classification	5
2.1	Features	5
2.1.1	Colour	5
2.1.2	Texture	6
2.1.3	Filter Outputs	6
2.1.4	Gradients	6
2.2	Descriptors	7
2.2.1	Gist	8
2.2.2	SIFT	8
2.2.3	HOG	9
2.2.4	LBP	9
2.2.5	CENTRIST	10
2.3	Extraction Method	10
2.3.1	Sparse	10
2.3.2	Dense	11
2.3.3	Multi-resolution	11
2.4	Descriptor Transformation	12
2.4.1	Hellinger Kernel	12
2.4.2	Principal Component Analysis	12
2.5	Intermediate Representations	13
2.5.1	Bag-of-Words Representation	13
2.5.2	Fisher Kernel	16
2.6	Spatial Information	18
2.6.1	Spatial Pyramid Matching	18
2.6.2	Spatial Fisher Vector	19
2.7	Learning Techniques	19
2.7.1	kNN	19
2.7.2	SVM	21
2.7.3	kNN-SVM	23
2.8	Datasets and Results	24

CONTENTS

2.8.1	Fifteen scene dataset	24
2.8.2	SUN database	25
2.8.3	Vogel & Schiele	25
2.9	Literature Shortcomings	25
2.9.1	Degree of Naturalness	25
2.9.2	Performance Evaluations	26
2.9.3	Comparing Improvements Separately	27
3	Classification Framework	29
3.1	Functionality	29
3.2	Features	30
3.2.1	Flexibility	30
3.2.2	Caching	32
3.2.3	Extensibility	32
3.2.4	Parallelization	32
3.3	Libraries	32
3.3.1	CImg	33
3.3.2	VLFeat	33
3.3.3	Boost	33
3.3.4	Yael	33
3.3.5	LIBSVM	34
3.3.6	LIBLINEAR	34
3.3.7	GMM-Fisher	34
4	Methodology	35
4.1	Objectives	35
4.2	Experimental Set-up	35
4.3	Definition of Natural	36
4.4	Default Values	36
4.5	Performance Analysis	37
4.6	New Dataset	37
4.7	Limitations	39
4.7.1	Execution time	39
4.7.2	Implementation complexity	39
4.7.3	Memory allocation	39
5	Detecting Nature	41
5.1	Experiments	41
5.1.1	Descriptor Type	41
5.1.2	Image Smoothing	42
5.1.3	Image Resizing	42
5.1.4	Colour Information	44
5.1.5	Descriptor Scale	44
5.1.6	Distance Transforms	45
5.1.7	Dimensionality Reduction	46
5.1.8	Encoding Method	46
5.1.9	Spatial Information	47
5.1.10	Type of Classifier	48
5.1.11	Tuning Classifier	49

CONTENTS

5.1.12	Number of Training Images	49
5.2	Optimization	50
5.2.1	Optimizing for Accuracy	50
5.2.2	Optimizing for Speed	51
5.3	Additional Information	52
5.3.1	Cross-dataset Generalization	52
5.3.2	Confidence Rating	54
6	Result Analysis	55
6.1	Comparing algorithms	55
6.1.1	Importance of Experimenting	55
6.1.2	Adapting Techniques	55
6.2	Building the Framework	56
6.3	Computational Cost	56
6.4	Mixing Datasets	56
6.5	Ignoring Irrelevant Information	56
6.6	Detailing Methodology	57
6.7	Human-in-the-Loop	58
6.8	Naturalness Ranking	58
6.9	Large Scale Viability	58
7	Conclusions	61
7.1	Future Work	62
	References	63
A	Implemented Techniques	67
A.1	Features	67
A.2	Descriptors	67
A.3	Extraction Method	67
A.4	Descriptor Transform	68
A.5	Intermediate Representations	68
A.6	Spatial Information	68
A.7	Learning Techniques	68
B	Configuration Parameters	69
B.1	image	69
B.1.1	type	69
B.1.2	maxResolution	69
B.1.3	forceSize	69
B.1.4	smoothingSigma	70
B.2	features	70
B.2.1	type	70
B.2.2	gridSpacing	70
B.2.3	patchSize	70
B.2.4	transforms	70
B.3	codebook	70
B.3.1	type	71
B.3.2	codewords	71

CONTENTS

B.3.3	pcaDimension	71
B.3.4	textonImages	71
B.3.5	totalFeatures	71
B.4	histogram	71
B.4.1	type	71
B.4.2	pyramidLevels	72
B.5	classifier	72
B.5.1	type	72
B.5.2	c	72
B.5.3	trainImagesPerClass	72

List of Figures

1.1	Screen captures of the protectedplanet.net website	2
2.1	Visual representations of different descriptors	7
2.2	SIFT descriptor example using a 2 by 2 patch	8
2.3	Multiple LBP configurations	9
2.4	Two distinct extraction techniques	10
2.5	Diagram of the bag-of-codewords technique	13
2.6	Clustering techniques for the codebook creation	15
2.7	Example of a three-level pyramid	18
2.8	Comparison of SPM and SFV	20
2.9	Images that NBNN has difficulty distinguishing	21
2.10	Example of using a kernel to map data into a higher dimensionality space	23
2.11	Example pictures of the fifteen scene dataset	24
2.12	Classification results for the SUN database	26
4.1	Images in the SUN Database with disputable labels	38
5.1	The SVM misclassification penalty parameter	50
5.2	Impact of number of training images in classification accuracy	51
6.1	Effect of humans on the foreground of an image	57
6.2	Naturalness ranking of several example images	59

LIST OF FIGURES

List of Tables

2.1	Classification results for the fifteen scene dataset	25
4.1	Classification results where mean per-class accuracy is better	36
4.2	Default settings used during the experiments	37
5.1	Classification results for different types of descriptors	41
5.2	Blurring images before extracting descriptors	42
5.3	Impact of scaling images before feature extraction	43
5.4	Using colour information during descriptor extraction	44
5.5	Impact of scale and spacing of SIFT descriptors	45
5.6	Transforming descriptors using Hellinger kernel	46
5.7	Reducing dimensionality of a SIFT descriptor with size 128	46
5.8	The impact of codebook size in the Fisher framework	47
5.9	The impact of codebook size when using Bag-of-Words	47
5.10	Using spatial information with Bag-of-Words	48
5.11	Differences between linear and non-linear SVM classifiers	49
5.12	Settings used to achieve the best accuracy	52
5.13	Confusion matrix for the best results	52
5.14	Settings used to achieve better speed	53
5.15	Confusion matrix for the balanced results	53
5.16	Confusion matrix for the 15-scene dataset merged into two classes	54
5.17	Correlation between confidence and correct classifications	54

LIST OF TABLES

Abbreviations

CENTRIST	CENsus TRansform hISTogram
CUDA	Compute Unified Device Architecture
EXIF	Exchangeable Image File
GPS	Global Positioning System
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
HSV	Hue, Saturation, Value
JSON	JavaScript Object Notation
kNN	k-Nearest Neighbours
LBP	Local Binary Pattern
NBNN	Naive Bayes Nearest Neighbour
OpenCL	Open Computing Language
PCA	Principal Component Analysis
RGB	Red, Green, Blue
SIFT	Scale-Invariant Feature Transform
SIMD	Single Instruction, Multiple Data
SFV	Spatial Fisher Vectors
SPM	Spatial Pyramid Matching
SVM	Support Vector Machine
UNEP	United Nations Environment Programme
WCMC	Word Conservation Monitoring Centre

Chapter 1

Introduction

Decision-makers responsible for the monitoring and legislation of issues related to biodiversity and world conservation require data and expert knowledge on which to base their decisions.

This is where UNEP World Conservation Monitoring Centre comes in, as it is their mission to ‘evaluate and highlight the many values of biodiversity and put authoritative biodiversity knowledge at the centre of decision-making’ (UNEP-WCMC [2013](#)).

With the objective of further expanding the range of information they can provide, UNEP-WCMC is also experimenting with automated data gathering and crowd-sourcing.

1.1 Context and Motivation

UNEP World Conservation Monitoring Centre, with data gathering and analysis as the core of their mission, described two different situations that could be improved with the use of computer vision and image processing techniques. Both of these scenarios, described in the following sections, refer to the same problem in their core, which is one of detecting nature in pictures.

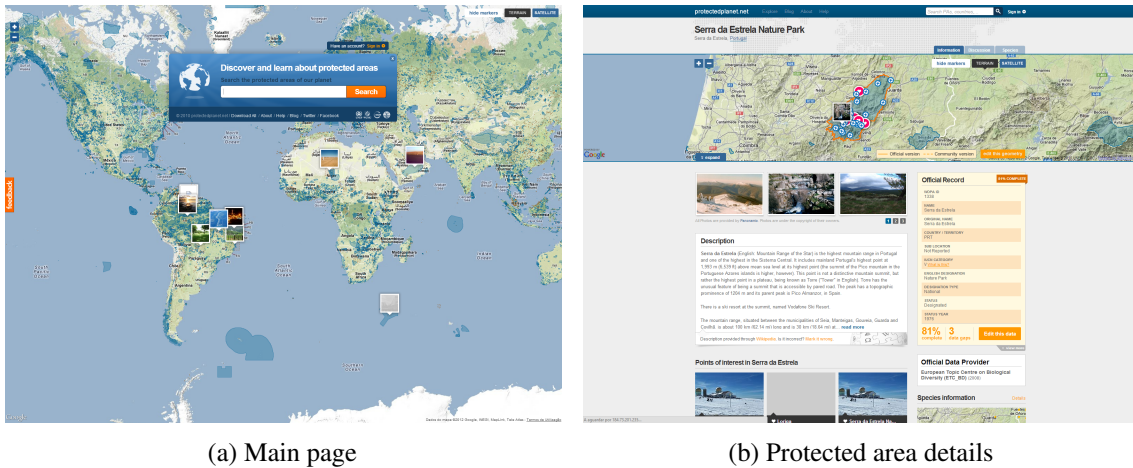
1.1.1 Protected Planet

Protectedplanet.net is an interactive website that uses ‘Citizen Science’ to boost the global interest in protected areas. This means users are encouraged to find and improve information on protected areas around the world. This information ranges from their description and points of interest to pictures which can be shared and scored by the users for each of the protected areas of our planet. Figure [1.1](#) shows the appearance of the website.

To ensure the correctness of the information available in the website, all information submitted by users is curated by the the Protected Areas Programme’s staff at UNEP-WCMC. This manual process incurs in extra costs and carries a degree of subjectivity on what does and does not get accepted.

When dealing with user-submitted images, in order to reduce the costs and decrease the subjectivity of the review process, an automated solution, supported by computer vision and image processing techniques is ideal.

Introduction



(a) Main page

(b) Protected area details

Figure 1.1: Screen captures of the protectedplanet.net website

1.1.2 Mapping Natural Areas

Websites like Flickr and Panoramio allow users to share geotagged images. The image's location can be either obtained directly from the image by using Exchangeable Image File format (EXIF) tags, or provided manually by the user.

By using an image's location and identifying whether it represents a natural scene or a man-made one, it is possible to determine the degree of naturalness on a particular location and time.

Processing a large number of images can fuel the creation of a map that identifies which areas of the Earth have been affected by human interference and which are still in their natural state. If an image's timestamp is also taken into account, then it is also possible to see the evolution of these man-made and natural areas.

This information is important to legislators and decision-makers as it provides them with details regarding possibly problematic areas that may require intervention or to study the impact of other actions that may have been taken previously. Another use for this data is that it can also be cross-referenced with information about protected areas to evaluate their effectiveness.

1.2 Problem Definition

The problem to be solved consists of the definition and implementation of a tool that is capable of automatically evaluating a picture and classifying it into one of two classes: natural or man-made.

In order to improve its acceptance as a replacement for manual evaluation, the tool should always output its confidence rating regarding the decision made, allowing for the final verdict to be deferred to a human operator when a low confidence decision is reached.

For an automation of the entire classification process, the tool should be able to be easily integrated into existing applications, such as the protectedplanet.net website, or to be plugged into a new application that may require the same kind of data evaluation, allowing for its parametrization for more effective image processing in each of these different contexts.

1.3 Main Goals

The main goals of this work can be summarized as follows:

- Compare the usefulness of several competing state-of-the-art scene classification algorithms when applied to the task of determining the degree of naturalness of an image.
- Provide an open-source, easy to extend framework for automated scene classification and algorithm evaluation.
- Evaluate the computational cost of the implemented techniques and the influence of changes in their parameters in said cost and analyse any trade-offs between speed and accuracy.
- Analyse the implemented algorithm's ability to generalize the information gathered from training on one dataset and applying it to a different data source.
- Test the framework's ability to cope with irrelevant information that may appear on images, such as the presence of humans in pictures.
- Provide a detailed description of all settings used to during the experiments to allow for easy reproduction of the results.

1.4 Document Structure

Chapter 2 reviews the current state-of-the-art techniques used in the scene classification area. An overview of each technique, as well as its possible advantages and shortcomings in comparison to other alternatives are also presented. Always taking into account the goals of this work, this chapter also identifies some of the knowledge gaps in the literature.

Details about the developed framework are presented in chapter 3. The main tasks that are required of such framework are listed, as well as other optional improvements that were also added. External libraries and details about their inclusion in the project are also described here.

Chapter 4 describes the procedure used to execute all of the reported experiments. Information about the dataset and how it was built, as well as all parameters is also included. The main limitations of the experiments that were performed are also detailed in this section.

The results of the experiments are presented in chapter 5. The algorithms identified in chapter 2 are compared, as well as their parameters, reporting their impact on the classification performance and computational cost. Taking into account these experiments, an overview of the impact of optimizing the framework's parameter for both speed and accuracy is provided.

Chapter 6 goes through the goals of this work, analysing and discussing their viability based on the results of the experiments.

Chapter 7 contains the final remarks about the work, as well as possible future work.

Introduction

Chapter 2

Scene Classification

Scene classification, or scene recognition, refers to the problem of automatically assigning a label to an image among a set of semantic categories (e.g. forest, mountain, and street) (Wu and Rehg 2011; Zhou, Zhou and Hu 2013). The applications for scene classifications range from acquiring context for object recognition (Torralba 2003), to content-based image indexing and retrieval (Vogel and Schiele 2007), and to robot and vehicle navigation (Wu and Rehg 2011).

One can define the problem of detecting whether the view represented in a picture corresponds to a natural or a man-made scene as a subset of the scene classification problem. In fact, determining the degree of naturalness of a picture is a part of the ‘Spatial Envelope’ technique proposed by Oliva and Torralba (2001) for scene classification.

The act of classifying an image has three major components. The first one is the extraction of a set of features from the image, which is detailed in sections 2.1 to 2.3. Then, a representation of the image is created, using the process outlined in sections 2.4 to 2.6. The final part is using a machine learning technique to classify the images, as described in sections 2.7 and 2.8.

2.1 Features

A feature of an image is a representation, characteristic or property of that image (Boutell, Brown and Luo 2002).

There are several features that can be used for scene classification, such as colour, texture, filter outputs and gradients.

2.1.1 Colour

Computers can represent colours in many different formats. Each of these may provide different invariance properties, such as light colour or light intensity changes and shifts, to image descriptors, when used as part of an image classification framework. Some of the most popular are RGB, Opponent and HSV (Van de Sande, Gevers and Snoek 2010).

RGB is the most common colour space used by computer images since it is the same representation required by computer screens. It uses three channels corresponding to the intensity of the

colours red, green and blue. These three components are added together using an additive colour model in order to get the final colour.

The opponent colour space corresponds to the Principal Component Analysis of the RGB colour representation on a set of images. The transformation described in eq. (2.1), obtained by using PCA on a set of natural images, can be used to decorrelate the colour components of a set of images (Boutell, Brown and Luo 2002).

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix} \quad (2.1)$$

HSV is another way to represent colours, by using the hue, saturation and value components. While the properties of this model make for a good colour picker system and is commonly used in image editing software, it has also been successfully used in the field of computer vision (Bosch, Zisserman and Muñoz 2008).

2.1.2 Texture

The texture of an image patch refers to an object's surface and structural properties. While the direct application of textural properties has been used to distinguish between different material and textile patterns, such as in the work by Ojala, Pietikainen and Maenpaa (2002), most works in scene classification opt for simpler properties to describe an image, such as gradients, described in section 2.1.4.

2.1.3 Filter Outputs

Several signal processing filters can be applied to image processing as well. Among these, the Fourier transform and Gabor filters are the most common. Gabor filters can be used as an edge detection tool as they can filter an image's components that are oriented in a certain angle. The Fourier transform, on the other hand, decomposes a signal into its sinusoidal components. When these two filters are combined, they can be used to determine the frequency of a signal in an image for a given orientation.

2.1.4 Gradients

Gradients can be used to determine orientations and borders of objects and shapes. There are several ways of representing gradients, ranging from simple binary representations, saying whether the value increased or decreased between two points, to more complex representations that calculate the gradient intensity and orientation.

Gradients are the most common features used by descriptors in scene classification (Lowe 2004; Dalal and Triggs 2005).

2.2 Descriptors

In order to turn the features described in the previous chapter into a more tractable representation, several descriptors have been proposed for image processing. These descriptors attempt to move away from pixel representations and to provide a description closer to the image semantics (Boutell, Brown and Luo 2002). There are several desirable properties in a descriptor for its use in scene classification (Wu and Rehg 2011). Descriptors should allow for a holistic representation of the scene, since a global view of an image can give more details about a scene than information about its components. They should be able to capture structural properties of a scene, such as shapes, flat surfaces, and tiles without getting distracted by detailed textural information. The descriptor should also be generalizable, allowing for the comparison of vastly different visual representations of the same semantic concept.

The next sections describe the most often used descriptors that achieve good results for scene classification: Gist, SIFT, HOG, LBP, and CENTRIST. Figure 2.1 shows the results of the application of some descriptors to an example picture.

