Approximate string matching distance for image classification

Hong-Thinh Nguyen

Ecole Doctorale ED488 Sciences, Ingenierie, Sante

Approximate String Matching Distance for Image Classification

by

Hong-Thinh NGUYEN

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Image processing and Computer Vision

JURY 2014/08/29:

Patrick Lambert	Professeur Université de Savoie	Rapporteur
Johel Mitéran	Professeur Université de Bourgogne	Rapporteur
Christian Wolf	Maître de Conférences HDR Université de Lyon	Examinateur
Amaury Habrard	Professeur Université Jean Monnet, Saint-Étienne	Examinateur
Christophe Ducottet	Professeur Université Jean Monnet, Saint-Étienne	Directeur
Cecile Barat	Maître de Conférences Université Jean Monnet, Saint-Étienne	Co-directeur

ABSTRACT

The exponential increasing of the number of images requires efficient ways to classify them based on their visual content. The most successful and popular approach is the *Bag of visual Word* (BoW) representation due to its simplicity and robustness. Unfortunately, this approach fails to capture the spatial image layout, which plays an important roles in modeling image categories.

Recently, Lazebnik et al (2006) introduced the *Spatial Pyramid Representation* (SPR) which successfully incorporated spatial information into the BoW model. The idea of their approach is to split the image into a pyramidal grid and to represent each grid cell as a BoW. Assuming that images belonging to the same class have similar spatial distributions, it is possible to use a pairwise matching as similarity measurement. However, this rigid matching scheme prevents SPR to cope with image variations and transformations.

The main objective of this dissertation is to study a more flexible string matching model. Keeping the idea of local BoW histograms, we introduce a new class of edit distance to compare strings of local histograms.

Our first contribution is a string based image representation model and a new edit distance (called SMD for String Matching Distance) well suited for strings composed of symbols which are local BoWs. The new distance benefits from an efficient Dynamic Programming algorithm. A corresponding edit kernel including both a weighting and a pyramidal scheme is also derived. The performance is evaluated on classification tasks and compared to the standard method and several related methods. The new method outperforms other methods thanks to its ability to detect and ignore identical successive regions inside images.

Our second contribution is to propose an extended version of SMD replacing insertion and deletion operations by merging operations between successive symbols. In this approach, the number of sub regions ie. the grid divisions may vary according to the visual content. We describe two algorithms to compute this merge-based distance. The first one is a greedy version which is efficient but can produce a non optimal edit script. The other one is an optimal version but it requires a 4th degree polynomial complexity. All the proposed distances are evaluated on several datasets and are shown to outperform comparable existing methods.

Keywords: Edit distance, Bag of visual words, Spatial matching, Image classification.

Acknowledgements

First of all I would like to thank my supervisors, Prof. Chirstophe Ducottet and Cecile Barat. Ever since they recruited me in Octobre 2010 to do research under their guidance, they offered me excellent work environment as well as various kinds of general support and advices. I am grateful that they gave me time, ideas, suggestions, corrections, comments and a lot of help. I could not have succeeded without the invaluable support of them. Without them I may not have gotten to where I am today.

My sincere thanks to the rapporteurs and examiners in my jury, Prof. Patrick Lambert and Prof. Johel Mitéran, Prof. Christian Wolf and Prof. Amaury Habrard, for their valuable questions, insightful comments and constructive suggestions which can guide me for my future research.

This thesis would not have started without the financial support by Vietnam government, project 322. I am also thankful for the support by the Laboratoire Hubert Curien for supporting me a nice and friendly research environment.

Many thanks to my colleagues in the laboratoire, my Vietnamese friends for for making my PhD student life an enjoyable and memorable one. Special thank to Amany and Coco who stayed in same office with me for nearly one year. Ill never forget the many wonderful lunches and fun activities we have done together. I also would like to thank Luu Manh Ha, my close friend from my bachelor, who always gave me help and comments when I need.

Above all, I thank my family for being the pillars of my life. I thank my parents for guiding me through life and encouraging me to pursuit my dreams. To Tung, my husband, thanks for showing your patience and giving me moral support. To Chom Chom, my little girl, thanks for giving me your unconditional love and every sweet moment.