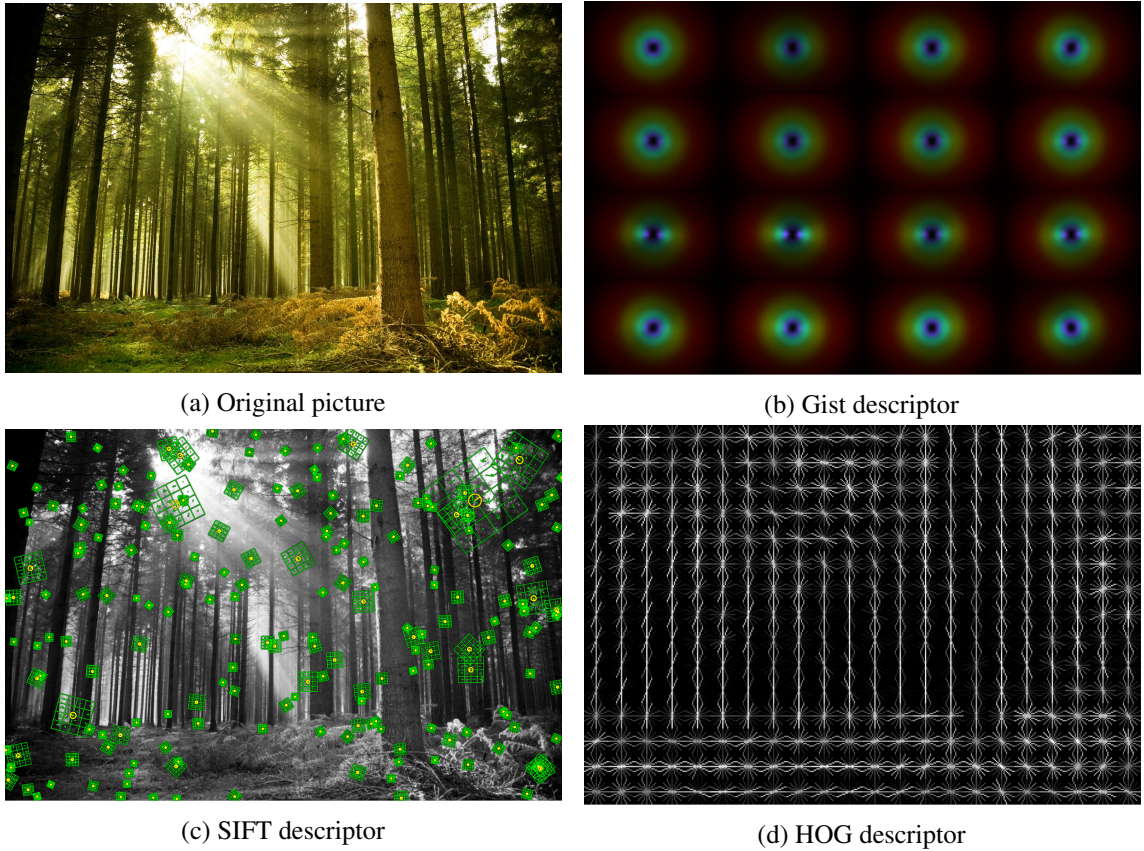


Figure 2.1: Visual representations of different descriptors

2.2.1 Gist

First described by Oliva and Torralba (2001) this descriptor attempts to represent spatial structures by using a ‘Discrete Fourier Transform’ to calculate the power spectra of an image and ‘Principal Component Analysis’ to filter the frequencies used to describe the image. This allows for a representation of the image that uses exclusively global features.

While an important work, as a pioneer for scene recognition without segmenting or recognizing local objects, its results have been surpassed by state-of-the-art approaches.

2.2.2 SIFT

‘Scale-Invariant Feature Transform’ is one of the most used descriptors in the scene classification literature (Bosch, Muñoz and Martí 2007; Wu and Rehg 2011). It was proposed by Lowe (2004) as a way to extract features that could be used to match different views of an object or scene.

Once the point where the descriptor will be extracted from has been determined, using one of the techniques described in section 2.3, including its scale and rotation, the SIFT can be calculated for that point. This is done by creating an histogram of gradient intensity and orientations, where Lowe (2004) uses a 4 by 4 grid of bins and 8 gradient orientations, resulting in a $4 \times 4 \times 8 = 128$ element feature vector, as exemplified in fig. 2.2. Each of these bins is weighted using a Gaussian window, meaning that bins further away from the descriptor centre are weighted less than the ones closer to the centre.

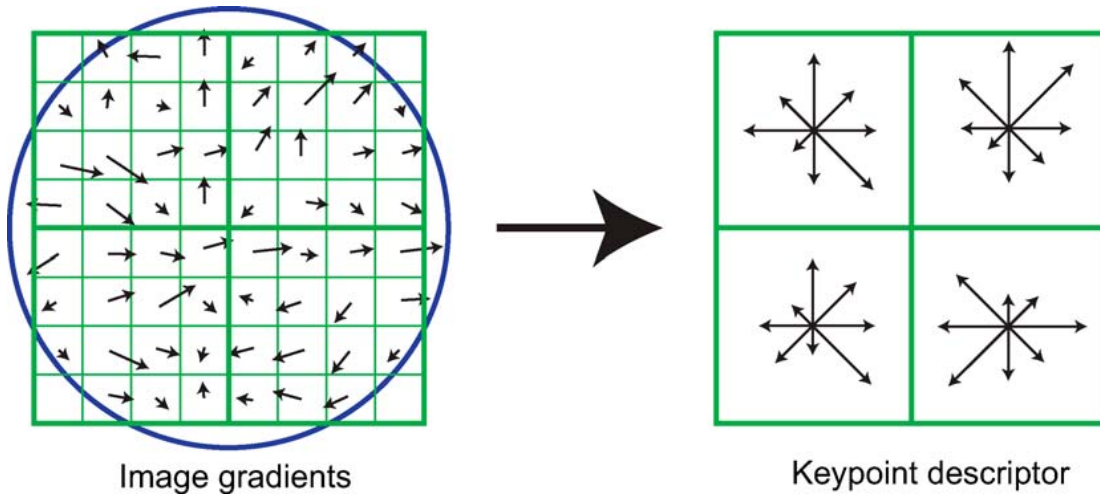


Figure 2.2: SIFT descriptor example using a 2 by 2 patch (Lowe 2004)

Once the descriptor has been calculated, it is normalized using a two step normalization process. First, the descriptor is L2-normalized, then the values of all bins are clamped to a maximum of 0.2, and the descriptor is, once again, normalized with L2-norm. These operations aim to reduce the effects of illumination changes.

A more recent approach, used by Vedaldi and Fulkerson (2008) employs an approximation, which instead of using a Gaussian window to weight the bins, uses a flat window which is then

reweighed by the average Gaussian of all bins. This speeds up the calculation of the descriptor by an order of magnitude, with little or no impact in the classification results.

The strengths of the SIFT descriptor are related to its invariance to scale and rotation of each feature. It is also robust against changes in distortion, viewpoint, noise, and illumination.

2.2.3 HOG

Originally described by Dalal and Triggs (2005) for human detection, the ‘Histogram of Oriented Gradients’ descriptor has been shown to perform very well in scene classification tasks (Xiao et al. 2010; Zhou, Zhou and Hu 2013).

This descriptor represents a picture by calculating an histogram of the orientation of gradients for a small patch of the image. In order to calculate the histograms, a number of orientations is defined, usually ranging from 4 to 16, and the intensity of the gradients for each of the orientations is calculated.

2.2.4 LBP

‘Local Binary Patterns’ is a grey scale and rotation invariant descriptor used for uniform pattern detection (Ojala, Pietikainen and Maenpaa 2002).

LBP works by comparing a central, anchor point, to a number of points along a circle of a pre-defined radius around the anchor. This comparison is a purely binary decision on whether the sampled point is larger than the central value or not, which provides a great invariance to light intensity changes. When the position of one of the points does not match with a pixel, interpolation is used to retrieve the image information.

The number of points and the radius of the circle are the only two parameters of this descriptor, which is represented as a subscript of the LBP name, starting with the number of points and followed by the radius. $LBP_{8,1}$ represents an LBP descriptor with 8 points sampled along a circle of radius 1. See fig. 2.3 for some examples of different sized LBP descriptors.

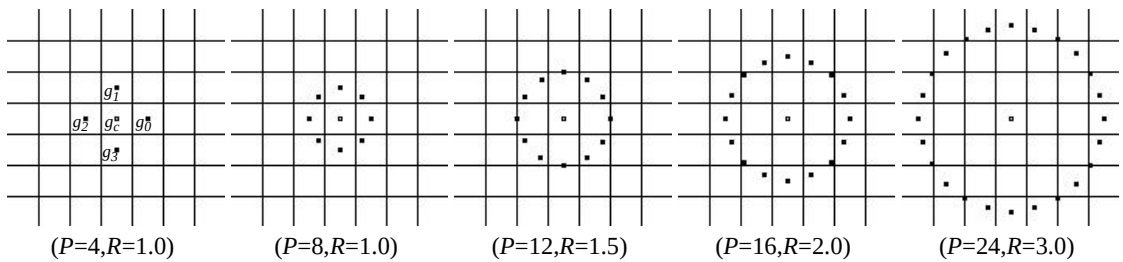


Figure 2.3: Multiple LBP configurations (Ojala, Pietikainen and Maenpaa 2002)

Further improvements are also proposed in order to achieve rotation invariance, by repeatedly applying a circular bit-wise shift, with the objective of minimizing the descriptor value. What this achieves is a descriptor that only cares about the general shape of the pattern being represented, as any rotation to that pattern will be cancelled by the bit-wise shift.

Another improvement proposed by the authors is the removal of descriptors that contain more than two bitwise 0/1 changes. While this operation filters a large number of descriptors, they only account for 10% of the total descriptors in the texture images used by Ojala, Pietikainen and Maenpaa (2002), and thus do not provide enough data for reliable image comparison.

The main advantage of this descriptor is its very small size, which can usually fit in a single integer value. In turn, this means that LBP descriptors can be extracted very densely, up to a descriptor for every pixel in the image, thus reducing the amount of information that may be discarded by other proposed descriptors.

2.2.5 CENTRIST

The newest descriptor, named ‘CENsus TRansform hISTogram’, was proposed by Wu and Rehg (2011) and was designed for place and scene recognition tasks. It is a simplification of the LBP descriptor, corresponding to the $LBP_{8,1}$, but using a scan-line approach to order the bits instead of the circular extraction used by Ojala, Pietikainen and Maenpaa (2002).

This descriptor boasts its lack of parameters to tune, its fast evaluation speed and ease of implementation. The results presented for the fifteen scene dataset, described in section 2.8.1, are on par or superior to those of the more popular SIFT descriptor.

2.3 Extraction Method

There are several techniques that can be applied to feature extraction when using the previous descriptors. Figure 2.4 exemplifies some of the extraction techniques which will be explained in detail in the following sections.

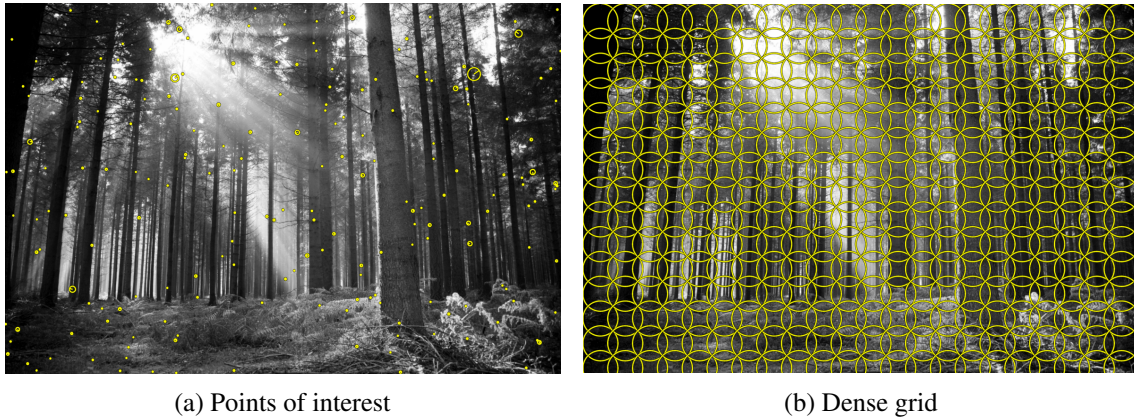


Figure 2.4: Two distinct extraction techniques

2.3.1 Sparse

SIFT descriptors, as defined by Lowe (2004), use a technique to discover points of interest. These points of interest are usually corners or edge points with high gradients which contain useful

information that can be used to compare objects or scenes taken from different viewpoints. The descriptors are then extracted for each of these points of interest.

In order to determine the points of interest a technique called difference-of-Gaussians is used. To apply this technique the image is blurred using several different Gaussian levels and the resulting images are subtracted from each other. The points corresponding to the extrema of these differences are then used as interest points.

2.3.2 Dense

Alternatively, features can be extracted on a dense grid. This involves the division of the image into small patches, wherein a feature is extracted, whether or not the patch contains any gradient or other useful characteristic. The distance between each of these patches, measured in pixels, can also be called ‘descriptor spacing’.

For scene classification, dense extraction of features has been shown to improve the results, since the lack of any significant features can be an important piece of information by itself (Fei-Fei and Perona 2005).

It has also been shown that overlapping the patches where the features are extracted can further improve the scene classification results (Bosch, Zisserman and Muñoz 2008).

2.3.3 Multi-resolution

The multi-resolution technique involves the extraction of features on multiple resolution levels of the image (Wu and Rehg 2011; Zhou, Zhou and Hu 2013). It requires the extraction of features, using either a dense or sparse approach, followed by the halving of the size of the image. This process can be repeated several times, resulting in several sets of features, one for each different resolution. Zhou, Zhou and Hu (2013) identify the use of three levels as an ideal amount of image scales to use in the context of scene classification.

When combined with dense feature extraction, the size of the patch is also scaled down, in order to gather roughly the same amount of descriptors for each image resolution.

An alternative to scaling an image directly is to extract the descriptors in multiple scales, which simulates the image resizing. Scaling descriptors involves calculating the descriptor using a larger or smaller number of pixels around a central point, which results in a descriptor that summarizes more or less information, respectively. Several works have used this technique, improving the classification results when compared to using a single scale descriptor (Bosch, Zisserman and Muñoz 2008; Perronnin, Sánchez and Mensink 2010; Krapac, Verbeek and Jurie 2011). The implementation of this multi-scale feature extraction may differ regarding the placement of the different scale descriptors, having some authors opted for concentric descriptor extraction and others for no alignment between different scales.

2.4 Descriptor Transformation

2.4.1 Hellinger Kernel

Many of the intermediate representations presented in the following section require the comparison of descriptors. To achieve this, techniques such as *k-means* traditionally use L2 distance, also known as Euclidean distance. Using the Euclidean distance has been shown to be less than ideal for the comparison of histogram representations. As descriptors such as SIFT and HOG are histograms themselves better results can be achieved if a more adequate distance comparison is used (Arandjelovic and Zisserman 2012).

A simple way to use another distance measure without rewriting the image processing pipeline is to apply a mapping function to the descriptors. Several approximate mappings for kernels have been proposed by Vedaldi and Zisserman (2010) but one in particular is very useful for descriptor transformation, as it can be applied in place, without changing the size of the descriptor, which is the Hellinger kernel.

The Hellinger mapping, also called RootSIFT when applied to SIFT descriptors, can be calculated in two simple steps. The first step is to normalize the descriptor using L1 distance, also known as Manhattan distance. Then, each element of the descriptor is replaced by its square root.

This simple and cheap to compute transformation has been shown to greatly improve the classification results in scene and object classification tasks (Arandjelovic and Zisserman 2012; Garg, Chandra and Jawahar 2012).

2.4.2 Principal Component Analysis

Principal Component Analysis (PCA), is a technique that can be used to decorrelate and reduce dimensionality of data. While using PCA to reduce the dimensionality of data results in loss of information, this analysis maximizes the compression to loss ratio (Smith 2002).

The first step of performing PCA on a dataset is to center the data so it has mean zero. This allows for the computation of the covariance matrix, which is a square matrix with side equal to the number of dimensions of the data and represents how likely it is for the data in one dimension to change when the data in another dimension also changes. Calculating the eigenvectors and the corresponding eigenvalues of this covariance matrix allows the dimensions to be transformed according to their relevance and importance for the representation of data, as the higher the eigenvalue, the higher the significance of the respective eigenvector. In order to reduce an N -dimensional data into a D -dimensional representation it has to simply be multiplied by a matrix containing the D N -dimensional eigenvectors with the highest eigenvalues.

PCA can be combined with other encoding techniques described in this chapter by using it to process and reduce the dimensionality of the descriptors before using them as one would do with any other descriptor.

2.5 Intermediate Representations

Some of the proposed techniques for scene classification use the extracted features to directly describe an image, such as is the case with NBN described in section 2.7.1. However, new developments in the scene classification literature have shown that using an intermediate representation of the extracted features can boost the accuracy of the scene classification task (Bosch, Muñoz and Martí 2007).

2.5.1 Bag-of-Words Representation

The Bag-of-Words technique, used for text analysis and categorization, has shown excellent performance when applied to scene classification. This technique, as described by Fei-Fei and Perona (2005), is named as Bag-of-Codewords, Bag-of-Features or Bag-of-Visual-Words in the literature.

Bag-of-Codewords abstracts the representation of an image by creating a codebook of features. This codebook is formed by extracting a large number of features from a diverse set of images and then grouping those features into a smaller, pre-defined number of codewords using clustering techniques. A diagram explaining this process can be found in fig. 2.5

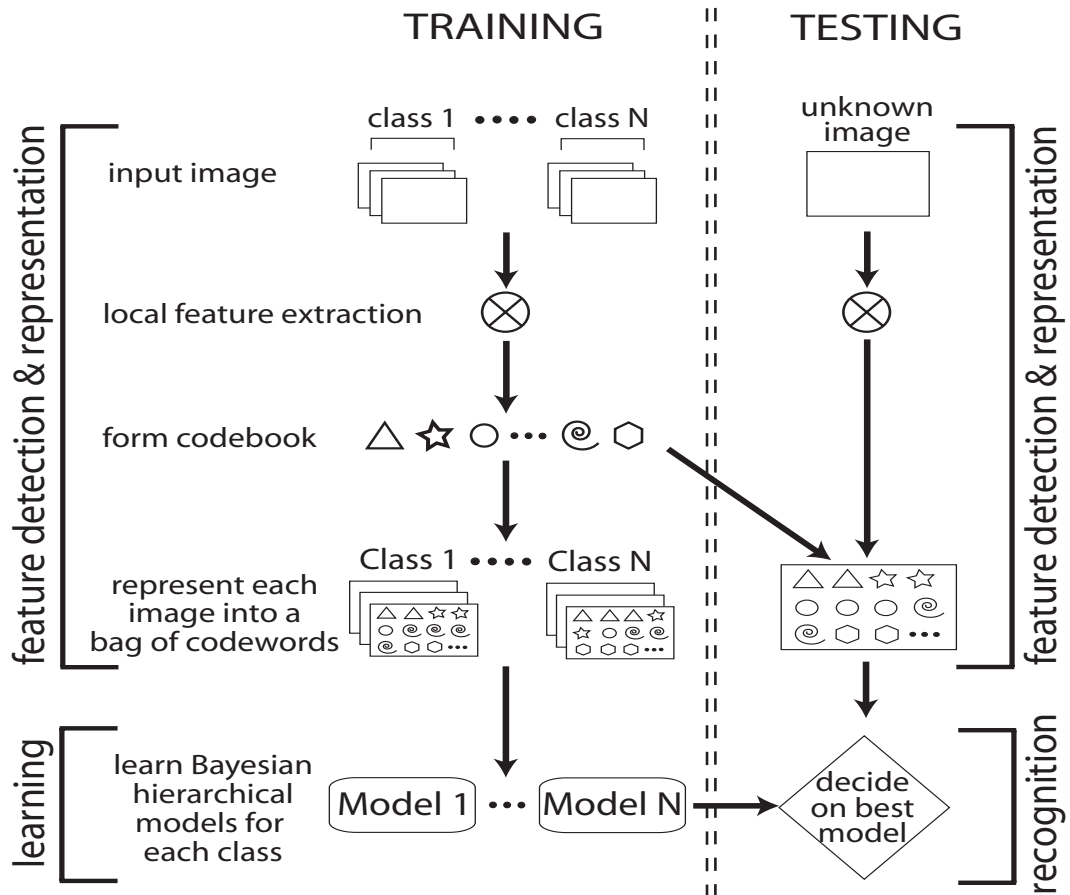


Figure 2.5: Diagram of the bag-of-codewords technique (Fei-Fei and Perona 2005)

Once each descriptor has been assigned to one of the codewords, an histogram of codeword occurrences is built for the entire image. It is this histogram that forms the basis of the comparison between images.

2.5.1.1 k-means Clustering

One of the most popular techniques used to cluster the extracted features of the images is *k-means* (Gao, Tsang and Chia 2013). This technique takes a number of representative features and attempts to calculate a pre-defined number of clusters, minimizing the distance between each cluster centroid and the features assigned to it.

This can be formulated as the solution to the problem described by eq. (2.2) (Yang et al. 2009), where $X = [x_1, \dots, x_M] \in \mathbb{R}^{M \times D}$ is a set of D -dimensional descriptors, $V = [v_1, \dots, v_K]$ are the K cluster centres to be determined by the algorithm, and $\|\cdot\|^2$ is the L2-norm of the vectors:

$$\min_V \sum_{m=1}^M \min_{k=1 \dots K} \|x_m - v_k\|^2 \quad (2.2)$$

The *k-means* algorithm solves this problem by using an iterative, two-step process, as described in algorithm 2.1. The first part of the algorithm is an ‘assignment step’, where each feature is assigned to its nearest cluster centre. This is followed by an ‘update step’, where new cluster centres are determined by averaging the assignments to each cluster. The entire process is then repeated until the centres do not change between two consecutive iterations.

Algorithm 2.1 k-means clustering algorithm

Input:

$X = [x_1, \dots, x_M]$ ▷ data to be clustered
 K ▷ number of clusters

Output:

$V = [v_1, \dots, v_K]$ ▷ cluster centres
 $D(x) = \text{centre closest to } x$ ▷ centre assignments

$V \leftarrow \text{initial cluster centres}$ ▷ e.g. random selection of X

repeat

for all $x_i \in X$ **do**

$D(x_i) \leftarrow \arg \min_{j \in \{1, \dots, K\}} \text{distance}(x_i, V_j)$ ▷ find the closest centre

end for

for $i = 1 \rightarrow K$ **do**

$v_i \leftarrow \text{centroid of } \{x | D(x) = i\}$ ▷ update cluster centres

end for

until V does not change

The main advantages of this algorithm is that not only is it very simple to implement, but it also quickly converges to a local optimum, which means it can be interrupted before fully converging and still return usable cluster centres.

On the other hand, as a clustering technique, it does have several shortcomings. Due to the

way *k-means* uses a simple L2-distance as a measure to find the closest cluster centres, it tends to generate equal sized clusters, which may not be the optimal encoding for the features. Its high dependency on the initial, randomly generated cluster centres, as well as its loss of information, in particular when considering the features closest to the cluster boundaries, are also issues that must be considered when using this technique (Gao et al. 2010).

2.5.1.2 Sparse Coding

Newer alternatives have been proposed to form the codebook using a sparse-coding algorithm to allow for soft assignment of features (Gemert et al. 2008). This means that each feature may be assigned to more than one cluster centroid. Figure 2.6 contrasts the different clustering techniques.

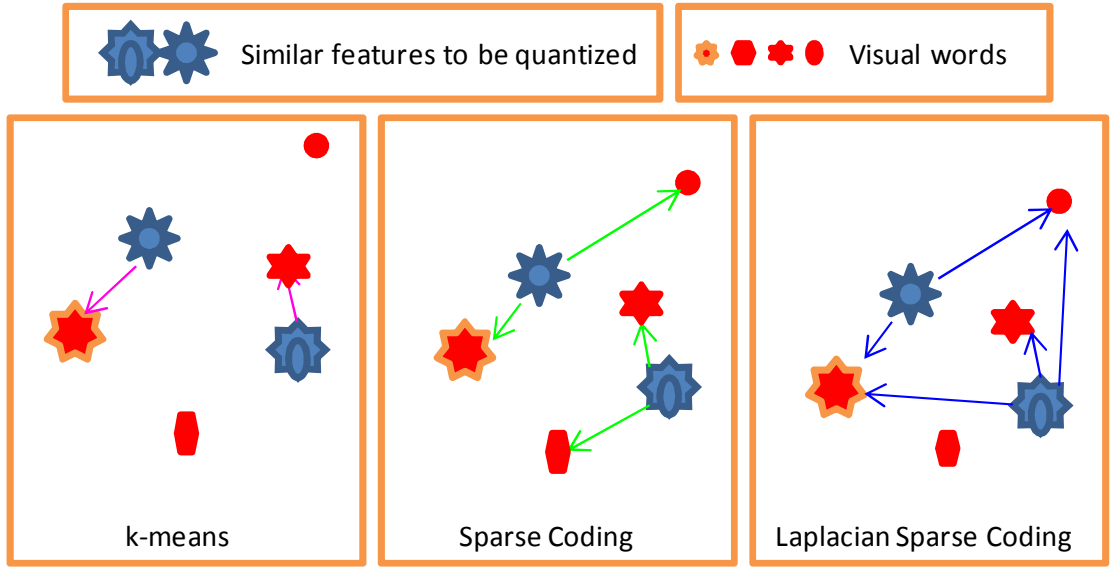


Figure 2.6: Clustering techniques for the codebook creation (Gao et al. 2010)

The sparse coding problem can be seen as a generalization of the hard clustering problem defined in eq. (2.2). Equation (2.2) can be transformed into a matrix factorization problem, having $U = [u_1, \dots, u_K]$ represent the cluster membership, $\|\cdot\|$ is the L2-norm of a vector, $|\cdot|$ is the L1-norm, and $\text{Card}(x)$ is the number of non-zero elements in x :

$$\min_{U, V} \sum_{m=1}^M \|x_m - u_m V\|^2 \quad (2.3)$$

subject to: $\text{Card}(u_m) = 1, |u_m| = 1, u_m \geq 0, \forall m$

By relaxing the $\text{Card}(u_m) = 1$ requirement we can turn the problem into one of soft assignment. In order to limit the number of non-zero assignments, a term containing the L1-norm u_m can be introduced. The importance of this term, which basically represents the sum of the cluster membership assignments, can be tweaked using the factor λ , resulting in the following formulation

for the sparse coding problem (Yang et al. 2009):

$$\begin{aligned} \min_{U,V} \sum_{m=1}^M \|x_m - u_m V\|^2 + \lambda |u_m| \\ \text{subject to: } \|v_k\| \leq 1, \forall k = 1, 2, \dots, K \end{aligned} \quad (2.4)$$

An improvement to the Sparse Coding technique has been proposed by Gao et al. (2010), called Laplacian Sparse Coding, who identify that descriptors that are similar amongst themselves sometimes end up being encoded using a completely different set of codewords. To fix this, they introduce the use of a Laplacian matrix, which characterizes the similarity of local features, in order to ensure similar features are encoded into similar codewords.

This category of techniques, which relax the descriptor assignment into codewords, resulting in less data loss, shows significantly better results over the standard *k-means* when applied to scene classification (Gemert et al. 2008; Gao et al. 2010). This is explained by Boiman, Shechtman and Irani (2008) who argue that, while feature quantization does significantly decrease the feature dimensionality, it also degrades the discriminative power of descriptors. As such, a representation that is closer to the original descriptor, such as Sparse Coding, will result in a more discriminative representation.

2.5.2 Fisher Kernel

The Fisher Kernel is a framework which combines both generative and discriminative approaches to characterise a signal. For the image classification problem, this framework can be adapted by considering an image as the input signal and by using a visual vocabulary as the generative model. For this vocabulary, Gaussian Mixture Models, or GMMs for short, can be used to represent the distribution of low-level features in images (Perronnin and Dance 2007).

GMMs are built by taking a number of descriptors and adapting a pre-defined number of Gaussians to that data. They do this with the help of an expectation-maximization algorithm which iteratively improves the fit between the gaussians and the data.

The kernel itself, which is used to compute the similarity between two Fisher vectors, can be computed using the following equation, where F_x is the Fisher vector for image x :

$$K(X, Y) = F_X^T I^{-1} F_Y \quad (2.5)$$

I represents the Fisher information matrix. Since this matrix can have a very high dimension, it is commonly replaced with a diagonal approximation (Krapac, Verbeek and Jurie 2011).

2.5.2.1 Fisher Vector

The contents of the Fisher vector itself can change depending on what information is included during its calculation. If all information is used, which results in a more detailed representation and

thus better classification results, it contains a total of $K(1 + 2D)$ dimensions, where K is the number of keywords and D is the dimension of each descriptor (Perronnin and Dance 2007). Of these dimensions, K correspond to the weights of each keyword, KD to the average of the descriptors assigned to a keyword, and another KD represent the standard deviation of those descriptors.

The biggest advantage this representation has over that of the Bag-of-Words technique is that, for the same codebook size, it results in a descriptor that is several orders of magnitude larger. This larger, more descriptive representation, allows for images to be classified using a simple linear classifier without any significant loss in performance. Section 2.7 provides additional information regarding classifiers.

2.5.2.2 Vector Normalization

Perronnin, Sánchez and Mensink (2010) determined that the Fisher vector representation contains some image-specific information. This means that two images containing the same subject but different amounts of background information, as is the case in pictures of the same subject in multiple scales, will result in different signatures. To reduce this problem, the Fisher vector can be normalized using L2-norm, allowing for better classification results. This is equivalent to replacing the classifier kernel with the following equation:

$$\frac{K(X, Y)}{\sqrt{K(X, X)K(Y, Y)}} \quad (2.6)$$

While using L2-normalization does improve the classification results, they can be further improved by combining it with another normalization technique. The need for another normalization process arises from the empirical observation that a larger number of Gaussians will result in sparser Fisher vector representations. Perronnin, Sánchez and Mensink (2010) argue that L2-distance is a poor measure of distance for sparse vectors and as such they propose the use of the transformation described in eq. (2.7), which they call ‘power normalization’, in order to reduce the effect of sparsity in the distance calculation, where $0 \leq \alpha \leq 1$ is a parameter of the normalization and z represents each element of the Fisher vector. Through experimentation, Perronnin, Sánchez and Mensink (2010) have determined that $\alpha = 0.5$ provides good results for the 256 Gaussians they used throughout their experiments.

$$f(z) = \text{sign}(z)|z|^\alpha \quad (2.7)$$

In addition to these normalizations, Krapac, Verbeek and Jurie (2011) perform whitening normalization on the data, thus ensuring that each dimension of the Fisher vector has zero mean and unit variance. In order to perform this normalization an additive and a multiplicative normalizer must be determined by calculating, respectively, the mean and the variance in each of the dimensions of the vector, using a set of representative Fisher vectors. The objective of this process is to approximate the Fisher information matrix, thus allowing the vector to be classified using a simple linear classifier. This whitening normalization is itself an alternative to the analytical

approximation of the Fisher information matrix proposed by Perronnin and Dance (2007).

When combined together, the normalizations should be applied by starting with the whitening normalization, followed by power normalization and finally L2-normalization (Garg, Chandra and Jawahar 2012).

2.6 Spatial Information

Once the descriptors have been converted into one of the intermediate representations described in section 2.5 all information regarding the location of the descriptor in the original image is lost. Several techniques have been proposed to reintroduce this spatial information back into the intermediate representation.

2.6.1 Spatial Pyramid Matching

The representation which is used the most often for image classification is the histogram of features or codewords (Lazebnik, Schmid and Ponce 2006). This type of representation provides a generic enough abstraction which will not confuse the classifiers while keeping enough details to enable an accurate distinction among the different categories.

One big drawback of these histograms is that they discard all spatial information associated with each feature. To overcome this problem, Lazebnik, Schmid and Ponce (2006) proposed the partitioning of the image into increasingly finer sub-regions, in a pyramid-like pattern, calculating the feature histogram for each of those regions. The resulting histograms are then used as an image representation. Figure 2.7 exemplifies this process, which is called Spatial Pyramid Matching (SPM).

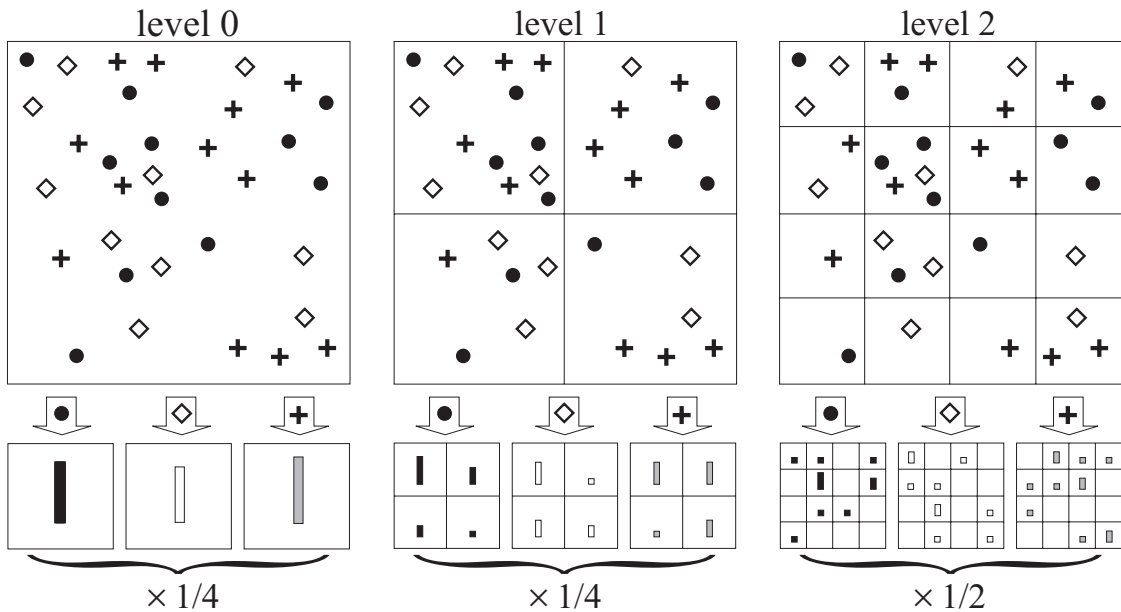


Figure 2.7: Example of a three-level pyramid (Lazebnik, Schmid and Ponce 2006)

Two improvements have been suggested to the Spatial Pyramid Matching proposed by Lazebnik, Schmid and Ponce (2006). The first one, described by Wu and Rehg (2011) uses overlapping sub-regions by shifting the division grid in order to reduce artefacts caused by non-overlapping regions. The second was proposed by Zhou, Zhou and Hu (2013) and uses vertical and horizontal slices instead of the square partitions of SPM, which achieves better results since many natural and man-made features, such as the horizon, trees, and buildings are located within these vertical or horizontal slices.

2.6.2 Spatial Fisher Vector

Spatial Fisher Vectors, based on the same model used by the Fisher Framework, are an alternative to the Spatial Pyramid Matching. They can be used to model the spatial information of an image's descriptors by calculating the mean and variance of the location of each descriptor assigned to a given codeword (Krapac, Verbeek and Jurie 2011).

Figure 2.8 compares this technique to Spatial Pyramid Matching. Along with the frequency of each codeword, represented on the left as an histogram, these techniques also capture spatial information. The Spatial Pyramid shown here divides an image into four sections and creates a codeword histogram for each section. SVF stores the average location and deviation of each codeword instead, which is represented by ellipses in the diagram.

While this kind of spatial representation is more difficult to implement and compute, it results in significantly smaller spatial representations. This is even more apparent when the size of the intermediate representation increases, as is the case with using Fisher vectors instead of Bag-of-Words.

2.7 Learning Techniques

Given a well defined picture representation, a process to distinguish between pictures of different classes must be used. The two most used techniques in scene classification are k-nearest neighbours and support vector machines (Bosch, Muñoz and Martí 2007).

2.7.1 kNN

The k-nearest neighbour classifier attempts to define the class of an unknown image by selecting the k nearest images from the training database, using euclidean distance. The class of the new image is then defined as the one that appears the most times on its nearest neighbours (Bosch, Zisserman and Muñoz 2008). For binary classification k is usually an odd number, in order to prevent ties.

The greatest strength of this classifier lies in its speed, as it requires no training step, and ease of implementation, while achieving a classification rate close to those of more complex methods (Bosch, Zisserman and Muñoz 2008; Zhang et al. 2006).

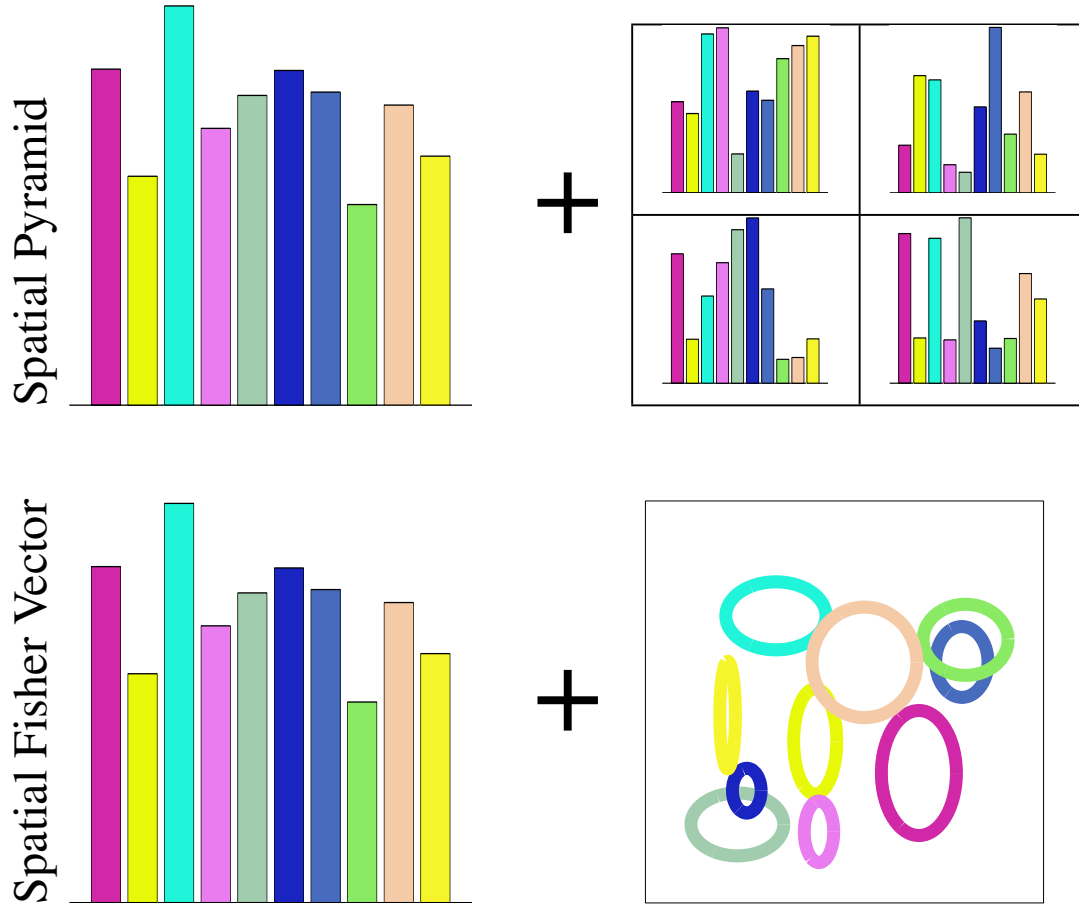


Figure 2.8: Comparison of SPM and SFV (Krapac, Verbeek and Jurie 2011)

In order to further boost the accuracy of the kNN classifier, Boiman, Shechtman and Irani (2008) proposed the introduction of some improvements to the technique, calling it ‘Naive Bayes Nearest Neighbour’, or NBNN. One such improvement involves the calculation of a image-to-class distance instead of the more commonly used image-to-image distance. Using this distance measure improves the generalization capabilities of the kNN classifier, in particular with a low number of training samples. The other improvement is based on the direct use of the original descriptors, without any kind of feature quantization step. They argue that the intermediate representations reduce the descriptive power of the descriptors and negatively impact the correct classification rate of new images.

Tuytelaars et al. (2011) raised some questions regarding the capability of the proposed nearest neighbour framework to classify certain types of images. They exemplify this problem with the image of a tire which would get classified as a car with high confidence since all of the features of a tire have a small distance to the car class, which also contains tire descriptors. Images which would be hard to distinguish using this technique are shown in fig. 2.9.



Figure 2.9: Images that NBNB has difficulty distinguishing (Tuytelaars et al. 2011)

2.7.1.1 k-d Tree

One technique often used to perform a fast nearest neighbour search, and which does not involve the calculation of the distance between the descriptors of an image and every other descriptor, is using k-d trees, or k-dimensional trees (Beis and Lowe 1997). These are binary space partitioning trees, which achieve an average search complexity of $O(\log n)$.

The way these trees are built is by taking the dimension with the highest variance and splitting all descriptors into two sets, based on whether their value on that dimension is higher or lower than the median value of all features.

In order to further boost the performance of k-d trees, at the expense of higher memory use, Silpa-Anan and Hartley (2008) proposed the use of multiple trees at once. This allows for parallel search of nearest neighbours among those trees, a technique that is further validated by Muja and Lowe (2009), who show that it performs well over a large range of problems.

While this data structure allows for very fast neighbour search, it requires all descriptors to be kept in memory, which may not be feasible for larger image representations or when a vast number of training images is required.

2.7.2 SVM

The Support Vector Machine is a binary classifier often used in scene classification (Bosch, Muñoz and Martí 2007).

In order to be able to perform non-linear classification, a kernel, which maps the inputs into an high-dimensional feature space, can be used. Sections 2.7.2.1 and 2.7.2.2 detail the differences between linear and non-linear SVM classifiers.

Since the SVM is inherently a binary classifier, when multi-class classification is required, an auxiliary technique must be used. The one-vs-all technique is extensively used for scene classification, where one SVM is trained for each class, using that class versus every other class as a training dataset (Bosch, Zisserman and Muñoz 2008; Zhou, Zhou and Hu 2013). An alternative is to use one-vs-one approach, which involves training one SVM for every pair of classes. While this results in a larger number of classifiers, each one of them is much faster to compute since the number of images used to train each classifier is much lower (Chang and Lin 2011).

For scene classification SVMs have been shown to perform better than kNN, at the cost of

being more computationally expensive (Bosch, Zisserman and Muñoz 2008).

2.7.2.1 Linear SVM

Linear SVMs try to split a labelled dataset by dividing it using an hyperplane. If the two labels are linearly separable then this plane divides the data in such way that the margin between both sets and the plane is maximized. If, on the other hand, there is an overlap between the two dataset and it is not possible to cleanly split them, then an additional parameter is added, usually called C , that defines how important it is to minimize the number of outliers versus maximizing the margin of the hyperplane.

This problem can be solved with eq. (2.8), given a set of data-label pairs $(x_i, y_i), i = 1, \dots, l, x_i \in \mathbb{R}^n, y_i \in \{-1, +1\}$ (Fan et al. 2008).

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^l \xi(w; x_i, y_i) \quad (2.8)$$

$\xi(w; x_i, y_i)$ is a loss function, for which it is common to use either eq. (2.9), called L1-SVM, or eq. (2.10), for a L2-SVM.

$$\max(1 - y_i w^T x_i, 0) \quad (2.9)$$

$$\max(1 - y_i w^T x_i, 0)^2 \quad (2.10)$$

Once the model w has been trained, a new data point x can be classified by calculating $w^T x$. A positive value gets placed in the label $+1$, while any other value gets labelled as -1 (Fan et al. 2008).

2.7.2.2 Non-linear SVM

Sometimes it is not possible to split the feature space using a simple hyperplane, in particular when the features have reduced dimensionality. To overcome this problem, the features can be mapped into a higher dimensionality space where it becomes easier to divide the features and train the classifier. To this end, a kernel function is used.

Several kernels have been successfully used in scene classification, with the most popular being the radial basis function, the histogram intersection, and the χ^2 kernels, which are calculated using eq. (2.11), eq. (2.12), and eq. (2.13), respectively. Kernel functions are usually typeset as $k(x, y)$, where x and y are vectors containing the intermediate representations of two images.

The radial basis function kernel can be implemented using a Gaussian kernel, which requires the optimization of the σ parameter by using cross-validation:

$$k(x, y) = \exp(-\sigma \|x - y\|^2) \quad (2.11)$$

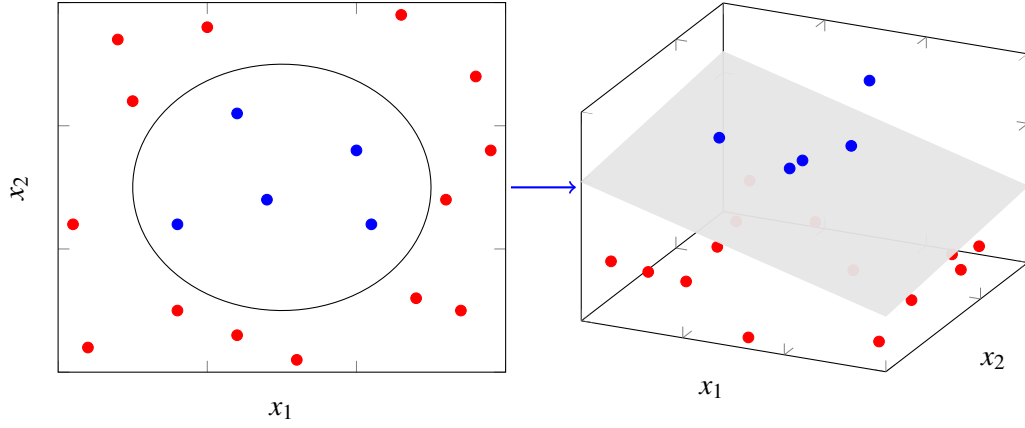


Figure 2.10: Example of using a kernel to map data into a higher dimensionality space

As the name implies, the histogram intersection kernel is particularly effective when the data being compared represents an histogram, be it the result of an encoding process, like the ones described in section 2.5 or a descriptor, like SIFT or HOG, which are themselves histograms of gradients. This kernel does not have any parameters and can be easily calculated using the following equation (Lazebnik, Schmid and Ponce 2006):

$$k(x, y) = \sum_{i=1}^n \min(x_i, y_i) \quad (2.12)$$

The χ^2 kernel has also been shown to be a viable alternative to the histogram intersection kernel when used for image classification (Zhou, Zhou and Hu 2013):

$$k(x, y) = 1 - \sum_{i=1}^n \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)} \quad (2.13)$$

The trade-off for having the higher descriptive power and better classification results of a non-linear classifier is that the computational cost for the classification of new data is much higher. It now requires the computation of the kernel function for every pair of new images and training data before being able to classify it, instead of applying a simple dot product between the image representation and the model determined during the training phase. This translates into a training complexity of $O(n^2 \sim n^3)$ and classification complexity of $O(n)$, with n being the number of training images, instead of the $O(n)$ training complexity and $O(1)$ classification complexity of linear SVMs (Yang et al. 2009).

2.7.3 kNN-SVM

While not extensively used in the scene classification field, a combination of both kNN and SVM has been proposed, harnessing the speed and simplicity of kNN classifiers while making full use of the more accurate classification provided by SVMs (Zhang et al. 2006).

2.8 Datasets and Results

In order to train, test, and compare the results with other works in the literature, a standard dataset should be used. For the scene classification literature, the most used dataset is composed by fifteen different classes. A newer and larger alternative dataset has also been proposed. The following sections describe both of these datasets.

2.8.1 Fifteen scene dataset

In this dataset, eight categories were gathered by Oliva and Torralba (2001), which correspond to tall building, inside city, street, highway, coast, open country, mountain, and forest. Fei-Fei and Perona (2005) extended this dataset by adding the classes office, kitchen, living room, bedroom, and suburb. Finally, two more categories were added by Lazebnik, Schmid and Ponce (2006), which are industrial and store. Figure 2.11 shows some example pictures of each category of the dataset. In total, this dataset contains between 200 and 400 images per category, with an average resolution of 300×250 pixels. In this dataset, only the eight original categories are available in color, while the remaining pictures are in grayscale.



Figure 2.11: Example pictures of the fifteen scene dataset

Table 2.1 shows the mean per-class classification accuracy obtained by several works in the scene classification literature for the fifteen scene image dataset. The best result is highlighted in bold.

Table 2.1: Classification results for the fifteen scene dataset

Author	Result (%)
Bosch, Zisserman and Muñoz (2008)	83.7
Gao, Tsang and Chia (2013)	89.8
Lazebnik, Schmid and Ponce (2006)	81.4
Wu and Rehg (2011)	83.9
Zhou, Zhou and Hu (2013)	84.2

2.8.2 SUN database

A more challenging dataset has been compiled by Xiao et al. (2010), which contains 899 categories and 130519 images, of which 397 well-sampled categories are used to compare the results of several scene classification techniques. This dataset is organized hierarchically, with the topmost classification being indoor, outdoor natural, and outdoor man-made. The results of several scene classification techniques for this dataset can be found in fig. 2.12.

2.8.3 Vogel & Schiele

A third dataset used in scene classification, was proposed by Vogel and Schiele (2007). This dataset contains a total of 700 natural images, all of them with a size of 720×480 pixels. Since this dataset does not contain man-made images it is not as useful in the context of this work.

2.9 Literature Shortcomings

While there are many works on the field of scene classification, and even more if peripheral areas of study are also considered, such as object recognition, these works are missing a great deal of information that is required to draw conclusions regarding their ability to classify a large number of natural and man-made images. The following sections show some of the gaps that the present work intends to fill.

2.9.1 Degree of Naturalness

Very few authors in the computer vision literature directly tackle the problem of natural versus man-made image classification. Some who include this type of classification, such as Oliva and Torralba (2001), do not focus exclusively on this task and are usually dated. Considering the extremely large intra-class variation of both man-made and natural categories, this classification problem poses a different set of challenges to scene classification techniques which may not be correlated to the results presented on other datasets.

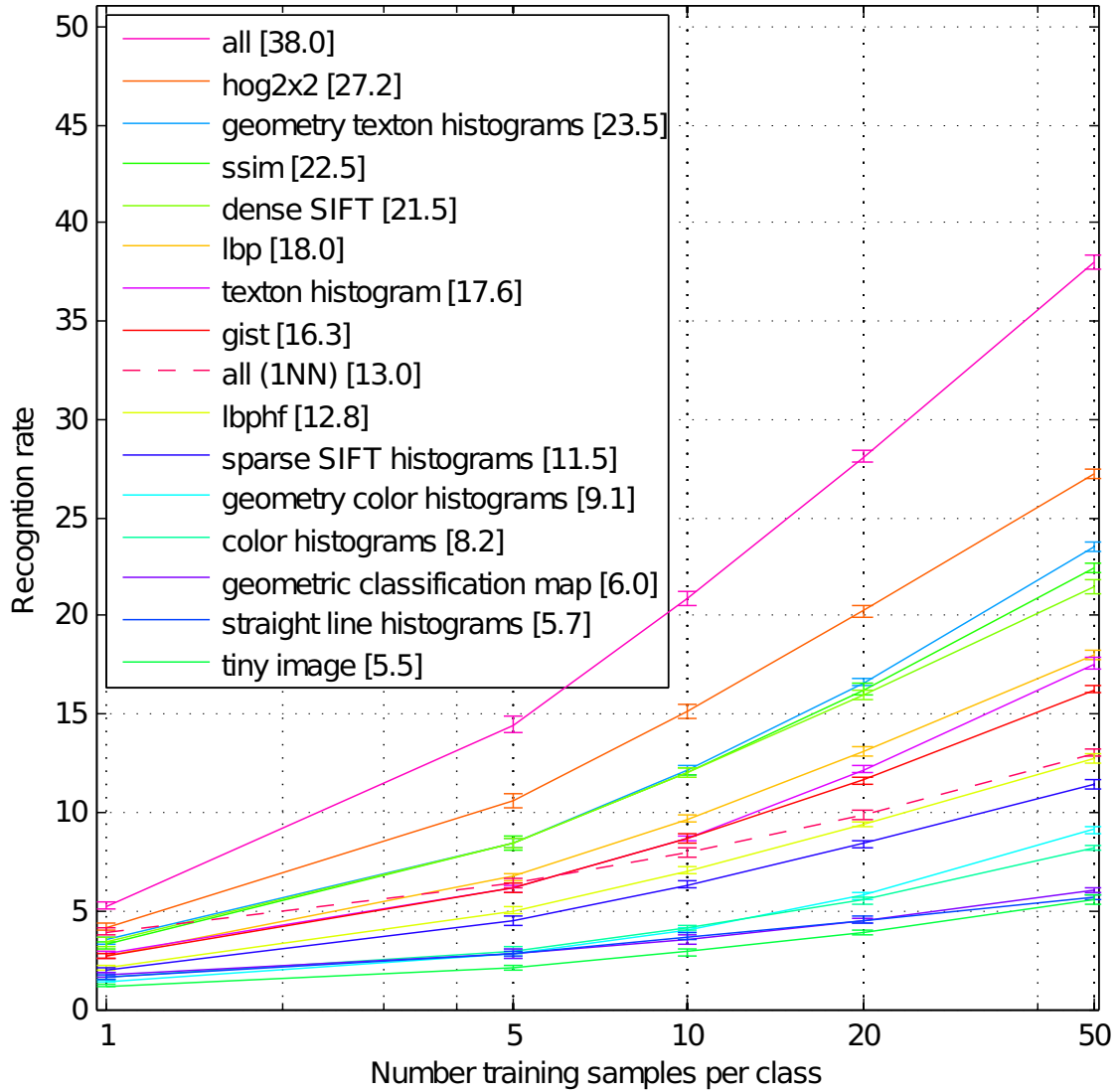


Figure 2.12: Classification results for the SUN database (Xiao et al. 2010)

2.9.2 Performance Evaluations

Few works in scene classification report the computation time or algorithmic complexity of the proposed technique. The popular Pascal VOC Challenge, which pitches multiple competing classification solutions, illustrates this problem, as it only takes into account the achieved classification accuracy, no matter how long it takes to process the data (Everingham et al. 2010). This results in a competition for the best results in a given dataset without taking into account the computational cost of applying the proposed algorithm. In order to take these algorithms from the scientific community and move them into industrial applications an extensive computational cost versus results analysis must be performed.

2.9.3 Comparing Improvements Separately

It's very rare for a work in scene classification to propose a single, isolated improvement. Without an individual comparison of each improvement against established techniques, it becomes quite difficult to determine which of the proposed changes directly resulted in an actual accuracy increase and which only apply when combined with other improvements.

Scene Classification

Chapter 3

Classification Framework

Using the knowledge gathered from other scene classification works it is possible to design a framework capable of integrating many distinct image classification techniques. The objective of such image classification framework is to help in the research of new techniques and comparison against current solutions, evaluating them using a variety of datasets, as well as allowing for an easy transition into a real case scenario, where it needs to be used as part of a larger application that requires images to be classified.

This section presents all the details of the framework that was built to classify images according to these objectives, including all the functionalities and techniques that were implemented as well as all third-party libraries that were used.

3.1 Functionality

The analysis of the goals of the project, presented in section 1.3, reveals a need for two different types of feature sets. When used as a research tool, the framework has a unique set of requirements that must be fulfilled, in order to provide an environment for quick and statistically significant data gathering related to the ability of certain algorithms to classify a given dataset:

- To automatically split a dataset into a random set of training and testing images. The training set should contain an equal number of random images per class.
- To report the overall classification accuracy for a dataset after classifying all testing images. This summarizes the success of a given technique at classifying images.
- To show a detailed confusion matrix of the multiple classes after each experiment. With this, it is possible to figure out which classes are being confused by the classifier and point towards a possible path for further research.
- To allow for the execution of the same experiment a predefined number of times in order to help achieve statistically significant results. The results of the multiple runs should then be reported as the mean and standard deviation of the experiment results.

When moving from a research environment to a production environment, where new images of unknown classes must be classified and the data formatted for further analysis or processing, a new set of requirements can be identified:

- Classify unlabelled images and identify their class. This output allows images to be then organized or further processed by external tools.
- Attempt to provide a probability associated with the identified class. Using this measure allows images to be filtered according to their naturalness rating.
- Measuring the time required to classify an image is also important, as it allows settings to be tweaked to achieve the desired time and accuracy balance.

All of the reported requirements were fully implemented into an open-source tool, which allows other researchers to expand and easily compare their algorithms to current scene classification solutions, as well as allowing anyone who requires an automated image classification tool to easily use it without having to change any source code. This tool can be found in GitHub¹.

3.2 Features

To boost the usability and overall usefulness of the framework, a few additional features were added. They are described in detail in the following sections.

3.2.1 Flexibility

Assuming its settings are tweaked to the problem at hand, the developed framework can be used for many different image classification tasks. As such, it is ideal to allow for these settings to be easily modifiable, even by those with no previous computer vision knowledge, as well as allowing the dataset to be swapped by others with different classes and content.

In order to achieve an easy to configure framework, all relevant parameters were moved to a separate configuration file which uses JSON to represent the data. Having all settings in a single file and being able to quickly tailor those settings to the current dataset, without delving into the code, greatly improves the framework adaptability. Listing 3.1 shows what a configuration file looks like.

The parameters are grouped into five groups, corresponding to the general steps of image classification: image loading and processing, feature extraction, feature encoding using a codebook, spatial information encoding, and classification. Changes to parameters in one of these groups has repercussions in the groups that follow, while having no effect on the previous groups. Appendix B explains the meaning of each parameter as well as the acceptable range of values for each one.

To make it straightforward to experiment with new datasets, a simple system was deployed to make it easier to replace the datasets where the experiments are executed. In order to create a

1. <https://github.com/uneqwcmc/DetectingNature>

```

1 {
2   "image": {
3     "type": "Greyscale", //Greyscale, Opponent, HSV
4     "maxResolution": 250,
5     "forceSize": false,
6     "smoothingSigma": 0.0
7   },
8
9   "features": {
10    "type": "SIFT", //SIFT (128 dim), HOG (36 dim), LBP (8 dim)
11    "gridSpacing": [10],
12    "patchSize": [20],
13    "transforms": ["Hellinger"] //Hellinger
14  },
15
16  "codebook": {
17    "type": "Fisher", //Fisher, KMeans
18    "codewords": 50,
19    "pcaDimension": 80,
20    "textonImages": 1500,
21    "totalFeatures": 500000
22  },
23
24  "histogram": {
25    "type": "Slices", //Slices, Squares
26    "pyramidLevels": 2
27  },
28
29  "classifier": {
30    "type": "Linear", //Linear, SVM
31    "c": 10.0,
32    "trainImagesPerClass": 100
33  }
34 }

```

Listing 3.1: Example configuration file

new dataset, a sub-folder for each of its classes is placed inside a root folder. Then, the images are simply dropped inside the sub-folder of their respective class. The framework will automatically recognize each sub-folder as a different class, regardless of how many there are, and will use the name of the folder as a label for the images. This also means that multiple datasets can be kept in separate folders and switching between them can be easily done using the command line parameters of the classification framework.

3.2.2 Caching

A change in the descriptor type means that all steps executed after feature extraction, such as converting an image into an intermediate representation and training a classifier, must be recalculated from scratch. On the other hand, a simple change to the classifier misclassification penalty setting does not require any of the data to be recalculated, with the exception of the final classifier.

Recomputing all these expensive steps means that trying new settings would require a long training and classification time. To counteract these redundant computations, a caching system can be implemented by serializing the data that results from each intermediate step as well as the settings used to generate that data. With this caching system, settings can be changed between multiple runs and no data will be unnecessarily recomputed.

3.2.3 Extensibility

Since testing and comparing multiple techniques for scene classification was one the main requirements, it was desirable to build a very modular and easy to extend framework.

The creation of such a flexible framework was made possible by the creation of algorithm-independent data structures, using classes such as ‘Histogram’ or ‘Image Descriptors’, allowing for a number of different algorithms to be chained together easily.

In order to keep dependencies between the different steps of the scene classification process at a minimum, the inversion of control technique was used throughout the framework. This technique allows for dependencies to be injected into another object at runtime, thus removing the need to explicitly create a hard dependency between two different classes.

3.2.4 Parallelization

Classifying images usually involves the processing of many images at a time, be it during the training phase, where a sizeable number of example images are required, or during the classification of new images.

Since most parts of the scene classification pipeline do not depend on more than one image, with the exception of some intermediate representations and classifiers, it becomes trivial to reduce the time required to process the images by simply processing them in parallel. Taking advantage of the computer’s multiple cores means the overall scene classification time can be slashed by a factor proportional to the number of cores.

Additional parallelization can be achieved using ‘Single Instruction, Multiple Data’ or SIMD which allows for the transformation of large vectors of data using a single CPU instruction, thus improving the overall performance of the application.

3.3 Libraries

Several libraries have been previously used in other computer vision and machine learning works. Reusing these libraries has several advantages, such as:

- allowing other works to match all the implementation details, which are often not described or detailed in the original articles;
- containing a lot of non-trivial optimizations and speed improvements, over a naive implementation;
- their open-source nature, together with their use by many authors, improves the robustness of the code and reduces the chances of introducing an error that will decrease the accuracy of the scene classifier.

The following subsections describe the major libraries used in this work, as well as the reasoning behind their inclusion over possible competing libraries.

3.3.1 CImg

A combination of ease of use and feature completeness make the CImg library a good choice for image loading and preprocessing. This library is used in this work to load images, resize them when they are too large for processing in a timely manner, and for image blurring.

Several alternatives to this library exist, but they are either lower-level libraries, containing no image post-processing, such as FreeImage, or they contain too many dependencies and extraneous features such as is the case with OpenCV.

3.3.2 VLFeat

VLFeat is a library that implements several computer vision algorithms (Vedaldi and Fulkerson 2008). Among the implemented techniques, it contains SIFT, Dense SIFT, HOG, k-means, and k-d trees which were used in this work. VLFeat is more flexible than OpenCV, allowing for fine tuning of parameters such as the step and size of descriptor patches, which makes it a better choice for experimentation on novel computer vision applications.

3.3.3 Boost

Boost is a compendium of libraries that extend the C++'s Standard Template Library by providing additional data structures and algorithms.

The main Boost functionalities used in this work are related to its platform-independent file system traversing library, which is used to list all the files in a dataset, as well as its serialization library, which helps with the implementation of a caching system to save processed data, thus significantly reducing the computational cost of repeated experiments over the same dataset.

3.3.4 Yael

Yael is a library that collects many computer vision algorithms, and which is optimized for multi-threaded processing. Along with a multi-threaded implementation of k-means it also includes sev-

eral SIMD operations for data in arrays, which is used to perform fast operations over descriptors and other intermediate representations.

3.3.5 LIBSVM

LIBSVM implements Support Vector Machine classifiers and supports the use of kernels for non-linear classification (Chang and Lin 2011). It can also be used as both a command line tool or as a library which makes it easier to experiment with SVMs before committing to a full integration of this classifier.

Its ability to use pre-computed kernel matrices makes it the perfect candidate for testing new kernels for scene classification, which when combined with its superior training performance, puts it above competing implementations.

3.3.6 LIBLINEAR

LIBLINEAR, like LIBSVM, provides an implementation of Support Vector Machines (Fan et al. 2008). The difference between them is that LIBLINEAR is optimized for linear classification and thus does not support the use of any kind of kernel. Consequently, this allows for much faster training and classification of large amounts of data, and is more appropriate for data representations with a large number of dimensions.

3.3.7 GMM-Fisher

While the Yael library also implements Fisher vectors and Gaussian Mixture Models, its implementation of the GMM component does not always converge, with its accuracy fluctuating throughout the training process and requiring it to be stopped after a certain number of iterations to progress with classification.

GMM-Fisher, a library originally developed by Perronnin, Sánchez and Mensink (2010) and shared as a sub-library in the implementation of Chatfield et al. (2011), always converges, usually achieving the same accuracy as Yael using only a small fraction of the iterations, significantly reducing the time required to complete the training phase of the classification framework.

For better results, GMM-Fisher requires manual initialization of the Gaussians, which is usually achieved using k-means. For this, the Yael implementation of k-means is used.

Chapter 4

Methodology

This chapter describes in detail the experimental protocol used to compute all the results presented in chapter 5. The dataset used, all limitations and other details that may influence the classification results are all presented here. The aim of this section is to ensure all experiments and results can be easily reproduced without any guesswork involved.

4.1 Objectives

The experiments were designed in order to evaluate whether the goals defined in the introduction of this work could be achieved using the framework described in chapter 3. To do this, the experiments contain not only tests related to the multiple algorithms in the scene classification literature, on their parameters, and computational cost, but also test the framework on a higher level regarding its applicability in the context of automated image validation and for the creation of a map. The results of these experiments are further analysed in chapter 6.

4.2 Experimental Set-up

In order to evaluate each of the implemented techniques, the dataset used is split into two sets: a group of training images and another of test images. The training set is collected by randomly selecting a predetermined number of images for each class, while the testing set contains all remaining images. As this random split influences the final classification results, each experiment is executed a total of six times with a random train and test split, with the results reported as the mean and standard deviation of all six experiments. The training and testing split do not overlap, which means that none of the training images are used during the testing phase.

The performance results for each experiment are based on the classification accuracy over the testing dataset. This accuracy is calculated as the number of correctly classified images divided by the total number of classified images. As several datasets contain an imbalanced number of images in each class, it is important that the result of each experiment is reported as the mean per-class accuracy. The accuracy is calculated for each class, independently, and the final result is

then the average score for all classes. Simply calculating the accuracy for all images may provide undesirable results when the number of images in each class is unbalanced.

Table 4.1 shows an example of such scenario, containing a total of 1000 images, 900 of which belong to class A and the remaining 100 belong to class B. If a classifier simply assigns all images to belong to class A, calculating the total accuracy will result in 90% for this dataset. Using the mean per-class accuracy will result in classification score of 50%, bringing it in line with the chance accuracy, and more accurately reporting the expected classification results.

Table 4.1: Classification results where mean per-class accuracy is better

	Classifier guessed A	Classifier guessed B
Real class is A	900	0
Real class is B	100	0

Not all the reported results are calculated using the optimal settings, as that would take a prohibitive amount of time to run all the experiments the desired six times.

While all of the explored techniques were also applied to the 15-scene dataset, their results are not discussed here, as they match the results described in the original works presented in chapter 2.

4.3 Definition of Natural

Given the context of this work, the definition of what is and isn't a natural image is very important, as different interpretations for the concept of naturalness can be proposed.

For these experiments, a natural scene is defined as one that does not permanently contain any elements that only occur in nature with human interference. This definition excludes elements that do not permanently affect the naturalness of a scene, such as is the case of camping tents or boats. On the other hand, this means that images that contain natural elements, but which are of man-made origin such as crops or gardens, will be considered as man-made scenes.

This definition is in line with the objective of identifying which areas of the world are being affected by humans and which are still in their natural and unaffected state.

4.4 Default Values

Unless otherwise stated in an experiment, each parameter takes its default value described in table 4.2. These settings were chosen, based on previous experimentation, as a good balance between accuracy and computational cost.

Table 4.2: Default settings used during the experiments

Setting	Value
Image colour	Grey-scale
Image scaling	Maximum side of 250px
Image smoothing	Disabled
Descriptor type	SIFT
Patch size	[20, 24, 28, 36, 40, 48, 60, 72]
Grid step	[10, 12, 14, 18, 20, 24, 30, 36]
Transformations applied	Hellinger and PCA
Final PCA dimension	80
Codebook type	Fisher vector
Number of codewords	50
Sample images in codebook	100
Maximum descriptors in codebook	500000
Spatial representation	None
Classifier type	Linear SVM
SVM misclassification penalty	10.0
Training images per class	50

4.5 Performance Analysis

All computational costs reported in the experiments described in the next chapter represent the results of tests that were executed in a machine with an Intel® Core™ 2 Duo E7200 CPU at 2.5GHz with 4Gb of RAM.

While the images are processed in parallel to take advantage of the multi-core CPU architecture, all timings are reported based on how long it takes to process an image on a single core. In practise, when classifying multiple images, this time can be divided by the number of cores in the CPU, as image classification is a highly parallel problem.

4.6 New Dataset

Inspired by the original work of Oliva and Torralba (2001), who divided their original 8-scene dataset into two classes containing man-made and natural images, an initial attempt was made to experiment with this same approach using the larger 15-scene dataset. Unfortunately, using such dataset as a basis for natural versus man-made image classification analysis was not viable, as initial experiments resulted in over 98% accuracy due to the lack of image diversity in this dataset. With such an high level of correct classifications it becomes difficult to accurately compare the effect of each new technique. To offset this and to clearly observe the impact of each change, a more challenging dataset must be used.

The SUN Database, containing images from almost 400 categories, is the ideal candidate, as

each of those categories is also labelled as indoor, outdoor man-made or outdoor natural. Combining all categories labelled as indoor and outdoor-manmade into one large man-made set, as well as using all categories labelled as outdoor natural as a large natural category, results in a natural vs man-made dataset containing over 130 thousand images. Initial experiments in this new dataset revealed a drop in accuracy to about 80% correct classifications.

This more diverse and challenging dataset poses another, very different, kind of problem. As it contains over 35GB of image data to be processed it may take over a day to run a single experiment on all images. As such, a smaller dataset based on the larger SUN Database was created, containing a subset of 2000 images for each class. To ensure this subset retained its diversity and did not affect the reported results, several experiments were executed on both the larger dataset and this new subset. All results matched without any statistically significant differences.

While experiments were also executed in the original datasets described in section 2.8, in order to validate the correctness of the implemented framework, since their results matched those in the original research, they are not reported in this work.

Empirical results show that this smaller subset contains enough image variety that the classification results remain unaffected when compared to classifying the entire SUN Database, despite taking only a small fraction of the time to process.

It is worth noting that this dataset contains images with disputable label. Examples of this are images containing camping tents and boats, which are man-made but do not necessarily change the background's class, and crops, which have natural content, but were placed there by humans. Figure 4.1 shows some example images with disputable labels. As such, for any production environment that requires the classification of natural versus man-made images, the direct use of this dataset should be avoided. Instead, a custom built dataset should be used in order to avoid any ambiguities or misinterpretations of what does and does not constitute a natural image.



(a) images defined as natural in the dataset



(b) images defined as man-made in the dataset

Figure 4.1: Images in the SUN Database with disputable labels

4.7 Limitations

4.7.1 Execution time

In order to get statistically significant results, each experiment must be executed several times. Unfortunately, due to the time it takes to fully train a classifier, which can range from 10 minutes up to four hours in each of the executed experiments, it is not feasible to explore every possible combination of techniques and parameters.

Before executing all the experiments reported in the next chapter, small scale trials were performed to determine which parameters had an effect on other parameters, thus allowing the final experiments to focus on the useful settings combinations.

4.7.2 Implementation complexity

Due to the complexity of several algorithms described in chapter 2, along with the time constraints this work was subject to, only the most promising techniques were chosen to be implemented and compared during this study. See appendix A for a full listing of the implemented techniques.

This means that techniques such as the Gist descriptor, which was proposed over 10 years ago and has been surpassed multiple times in more recent works, or Laplacian Sparse Coding, which is very challenging to implement efficiently, were not part of this study. Instead, widely supported descriptors such as SIFT were used, and alternative techniques such as the Fisher framework replace Sparse Coding techniques, since they are mutually exclusive, provide similar results and are significantly easier to implement.

4.7.3 Memory allocation

Certain technique combinations were not fully explored due to their massive memory requirements when combined together. The following list shows some of the worst offenders when it comes to memory consumption.

- Using colour descriptors requires additional memory for each colour channel. Since these usually contain three channels, they require three times more memory to process when compared to grey-scale images.
- During the training phase of k-means and GMM, all descriptors must be kept loaded in memory, creating an upper limit on the number of training images that can be used. This limit is further reduced when the size of each descriptor increases, as described in the previous point.
- Nearest Neighbour classifiers, depending on which technique is used, require either all intermediate representations or all descriptors of the example images to be kept in memory during the classification process.

Methodology

- Calculating the kernel matrix on non-linear SVM classifiers requires the computation of the distance between an image and every training sample. This means that the intermediate representation of all training images must be kept in memory while classifying new images.
- Processing high resolution images without scaling them down can result in a very large number of descriptors being extracted for that image, as the number of descriptors grows linearly with the area of the image.
- Using a very small step when using dense descriptor extraction will result in an extremely large number of descriptors being extracted from an image.

Chapter 5

Detecting Nature

Taking into account the scene classification framework set-up, parameters, and limitations described in chapter 4, it is now possible to evaluate the feasibility and impact of each technique for the task of natural and man-made image classification.

The results of several experiments evaluating the impact of each technique are detailed in this chapter. The ‘Performance’ value that is reported corresponds to the mean per-class accuracy, which is detailed in section 4.2.

5.1 Experiments

Each of the following sections describes one set of experiments and its results regarding a specific type of techniques or parameters.

5.1.1 Descriptor Type

As the basis for the entire image encoding and classification process, the descriptor type plays one of the most important roles.

Since LBP descriptors cannot be computed on several scales, and for a fairer comparison, all descriptors were extracted at a single scale, as opposed to the default multi-scale settings described in section 4.4. To analyse the impact of using several scales when extracting features see section 5.1.5.

Table 5.1: Classification results for different types of descriptors

Descriptor type	Scale	Spacing	Performance (%)
SIFT	20	10	89.5 ± 0.8
HOG	20	-	77.6 ± 0.9
LBP	-	1	83.0 ± 0.9

The results in table 5.1 show the classification performance for several different descriptors. SIFT descriptors provide the best classification results, by a significant margin. SIFT descriptors are the most popular descriptor in the scene classification literature and these results show that it is not without merit. HOG and LBP, which are respectively faster to run and faster to implement, provide a smaller descriptors of 36 and 8 dimensions, which may be be their downfall when it comes to representing the same data that a 128 dimensional SIFT descriptor can.

5.1.2 Image Smoothing

The original work by Lowe (2004) uses an image smoothing or blurring proportional to the scale of the feature to be extracted. Vedaldi and Fulkerson (2008) suggests that for scene classification using lower or no smoothing improves the classification accuracy.

An higher value for sigma results in a more blurred image and zero means that no blur is applied at all. Table 5.2 shows the impact of several levels of image smoothing on the classification performance.

Table 5.2: Blurring images before extracting descriptors

Gaussian sigma	Cost per image (ms)	Performance (%)
0	550	90.2 $\pm$ 0.6
1	1200	90.5 $\pm$ 0.6
2	1200	89.1 $\pm$ 0.7

The results show that a small blurring benefits the classification accuracy, but increasing it further will degrade the results. This may be explained due to the fact that since the images have different original sizes, a low level of smoothing may be able remove small artefacts, bringing the images closer together. On the other hand, too much smoothing results in the loss of information that could have otherwise been used to distinguish between pictures.

5.1.3 Image Resizing

Resizing an image before extracting descriptors can be used for both performance and accuracy reasons.

Since the number of descriptors rises linearly with the area of an image, artificially limiting the size of an image creates an upper cap to the maximum amount of descriptors for each image. This reduces both the computational cost and the memory requirements to process the images.

Another reason for resizing images is that this creates an uniformity on the number of descriptors of images with different original sizes. This uniformity may allow for more consistent and accurate classification results.

Due to memory limitations combined with the very high resolution of some images in this dataset, the experiments in this section are performed using a single scale for descriptors. In this

case, SIFT descriptors with a patch size of 20 pixels and a spacing of 10 pixels are extracted, instead of the multi-scale default value described in section 4.4.

Three possible image scaling techniques were tested during these experiments:

- ‘Full resolution’ means that images are not scaled before feature extraction.
- ‘Maximum side of x px’ resizes images which have one side larger than x pixels so that its largest size is x pixels wide.
- ‘Resize all to x px’ will scale all images, even the ones with size smaller than x pixel, so that its largest side is exactly x pixels wide.

Table 5.3 shows the results of applying different scaling techniques and target sizes to the images before extracting features. The target sizes used were 250, 500, and 1000 pixels.

Table 5.3: Impact of scaling images before feature extraction

Image scaling	Cost per image (ms)	Performance (%)
Full resolution	4500	87.6 ± 1.3
Maximum side of 1000px	2500	88.7 ± 1.6
Resize all to 1000px	4000	86.4 ± 1.4
Maximum side of 500px	1000	90.1 ± 0.4
Resize all to 500px	950	89.8 ± 0.5
Maximum side of 250px	550	89.4 ± 0.9
Resize all to 250px	600	89.6 ± 0.6

Resizing an image so that it becomes larger, which can happen when using the ‘resize all’ approach, shows a negative impact in the classification accuracy. This is more evident the more you scale the image, at higher target resolution. On the other hand, having images of different sizes is not necessarily a problem, as the impact in the classification accuracy of having more variation in image size, as is the case with the ‘maximum side’ technique at higher target resolutions, is not very large. This shows that the implemented techniques are generally robust against having images with different sizes and definitions, but only up to a certain point. When using full resolution, where images can range from 126×106 to 3752×5000 , which represents an increase in the number of descriptors by a factor of over 1400 times, a decrease in performance becomes very apparent.

Note that these experiments were made with a fixed patch size and spacing for the descriptors. Using different descriptor parameters may have an impact on the ideal image size and as such these should not be seen as universal and ideal image sizes.

Considering the vast difference in the cost of classifying a full-scale high resolution image versus using a scaled down version, combined with the fact that sizing down images, so they are

more uniform in size, can actually improve the classification performance, it becomes apparent that resizing an image is a very important step for this type of image classification.

5.1.4 Colour Information

In order to ensure that no data related to the colour was excluded by the Principal Component Analysis, the full size of the descriptors was used during this experiment. Since the usage of the full-sized descriptors results in a significantly higher memory usage, the descriptors were extracted at a single scale, with a spacing of 10 pixels and a scale of 20 pixels, in order to reduce the total number of descriptors. Table 5.4 compiles the results of using colour information for scene classification.

Table 5.4: Using colour information during descriptor extraction

Colour format	Descriptor size	Cost per image (ms)	Performance (%)
Grey-scale	128	600	89.1 ± 0.6
Opponent	384	1000	85.5 ± 0.4
HSV	384	1200	84.7 ± 1.8

While most works in scene classification do not use colour, Bosch, Zisserman and Muñoz (2008) showed an improvement on the classification results on the 15-scene dataset by using colour SIFT. For the natural versus man-made classification, using colour information decreased the accuracy instead.

5.1.5 Descriptor Scale

The scale at which a descriptor is extracted can be changed, which corresponds to the pixel diameter of a circle, centred in the point where the descriptor is extracted, which delimits the data that is used for the calculation of the descriptor. The descriptor spacing, on the other hand, is the distance, in pixels, between each point of a dense grid, where descriptors will be extracted. Both of these values can be tweaked to achieve better classification results.

The idea of using multiple scales when extracting descriptors can be found in several articles in the scene classification literature. Bosch, Zisserman and Muñoz (2008) suggest that using concentric SIFT descriptors at four different scales, while Krapac, Verbeek and Jurie (2011) use descriptors extracted at 8 different scales, separated by a factor of 1.2.

Table 5.5 compares several different scales and spacing combinations in order to evaluate the impact of these two settings. Using more than one scale at once is also studied.

A lower spacing between descriptors results in higher classification performance at the cost of a much higher memory requirement. On a small 250×250 pixel picture, using a scale and spacing of 20 and 10, respectively, results in 289 descriptors, while using a scale and spacing of

Table 5.5: Impact of scale and spacing of SIFT descriptors

Scale	Spacing	Cost per image (ms)	Performance (%)
10	5	1000	89.2 ± 0.8
10	10	650	87.1 ± 0.4
10	20	600	81.5 ± 1.3
10	50	500	65.8 ± 1.1
20	5	950	90.5 ± 0.5
20	10	650	89.4 ± 0.3
20	20	550	86.0 ± 1.4
20	50	500	70.3 ± 1.4
50	5	900	88.7 ± 1.1
50	10	650	89.0 ± 0.5
50	20	550	87.2 ± 1.4
50	50	500	77.3 ± 2.0
[20, 50]	[5, 10]	1200	90.1 ± 0.6

20 and 5, results in 1156 descriptors instead, an increase by a factor of four. This higher number of descriptors is also reflected in the amount of time it takes to process the data.

It is also possible to note that using a spacing higher than the scale, which results in gaps in the image that are not covered by the descriptors, has a very significant negative impact in the classification results.

Instead of boosting the classification results, using multiple scales reveals a performance that is the average of the used scales and spacings. This results in a higher computational cost with a negative impact on the classification performance. A better alternative to using multiple descriptors is to test and determine the ideal combination of scale and spacing for the dataset at hand.

5.1.6 Distance Transforms

This experiment was introduced in order to test if the advice related to the transformation of descriptors, given by Arandjelovic and Zisserman (2012) for object retrieval and detailed in section 2.4.1, also applies to the scene classification.

Table 5.6 highlights the differences between using the original version of SIFT and using SIFT after transforming it using the Hellinger mapping. This transformation is applied to the descriptors before using them in the remaining steps of the classification framework.

Transforming the SIFT descriptors before using them to classify images provides a minimal improvement in the classification results but comes with no negative impact in the computational cost. As a very simple to implement technique and with negligible impact in the classification cost, this is a good improvement to use on scene classification problems.

Table 5.6: Transforming descriptors using Hellinger kernel

Descriptor type	Cost per image (ms)	Performance (%)
Regular SIFT	350	89.7 ± 1.0
Hellinger-transformed SIFT	350	90.2 ± 0.7

5.1.7 Dimensionality Reduction

Applying an algorithm to reduce the dimensionality of the descriptors, such as Principal Component Analysis, can impact the results in several different ways. The first is that by reducing the total size of the data that must be processed, the computational cost to classify an image is also reduced. A reduction in the memory required to store the descriptors is another impact of PCA.

Table 5.7 shows what happens when PCA is used to reduce the dimensionality of a SIFT descriptor. SIFT descriptors, using their default parameters, have 128 dimensions. As such, the entry in the table with 128 PCA dimensions acts as a baseline value for this experiment.

The classification cost per image, presented in table 5.7, does not take into account the time required to extract the descriptors themselves as this value is the same for all experiments, and instead shows the time required to perform PCA on the descriptors, build their intermediate representation and classify the image.

Table 5.7: Reducing dimensionality of a SIFT descriptor with size 128

PCA dimension	Cost per image (ms)	Performance (%)
64	300	90.4 ± 0.8
80	350	90.3 ± 0.3
96	380	90.5 ± 1.4
112	410	90.6 ± 0.5
128	450	91.1 ± 0.8

While there is a trend in the experiments showing that the more the descriptor size is reduced using PCA, the worse the results, the difference is not very significant. Combined with the reduction in time required to classify an image, this makes PCA a viable way of reducing the classification cost without much loss in accuracy.

5.1.8 Encoding Method

The codebook plays an important role in the scene classification framework as it is the basis used for encoding all images before they can be classified. One parameter that can have an impact in the quality of the codebook is the number of codewords that it contains.

Since the way that the Fisher framework uses the codebook is very different from the Bag-of-Words technique, two sets of experiments were performed, as reported in table 5.8 and table 5.9.

Table 5.8: The impact of codebook size in the Fisher framework

Codebook size	Cost per image (ms)	Performance (%)
10	150	89.5 ± 0.8
20	200	90.3 ± 0.3
50	350	90.7 ± 0.6
100	550	91.0 ± 0.5
200	1000	91.1 ± 0.3

In the Fisher framework the classification results show a steady improvement in the classification performance with a larger codebook. The major drawback is that the cost of the classification increases steeply with a larger number of codewords, making it impractical to keep increasing the size of the codebook until the performance saturates.

Table 5.9: The impact of codebook size when using Bag-of-Words

Codebook size	Cost per image (ms)	Performance (%)
50	10	80.2 ± 2.6
100	20	80.7 ± 0.9
200	40	81.6 ± 2.1
500	100	81.7 ± 1.6
1000	200	79.7 ± 2.0

With Bag-of-Words the performance does not steadily increase like in the Fisher framework. Instead, the performance peaks at between 200 and 500 keywords, which is consistent with the results reported by (Lazebnik, Schmid and Ponce 2006).

Notice that increasing the size of the codebook implies an increase in the size of the intermediate representation. This size increase, in turn, means that every piece of the classification process is going to take longer to compute.

5.1.9 Spatial Information

Using spatial information together with the Fisher representation was not explored in this work due to its implementation complexity and the fact that Garg, Chandra and Jawahar (2012) have shown that the impact of using spatial information with Fisher vectors, while positive, is much lower than the improvements achieved when spatial information is combined with Bag-of-Words representations (Lazebnik, Schmid and Ponce 2006).

This set of experiments compares the use of no spatial information versus the use of one of two possible techniques for spatial representation: using slices or squares.

For each pyramid level of the ‘slices’ technique, described in section 2.6.1, each slice of the previous level is divided in half. This technique creates two sets of representations, one containing only horizontal slices and another containing only vertical slices. In practice this means that a three level pyramid with slice partitioning will contain a total of thirteen slices with the following distribution: level 1 contains one representation for the whole image, level 2 has two horizontal slices and two vertical slices, and level 3 uses four horizontal and four vertical slices.

The ‘squares’ technique recursively divides an image into squares for each pyramid level. This means that each square of a pyramid level will result in four new squares in the next level. As such, a three level pyramid has twenty-one divisions with one in the first level, four in the second and sixteen in the third.

Table 5.10: Using spatial information with Bag-of-Words

Partition type	Pyramid levels	Cost per image (ms)	Performance (%)
None	1	40	81.6 ± 2.1
Slices	2	45	79.1 ± 1.9
Slices	3	50	79.6 ± 3.0
Squares	2	45	78.2 ± 3.1
Squares	3	45	77.4 ± 2.5

Table 5.10 shows that, unlike the very promising results presented by Lazebnik, Schmid and Ponce (2006), spatial information has a much lower impact when classifying natural and man-made images. One possible explanation for this behaviour is the very high intra-class variation, combined with significant differences between classes in this dataset. This means that, unlike when trying to distinguish between similar classes such as the ‘highway’ versus ‘inside city’ that appear in the 15-scene dataset, it is more important to have a robust technique that can evaluate the overall differences between the images, instead of one that tries to determine similarities on a spatial basis.

5.1.10 Type of Classifier

While Bag-of-Words representations use histograms to represent the image data, which benefit from a non-linear classifier, Fisher Vectors provide a very high dimensional representation, which is usually able to achieve good results with a simple linear classifier (Perronnin and Dance 2007).

Table 5.11 shows the differences between a linear and a non-linear classifier when applied to both the Fisher Vector and the Bag-of-Words representations.

As expected, using an appropriate kernel for the histogram representation of the Bag-of-Words technique results in a very significant improvement, while the Fisher vector representation, which

Table 5.11: Differences between linear and non-linear SVM classifiers

Encoding technique	Classifier type	Cost per image (ms)	Performance (%)
Fisher Vector	Linear	350	90.6 ± 0.8
Fisher Vector	Non-linear	360	91.3 ± 0.3
Bag-of-Words	Linear	35	81.2 ± 1.5
Bag-of-Words	Non-linear	45	87.9 ± 1.1

has much higher dimensionality, sees a much more modest performance increase. While using a non-linear classifier does bring the performance of both encoding techniques much closer, the Fisher framework still outperforms the Bag-of-Words representation.

5.1.11 Tuning Classifier

Support Vector Machine classifiers have one parameter that corresponds to the penalty applied to misclassified data points during its training step, which is usually typeset as ‘C’. The penalty is defined in $\mathbb{R}^+$, where a low value will result in a classifier that tries to reduce the number of support vectors, thus generalizing the classifier, while a high penalty will encourage to classifier to overfit the data with a higher number of support vectors. The ideal value for this parameter must be determined experimentally, using cross-validation, taking into account that using a logarithmic scale to space the values is the most common strategy used for this process.

As fig. 5.1 shows, this parameter can have a significant impact on the classification result, so it is important to determine its ideal value, as it does not impact the classification cost once optimized.

5.1.12 Number of Training Images

The example images provided to the classifier are essential to obtain good classification results. Figure 5.2 shows the impact of the number of images used during training over the classification accuracy.

The graph shows a distinct logarithmic growth, showing that increasing the number of training images suffers from diminishing returns.

Increasing the number of training images per class from 50 to 500 results in an increase in accuracy of just 3% on average, over the six experiments.

Another advantage of using a larger number of training images is that the results become much more consistent, no matter what subset of images is chosen to train the classifier.

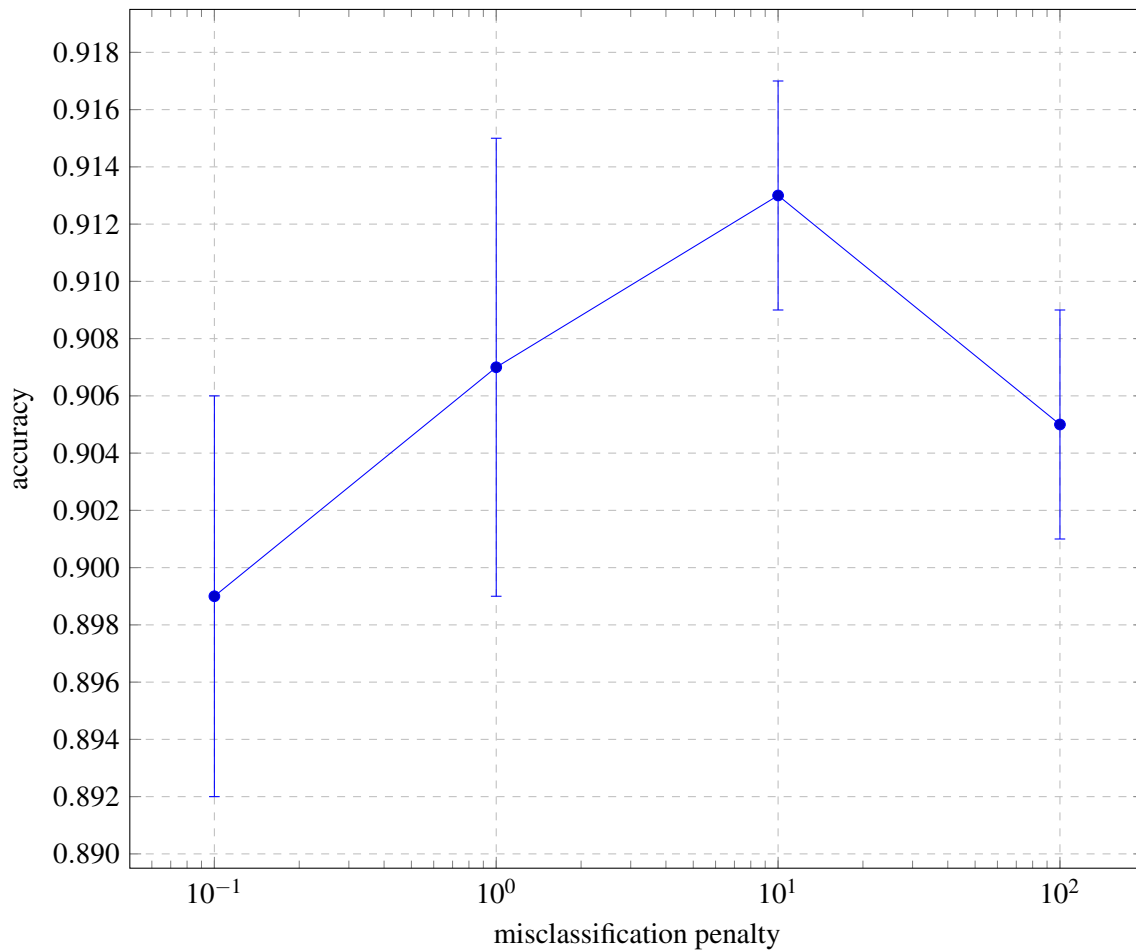


Figure 5.1: The SVM misclassification penalty parameter

5.2 Optimization

The previous experiments revealed that very often the time required to process an image and the classification results do not grow at the same rate. While most scene classification works opt to present the best results possible, it is desirable that the time required to classify an image and the achieved accuracy are balanced when applying these techniques to a real world scenario. The following two sections summarize the results that can be achieved by properly tweaking all parameters, based on the results of the previous section.

5.2.1 Optimizing for Accuracy

The results in this section show the best possible classification accuracy that can be achieved with the implemented techniques. In order to achieve these results, the value used for each parameter corresponds to the one that achieved the best results in its respective experiment in section 5.1. Table 5.12 compiles the settings used in this section.

With these settings, the average accuracy of all experiments is $94.0\% \pm 0.2$. This value can be

Detecting Nature

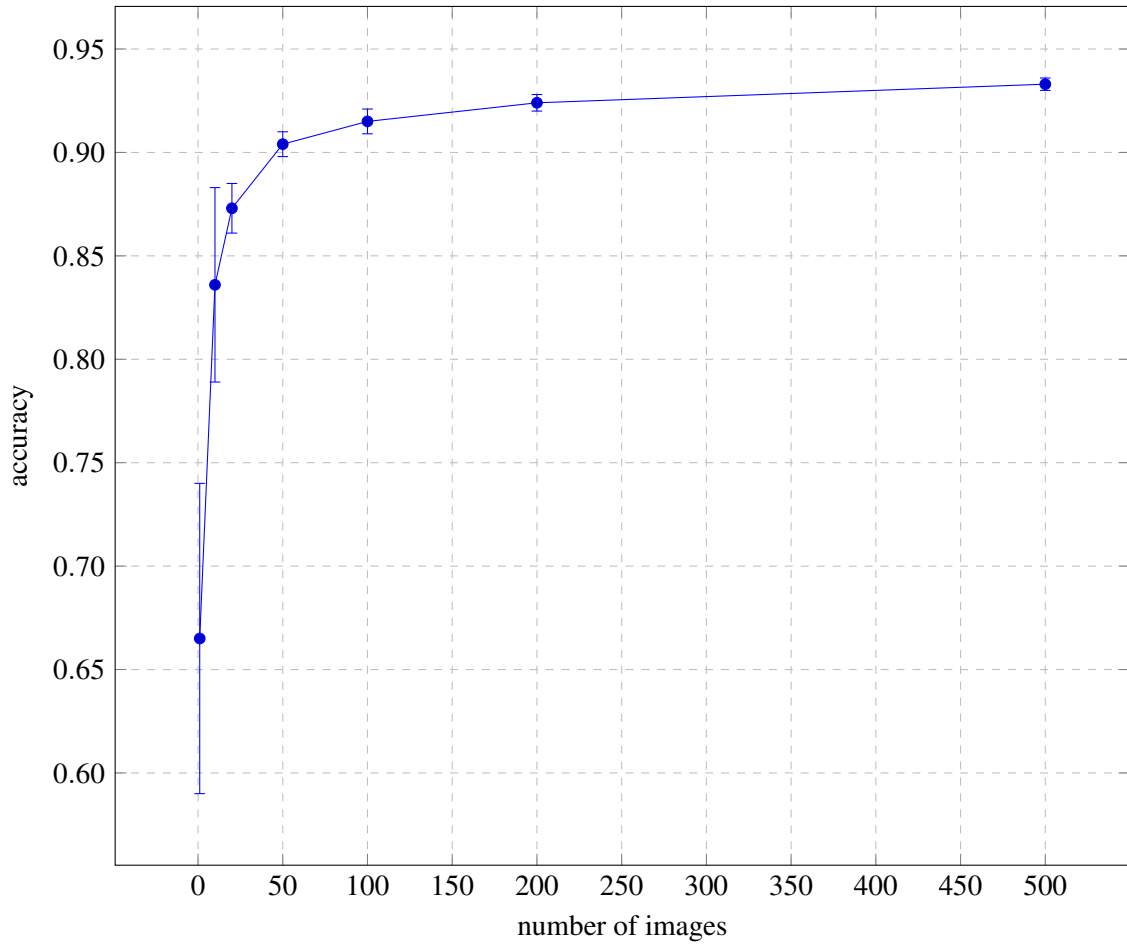


Figure 5.2: Impact of number of training images in classification accuracy

broken down into a confusion matrix, which is presented in table 5.13.

While these are excellent results, they come at a cost, since by using these settings, it takes an average of 6.5 seconds to classify each image.

5.2.2 Optimizing for Speed

This section shows the speed that can be achieved by choosing algorithms and their parameters with the objective of achieving a good balance between accuracy and speed. The settings used are presented in table 5.14.

The results are presented in table 5.15 which corresponds to an accuracy of $90.7\% \pm 0.4$.

This corresponds to a 3.5% drop in accuracy when compared to the optimal settings. On the other hand, considering that it now takes only 500 milliseconds to classify an image, the classification cost dropped by a staggering 92.3%.

Table 5.12: Settings used to achieve the best accuracy

Setting	Value
Image colour	Grey-scale
Image scaling	Maximum side of 500px
Image smoothing	Sigma of 1.0
Descriptor type	SIFT
Patch size	[20]
Grid step	[5]
Transformations applied	Hellinger and PCA
Final PCA dimension	128
Codebook type	Fisher vector
Number of codewords	200
Sample images in codebook	500
Maximum descriptors in codebook	500000
Spatial representation	None
Classifier type	Non-Linear SVM
SVM misclassification penalty	10.0
Training images per class	500

Table 5.13: Confusion matrix for the best results

	Classifier guessed Natural	Classifier guessed Man-made
Real class is Natural	95.3	4.7
Real class is Man-made	7.4	92.6

5.3 Additional Information

This section presents some experiments that are not directly related to the classification accuracy that can be achieved by the framework, but show the behaviour of the framework when exposed to different situations.

5.3.1 Cross-dataset Generalization

In order to ensure that there is no bias in the dataset used in these experiments, several experiments were conducted by training a classifier in one dataset and then using that same classifier on a different dataset.

If there is no sharp decrease in the classification rate when doing this cross-dataset validation we can assume that there is no particular bias to any one dataset and that it is indeed capturing the differences between the natural and man-made images.

In order to test this hypotheses, a classifier was trained using the SUN database subset de-

Table 5.14: Settings used to achieve better speed

Setting	Value
Image colour	Grey-scale
Image scaling	Maximum side of 250px
Image smoothing	Disabled
Descriptor type	SIFT
Patch size	[50]
Grid step	[20]
Transformations applied	Hellinger and PCA
Final PCA dimension	64
Codebook type	Fisher vector
Number of codewords	10
Sample images in codebook	500
Maximum descriptors in codebook	500000
Spatial representation	None
Classifier type	Linear SVM
SVM misclassification penalty	10.0
Training images per class	500

Table 5.15: Confusion matrix for the balanced results

	Classifier guessed Natural	Classifier guessed Man-made
Real class is Natural	91.8	8.2
Real class is Man-made	10.3	89.7

scribed in section 4.6. This classifier was then applied to a set of 700 natural images used in the work of Vogel and Schiele (2007). Of these, 99.6% of the images were correctly classified as natural.

Another experiment consisting of using the 15-scene dataset split into two classes containing the man-made categories (bedroom, highway, industrial, inside city, kitchen, living room, office, store, street, suburb, tall building) and the natural categories (coast, forest, mountain, open country) was also performed. The overall accuracy was 96.1%, broken down in the confusion matrix in table 5.16.

These results show that not only was the classifier capable of distinguishing between man-made and natural images on these datasets without having been trained with this data, the results were also superior to those in the original dataset. This can be explained by the lower diversity in these datasets, as well as a larger percentage of purely natural and purely man-made scenes they contain.

Table 5.16: Confusion matrix for the 15-scene dataset merged into two classes

	Classifier guessed Natural	Classifier guessed Man-made
Real class is Natural	98.3	1.7
Real class is Man-made	6.1	93.9

5.3.2 Confidence Rating

Associated with the class identified for each image, there is a confidence value regarding that decision. This value represents the probability of an image belonging to a certain class and is calculated by the classifier itself, based on how close the image is to the class boundary.

To evaluate whether or not this confidence value provides any kind of useful information, an experiment was executed over the original SUN Database, excluding all images from categories that may belong to both the natural and man-made categories. Table 5.17 shows the results.

Table 5.17: Correlation between confidence and correct classifications

	Total images	Percentage of total
All images	98758	100.0
Images with confidence above 80%	81242	82.3
Correct images with confidence above 80%	78209	96.3
Images with confidence below 80%	17516	17.7
Correct images with confidence below 80%	11974	68.4

The results show that there is a very high correlation between a high confidence rating and the percentage of correctly classified images. Combined with the relatively low percentage of images with low confidence, this means it's easy to set up a manual review process for images that cause trouble to the classifier, possibly adding some of those images to the training dataset, thus improving the overall accuracy of the classifier.

Chapter 6

Result Analysis

Analysing the results presented in the previous chapter can provide some insights towards the possible application of the scene classification framework to the problems initially described in this document.

6.1 Comparing algorithms

Comparing the impact of the different algorithms and techniques proposed in the scene classification literature reveals some interesting patterns that are summarized in the following sections.

6.1.1 Importance of Experimenting

Scene classification techniques are not universal, as the experiments regarding the use of spatial information have shown. While in the original 15-scene dataset Lazebnik, Schmid and Ponce (2006) reported that using this spatial information resulted in an classification accuracy increase of 12%, using this same technique for natural versus man-made image classification decreased the accuracy instead. This shows that while techniques may seem promising at first, careful experimentation and comparison must be performed to determine what does improve the classification results on a certain dataset.

6.1.2 Adapting Techniques

Some techniques were developed with object recognition in mind and, while they can also be applied to other areas such as scene classification, they may not be ideal for these tasks. One such example are HOG descriptors, which show very good results when applied to the task of human detection, but have underwhelming results when compared to other descriptors for scene classification tasks.

Other techniques show their versatility in that they were originally developed to perform a task and show excellent performance when applied to other tasks such as natural image classification. SIFT is the prime example of this as it was originally developed in order to match different

viewpoints of objects and scenes, but performs extremely well in all of the executed experiments.

It's important to note that some of these techniques, like SIFT descriptors, require some adaptations to perform better in scene classification problems. In this case, SIFT descriptors are extracted using dense descriptor extraction instead of the original region of interest detection technique.

6.2 Building the Framework

Having a flexible framework that could easily and autonomously run and repeat an experiment was crucial to the success of this work. Combined with its extensibility it becomes much easier to apply this framework not only to the problem described in this work but to many other scene and object classification problems.

Caching and parallelization were also very important components of this framework as they allowed for a much larger number of experiments to be executed. Caching also doubles as an extra feature for a production environment, as it allows new images to be added to the training set and recreating the classifier much faster than it would be otherwise possible.

6.3 Computational Cost

With a few exceptions, such as Perronnin and Dance (2007), very few works in the scene classification literature mention the computational complexity of the proposed algorithms. This is an important factor to consider when moving from a research environment to a real world application. As such, a one percent increase in the correct classification rate may not be worth it when it comes with a classification cost that is an order of magnitude higher.

The results presented in section 5.2 show the differences between settings that were optimized for accuracy, which are the values usually reported in the scene classification literature, and those that can be obtained using a more conservative and balanced set of parameters. This shows it is possible to achieve a much faster classification speed with only a few compromises regarding the ability of the framework to classify images accurately.

6.4 Mixing Datasets

Using a representative set of natural and man-made images and then using a classifier trained on that set to classify images from other sources, such as Flickr is a possibility. This is due to the results in section 5.3.1 which show that the classifier can indeed capture the real differences between a natural and a man-made image and is not simply capturing a bias in the dataset.

6.5 Ignoring Irrelevant Information

The presented techniques have a very important property, in that they can reduce the weight given to certain types of information that appear on several classes. For the natural scene classification

problem, this means that if the training dataset contains enough images with humans on both the natural and the man-made categories, they end up having no effect in the classification results. As an example, fig. 6.1 shows images that were classified correctly and with high confidence, despite the focus of the image being the persons and not the scene in the background.



Figure 6.1: Effect of humans on the foreground of an image

This confirms that the idea of using image sharing websites, such as Flickr or Panoramio, as a data source for the creation of a large database of geo-tagged natural and man-made images is feasible. If the classification framework was not robust enough to overcome these external influences this would not be a viable alternative, since many holiday pictures with people as the main subject and other images that are not entirely focused on the landscape can be found in these websites.

6.6 Detailing Methodology

Placing all settings into an independent file simplified the process of reporting the parameters used during all experiments, ensuring that no setting was forgotten. Combined with the open-source nature of the developed framework, this allows each and every result presented in this work to be easily reproduced without any tinkering or guessing involved.

6.7 Human-in-the-Loop

When training a classifier, a small subset of representative images is enough to achieve good classification results. This means that it is feasible to have a human-in-the-loop solution, allowing for faster feedback and correction of possibly misclassified images.

6.8 Naturalness Ranking

A simple procedure can be applied in order to visualise what types of images this classification framework has no trouble classifying as natural or man-made and those that it finds ambiguous. To do this, a large number of assorted images is classified and then sorted by their class and classification confidence rating. Section 5.3.2 provides additional details about this confidence rating.

Figure 6.2 shows images sorted from the most natural to the most man-made, according to the classifier's confidence. Images marked with an asterisk represent images that were misclassified, according to the label assigned by the original authors of the SUN database and the definition of a natural image presented in section 4.3.

6.9 Large Scale Viability

In order to create a large-scale map of the natural and man-made areas of the Earth's surface, it is necessary to process a large enough number of images to cover its surface.

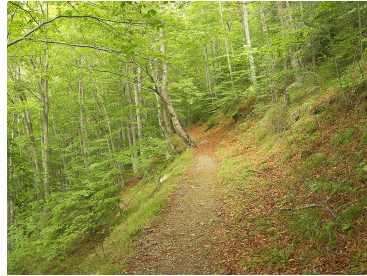
The average size of the images in the SUN Database is around 200kB, which means that in 1TB of data there are 5 million images. When compared to the total surface area of Earth, which is approximately 510 million square kilometres, that 1TB of data corresponds to just 0.01 images per square kilometre. This poses a big challenge when trying to build a large-scale map of the natural and man-made areas in Earth using internet-sourced images.

On the other hand, the experiments that were performed have shown that there is a great degree of flexibility regarding the classification cost of each image. By using just two regular quad-core computers and balancing the speed versus accuracy parameters as described in section 5.2.2, it is possible to process all that data in just under 4 days. This means it is viable to use this classification framework to solve the problem of classifying a large enough number of images to cover a map of Earth.

Result Analysis



(a) 99% natural



(b) 96% natural



(c) 80% natural*



(d) 77% natural



(e) 70% natural



(f) 65% natural



(g) 64% natural*



(h) 50% natural



(i) 53% manmade



(j) 68% manmade



(k) 83% manmade



(l) 99% manmade

Figure 6.2: Naturalness ranking of several example images

Result Analysis

Chapter 7

Conclusions

UNEP-WCMC proposed a very well defined problem: detect whether a picture represents a natural or a man-made scene. This problem can be generalized as a scene classification task, which is an active area of research in the computer vision and image processing fields.

The scene classification area of research is very active and has shown vast progress over the last decade, with new techniques being proposed and new applications being discovered in related image processing and indexing areas of study.

Analysing the scene classification literature can reveal an underlying pattern for scene classification: the extraction of relevant features and its representation, followed by its transformation into an intermediate representation and image description, having an image classifier as the last step.

In order to solve the proposed problem, a framework capable of supporting the evaluation of different image classification techniques was developed. Features other than those strictly necessary for the correct classification of images have also been added, so as to improve the utility and longevity of the tool.

While some techniques greatly improve the classification results on both the original context of their work and in the context of determining the degree of naturalness of an image, such as the Fisher framework, others that show excellent results on the original work, have a negative impact in this type of natural versus man-made classification, such as is the case with Spatial Pyramid Matching.

The developed framework can classify the images in a very diverse dataset containing two classes, corresponding to natural and man-made scenes, with up to 94% accuracy. Experiments have also shown that the computational cost of classifying an image can be tweaked at the expense of accuracy. A good balance can reduce the computational cost by 92.3% while only reducing the classification accuracy by 3.5%.

The results are very promising and show that using the developed framework to act as both a tool to validate user-submitted pictures or to create a map of both the man-made and natural areas of our planet is feasible.

7.1 Future Work

With the development of frameworks such as OpenCL and CUDA, it is now possible to develop applications that run directly on the GPU, which can contain hundreds of cores and are optimized for highly parallel work. Since image classification is a highly parallel problem, as remarked in section 3.2.4, it would be advantageous to harness the power of these devices. Allowing the developed classification framework to take advantage of the processing power of the computer's GPU would greatly reduce the total time required to classify a set of images.

Another possible improvement stems from the fact that the focus of this work was about the feasibility of using pictures taken by a regular camera and classifying them into either natural or non-natural. Using other sources of images, such as satellite imagery, to complement the classification results, could allow for a more accurate representation of the man-made versus natural status of a given geographic area.

References

- Arandjelovic, R., and A. Zisserman. 2012. ‘Three things everyone should know to improve object retrieval’. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2911–2918. doi:[10.1109/CVPR.2012.6248018](https://doi.org/10.1109/CVPR.2012.6248018).
- Beis, Jeffrey S., and David G. Lowe. 1997. ‘Shape indexing using approximate nearest-neighbour search in high-dimensional spaces’. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR ’97)*, 10001006. doi:[10.1109/CVPR.1997.609451](https://doi.org/10.1109/CVPR.1997.609451).
- Boiman, O., E. Shechtman and M. Irani. 2008. ‘In defense of Nearest-Neighbor based image classification’. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, 1–8. doi:[10.1109/CVPR.2008.4587598](https://doi.org/10.1109/CVPR.2008.4587598).
- Bosch, A., A. Zisserman and X. Muñoz. 2008. ‘Scene Classification Using a Hybrid Generative / Discriminative Approach’. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, number 4 (April): 712–727. ISSN: 0162-8828. doi:[10.1109/TPAMI.2007.70716](https://doi.org/10.1109/TPAMI.2007.70716).
- Bosch, Anna, Xavier Muñoz and Robert Martí. 2007. ‘Which is the best way to organize/classify images by content?’ *Image and Vision Computing* 25, number 6 (June): 778–791. ISSN: 0262-8856. doi:[10.1016/j.imavis.2006.07.015](https://doi.org/10.1016/j.imavis.2006.07.015).
- Boutell, Matthew, Christopher Brown and Jiebo Luo. 2002. ‘Review of the State of the Art in Semantic Scene Classification’. *University of Rochester, Rochester, NY*.
- Chang, Chih-Chung, and Chih-Jen Lin. 2011. ‘LIBSVM: A library for support vector machines’. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, *ACM Transactions on Intelligent Systems and Technology* 2 (3): 27:1–27:27. doi:[10.1145/1961189.1961199](https://doi.org/10.1145/1961189.1961199).
- Chatfield, Ken, Victor Lempitsky, Andrea Vedaldi and Andrew Zisserman. 2011. ‘The devil is in the details: an evaluation of recent feature encoding methods’. In, 76.1–76.12. British Machine Vision Association. ISBN: 1-901725-43-X. doi:[10.5244/C.25.76](https://doi.org/10.5244/C.25.76).
- Dalal, N., and B. Triggs. 2005. ‘Histograms of oriented gradients for human detection’. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 1, 886–893 vol. 1. June. doi:[10.1109/CVPR.2005.177](https://doi.org/10.1109/CVPR.2005.177).
- Everingham, Mark, Luc Van Gool, Christopher K. I. Williams, John Winn and Andrew Zisserman. 2010. ‘The Pascal Visual Object Classes (VOC) Challenge’ [in en]. *International Journal of Computer Vision* 88, number 2 (June): 303–338. ISSN: 0920-5691, 1573-1405. doi:[10.1007/s11263-009-0275-4](https://doi.org/10.1007/s11263-009-0275-4).

REFERENCES

- Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang and Chih-Jen Lin. 2008. ‘LIB-LINEAR: A Library for Large Linear Classification’. *Journal of Machine Learning Research* 9:1871–1874. doi:[10.1145/1390681.1442794](https://doi.org/10.1145/1390681.1442794).
- Fei-Fei, L., and P. Perona. 2005. ‘A Bayesian hierarchical model for learning natural scene categories’. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 2, 524–531 vol. 2. June. doi:[10.1109/CVPR.2005.16](https://doi.org/10.1109/CVPR.2005.16).
- Gao, Shenghua, I.W. Tsang, Liang-Tien Chia and Peilin Zhao. 2010. ‘Local features are not lonely – Laplacian sparse coding for image classification’. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3555–3561. June. doi:[10.1109/CVPR.2010.5539943](https://doi.org/10.1109/CVPR.2010.5539943).
- Gao, Shenghua, I.W.-H. Tsang and Liang-Tien Chia. 2013. ‘Laplacian Sparse Coding, Hypergraph Laplacian Sparse Coding, and Applications’. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, number 1 (January): 92–104. ISSN: 0162-8828. doi:[10.1109/TPAMI.2012.63](https://doi.org/10.1109/TPAMI.2012.63).
- Garg, Vinay, Siddhartha Chandra and C. V. Jawahar. 2012. ‘Sparse discriminative Fisher vectors in visual classification’. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing, 55:155:8. ICVGIP '12*. ACM. ISBN: 978-1-4503-1660-6. doi:[10.1145/2425333.2425388](https://doi.org/10.1145/2425333.2425388).
- Gemert, Jan C. van, Jan-Mark Geusebroek, Cor J. Veenman and Arnold W. M. Smeulders. 2008. ‘Kernel Codebooks for Scene Categorization’. In *Computer Vision - ECCV 2008*, edited by David Forsyth, Philip Torr and Andrew Zisserman, 696–709. Lecture Notes in Computer Science 5304. Springer Berlin Heidelberg, January. doi:[10.1007/978-3-540-88690-7_52](https://doi.org/10.1007/978-3-540-88690-7_52).
- Krapac, J., J. Verbeek and F. Jurie. 2011. ‘Modeling spatial layout with Fisher vectors for image categorization’. In *2011 IEEE International Conference on Computer Vision (ICCV)*, 1487–1494. doi:[10.1109/ICCV.2011.6126406](https://doi.org/10.1109/ICCV.2011.6126406).
- Lazebnik, S., C. Schmid and J. Ponce. 2006. ‘Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories’. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:2169–2178. doi:[10.1109/CVPR.2006.68](https://doi.org/10.1109/CVPR.2006.68).
- Lowe, David G. 2004. ‘Distinctive Image Features from Scale-Invariant Keypoints’. *International Journal of Computer Vision* 60, number 2 (November): 91–110. ISSN: 09205691. doi:[10.1023/B:VISI.0000029664.99615.94](https://doi.org/10.1023/B:VISI.0000029664.99615.94).
- Muja, Marius, and David G. Lowe. 2009. ‘Fast approximate nearest neighbors with automatic algorithm configuration’. In *VISAPP International Conference on Computer Vision Theory and Applications*, 331340.
- Ojala, T., M. Pietikainen and T. Maenpaa. 2002. ‘Multiresolution gray-scale and rotation invariant texture classification with local binary patterns’. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (7): 971–987. ISSN: 0162-8828. doi:[10.1109/TPAMI.2002.1017623](https://doi.org/10.1109/TPAMI.2002.1017623).
- Oliva, Aude, and Antonio Torralba. 2001. ‘Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope’. *International Journal of Computer Vision* 42, number 3 (May): 145–175. ISSN: 0920-5691, 1573-1405. doi:[10.1023/A:1011139631724](https://doi.org/10.1023/A:1011139631724).

REFERENCES

- Perronnin, F., and C. Dance. 2007. ‘Fisher Kernels on Visual Vocabularies for Image Categorization’. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07*, 1–8. doi:[10.1109/CVPR.2007.383266](https://doi.org/10.1109/CVPR.2007.383266).
- Perronnin, Florent, Jorge Sánchez and Thomas Mensink. 2010. ‘Improving the Fisher Kernel for Large-Scale Image Classification’. In *Computer Vision - ECCV 2010*, edited by Kostas Daniilidis, Petros Maragos and Nikos Paragios, 6314:143–156. Lecture Notes in Computer Science. Springer Berlin Heidelberg. ISBN: 978-3-642-15560-4. doi:[10.1007/978-3-642-15561-1_11](https://doi.org/10.1007/978-3-642-15561-1_11).
- Silpa-Anan, C., and R. Hartley. 2008. ‘Optimised KD-trees for fast image descriptor matching’. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, 1–8. doi:[10.1109/CVPR.2008.4587638](https://doi.org/10.1109/CVPR.2008.4587638).
- Smith, Lindsay I. 2002. ‘A tutorial on principal components analysis’. *University of Otago, New Zealand* 51:52. Visited on 10th June 2013. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf.
- Torralba, Antonio. 2003. ‘Contextual Priming for Object Detection’. *International Journal of Computer Vision* 53, number 2 (July): 169–191. ISSN: 0920-5691, 1573-1405. doi:[10.1023/A:1023052124951](https://doi.org/10.1023/A:1023052124951).
- Tuytelaars, T., M. Fritz, K. Saenko and T. Darrell. 2011. ‘The NBN kernel’. In *2011 IEEE International Conference on Computer Vision (ICCV)*, 1824–1831. doi:[10.1109/ICCV.2011.6126449](https://doi.org/10.1109/ICCV.2011.6126449).
- UNEP-WCMC. 2013. ‘UNEP-WCMC: What We Do’. Visited on 5th February 2013. http://www.unep-wcmc.org/about-us_17.html.
- Van de Sande, K. E A, T. Gevers and C. G M Snoek. 2010. ‘Evaluating Color Descriptors for Object and Scene Recognition’. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (9): 1582–1596. ISSN: 0162-8828. doi:[10.1109/TPAMI.2009.154](https://doi.org/10.1109/TPAMI.2009.154).
- Vedaldi, A., and B. Fulkerson. 2008. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. <http://www.vlfeat.org/>.
- Vedaldi, A., and A. Zisserman. 2010. ‘Efficient additive kernels via explicit feature maps’. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3539–3546. doi:[10.1109/CVPR.2010.5539949](https://doi.org/10.1109/CVPR.2010.5539949).
- Vogel, Julia, and Bernt Schiele. 2007. ‘Semantic Modeling of Natural Scenes for Content-Based Image Retrieval’. *International Journal of Computer Vision* 72, number 2 (April): 133–157. ISSN: 0920-5691, 1573-1405. doi:[10.1007/s11263-006-8614-1](https://doi.org/10.1007/s11263-006-8614-1).
- Wu, Jianxin, and J.M. Rehg. 2011. ‘CENTRIST: A Visual Descriptor for Scene Categorization’. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, number 8 (August): 1489–1501. ISSN: 0162-8828. doi:[10.1109/TPAMI.2010.224](https://doi.org/10.1109/TPAMI.2010.224).
- Xiao, Jianxiong, J. Hays, K.A. Ehinger, A. Oliva and A. Torralba. 2010. ‘SUN database: Large-scale scene recognition from abbey to zoo’. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3485–3492. June. doi:[10.1109/CVPR.2010.5539970](https://doi.org/10.1109/CVPR.2010.5539970).
- Yang, Jianchao, Kai Yu, Yihong Gong and T. Huang. 2009. ‘Linear spatial pyramid matching using sparse coding for image classification’. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, 1794–1801. doi:[10.1109/CVPR.2009.5206757](https://doi.org/10.1109/CVPR.2009.5206757).

REFERENCES

- Zhang, Hao, A.C. Berg, M. Maire and J. Malik. 2006. ‘SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition’. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:2126–2136. doi:[10.1109/CVPR.2006.301](https://doi.org/10.1109/CVPR.2006.301).
- Zhou, Li, Zongtan Zhou and Dwen Hu. 2013. ‘Scene classification using a multi-resolution bag-of-features model’. *Pattern Recognition* 46, number 1 (January): 424–433. ISSN: 0031-3203. doi:[10.1016/j.patcog.2012.07.017](https://doi.org/10.1016/j.patcog.2012.07.017).

Appendix A

Implemented Techniques

This appendix provides a list of the techniques that were included in the developed framework.

A.1 Features

In order to take advantage of colour information, the following representations were included:

- Grayscale
- Opponent colour space
- Hue, Saturation, Value colour space

A.2 Descriptors

Three descriptor types were added to the framework:

- Scale Invariant Feature Transform
- Histogram of Oriented Gradients
- Local Binary Patterns

A.3 Extraction Method

The extraction methods in the framework correspond to:

- Dense
- Multi-scale

A.4 Descriptor Transform

The following descriptor transforms are available:

- Principal Component Analysis
- Hellinger Kernel

A.5 Intermediate Representations

To generate a codebook and encode the descriptors, the following techniques were implemented:

- Bag-of-Words
- Fisher Kernel

A.6 Spatial Information

The framework contains two techniques to add spatial information to the image representations:

- Spatial Pyramid Matching with square partitioning
- Spatial Pyramid Matching with slice partitioning

A.7 Learning Techniques

These techniques are available to classify an image:

- Linear Support Vector Machine
- Support Vector Machine with support for kernels

Appendix B

Configuration Parameters

This appendix describes the parameters that are present in the configuration file of the framework. The values that they can take and a short description of what they do is also included.

B.1 image

The parameters in the ‘image’ section are related to the processing of images after they are loaded and before the descriptors are extracted.

B.1.1 type

Type	string
Valid values	“Greyscale”, “Opponent”, “HSV”
Description	The colour space of the image. ‘Opponent’ and ‘HSV’ result in images that are three times larger than ‘Greyscale’.

B.1.2 maxResolution

Type	number
Valid values	integers > 0
Description	The maximum resolution of an image.

B.1.3 forceSize

Type	boolean
Valid values	“true”, “false”
Description	If enabled forces all images to have their largest side equal to the ‘maxResolution’ value. Otherwise, only larger images will be scaled.

B.1.4 smoothingSigma

Type	number
Valid values	floating-points ≥ 0
Description	The amount of smoothing to apply to the image. 0 disables smoothing.

B.2 features

The parameters in this section are related to the extraction of features of an image.

B.2.1 type

Type	string
Valid values	“SIFT”, “HOG”, “LBP”
Description	The type of descriptor to use. ‘SIFT’ has 128 dimensions, ‘HOG’ has 36, and ‘LBP’ has 8.

B.2.2 gridSpacing

Type	array of numbers
Valid values	arrays of integers > 0
Description	The spacing between each extracted descriptor. When the list contains more than one element, multiple sets of descriptors will be extracted for each spacing, using the ‘patchSize’ in the corresponding position of the list.

B.2.3 patchSize

Type	array of numbers
Valid values	arrays of integers > 0
Description	The scale of the extracted descriptor. When the list contains more than one element, multiple sets of descriptors will be extracted for each size, using the ‘gridSpacing’ in the corresponding position of the list.

B.2.4 transforms

Type	array of strings
Valid values	[“Hellinger”]
Description	A list of transformations to apply to the descriptor after it has been extracted.

B.3 codebook

The ‘codebook’ section contains all parameters related to the creation of the codebook.

B.3.1 type

Type	string
Valid values	“Fisher”, “KMeans”
Description	The type of codebook to use to encode the descriptors.

B.3.2 codewords

Type	number
Valid values	integers > 0
Description	The number of codewords to use in the generated codebook.

B.3.3 pcaDimension

Type	number
Valid values	integers > 0 and $<$ the original size of the descriptor
Description	The final size of the descriptor after applying PCA.

B.3.4 textonImages

Type	number
Valid values	integers > 0 and $<$ the total number of training images
Description	The total number of images to use to build a codebook.

B.3.5 totalFeatures

Type	number
Valid values	integers > 0
Description	The upper limit of the number of features to use when computing the codebook. If the total number of descriptors in ‘textonImages’ exceeds this value, a random subset will be chosen.

B.4 histogram

The parameters in the ‘histogram’ section are related to the spatial information. The values in this section only have an effect when the codebook being used is ‘KMeans’.

B.4.1 type

Type	string
Valid values	“Slices”, “Squares”
Description	The type of spatial partitioning to use when building a spatial pyramid.

B.4.2 pyramidLevels

Type	number
Valid values	integers ≥ 0
Description	The highest level in the pyramid. A value of zero means that only the full image is used and the partition type has no effect.

B.5 classifier

The ‘classifier’ sections defines all parameters related to the classifier type and settings.

B.5.1 type

Type	string
Valid values	“Linear”, “SVM”
Description	The type of classifier to use.

B.5.2 c

Type	number
Valid values	floating-point > 0
Description	The misclassification penalty to apply to outliers when training the classifier.

B.5.3 trainImagesPerClass

Type	number
Valid values	integers > 0 and $<$ the number of images in the smallest class
Description	The number of images for each class to train the classifier.