Contents

List of Figures

List of Tables

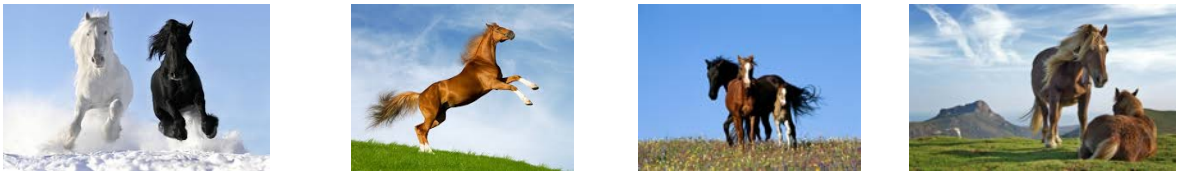
Chapter 1

Introduction

1.1 Background and Objectives

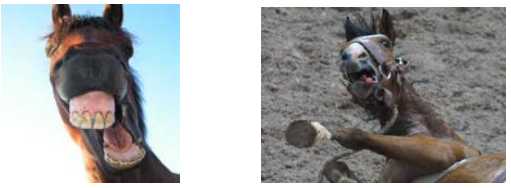
We are living in a digital era, where data are saved, accessed and shared efficiently on digital format. Particularly, digital images are more and more present in our daily life since image capture devices have become more cheaper and popular, image sharing sites offer almost unlimited storage and finally high speed internet access enables to manipulate and exchange images easily. An obvious consequence is the exponential growth of the number of images along with an increase of the number of databases and the number of categories within those databases. The tasks of classifying, organizing and accessing these huge image databases become very challenging.

Initially, some proposed solutions to solve these tasks were based on the use of text meta data. However, using text captions, tags or descriptions available with images is not always a good option because this information is not always available, not always relevant, or can strongly depends on a personal point of view. Of course, manual indexing of images is just not possible, especially in case of very huge databases.
Following the old saying *a picture is worth a thousand words* , a new trend is to use directly the image visual content with the help of computer vision techniques. It seems to be a feasible solution, although several factors make this task very challenging. These factors include as scale, viewpoint, illumination changes, occlusion, background clutter, deformation (Figure 1.1), but also inter-class variations (Figure 1.2) or intra-class variations (Figure 1.3).

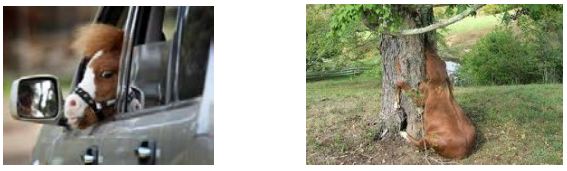


(a). Horses are captured at different scales and poses.

(b). Light conditions change.



(c). Deformation



(d). Occlusion

(e). Background clutter

FIGURE 1.1: Examples of images containing instances of the same object class (horses) that look very different because of different factors. Visual content based recognition/classification approaches need to be very robust to these variations.

Lobster Crayfish Ibis Flamingo

FIGURE 1.2: Examples of inter-class variations: images containing instances of different object class (lobster and crayfish, ibis and flamingo) can be incorrectly classified because of their very similar shape or color

FIGURE 1.3: Examples of intra-class variations: the chairs are in various shapes and colors. It is not easy for a machine to learn and classify them.

Since the 2000s, thousands of publications have focused on finding a good solution for visual content classification and retrieval. One of the most successful approach is the Bag of visual Words (BoW) representation [Sivic and Zisserman, 2003; Csurka et al., 2004]. The idea of this method is to use a set of image local features, to quantize them and finally present an image as a histogram of its quantized local features. The strengths of the method are its simplicity, its computational efficiency and its invariance to affine transformations, as well as occlusion and lighting variations. This representation led to many state of the art results in different vision tasks, especially in object and scene classification.

One of the main drawbacks of the Bag of Word approach is that it does not use any spatial information. An image is represented as a occurrence histogram of its visual words regardless of their position. Thus, two images having the same visual words but located at different positions will be considered as similar whereas their visual content may be totally different.

To overcome this problem, *spatial matching* based approaches such as [Lazebnik et al., 2006; Cao et al., 2010; Li et al., 2011] have successfully incorporated some spatial information into the matching scheme. The idea of such approaches is to divide images into sub-regions and to compute a region to region matching between local sets of features. Each sub-image is described by a local BoW and the matching is done between the corresponding local BoWs.

Local Bag of Word approaches have also some drawbacks. Firstly, the image decomposition scheme must be predefined and applied identically on all images, without taking into account their visual content. Although the rigid partitioning

of an image into rectangular blocks preserves certain spatial information, it often breaks objects into several blocks or puts several parts of different objects into a single block. Thus, visual information about objects, which could be useful for image categorization, may be destroyed by this rigid partitioning.

Moreover, when comparing two images, the region by region matching is sensitive to spatial translation of objects. Indeed, in the case of two images depicting the same object but at different positions, the matching could fail while the content is similar.

In addition, several works of [Smith and Li, 1999; Li et al., 2000; Gokalp and Aksoy, 2007] have shown that regions relationships can considerably improve classification performance.

Motivated by those results, we consider the problem of improving the standard local Bag of Word model. Our objective is to create a new image representation and a new matching strategy which takes regions relationship into account.

1.2 Contributions

The main contribution is to improve the matching scheme in local BoW approaches by incorporating neighboring information between regions. For that purpose, we propose to represent images as sets of strings of local BoW and to use string matching tools to compare them.

In this dissertation, we first investigate limitations of rigid Local Bag of Word matching and advantages of using strings to capture the order of regions inside an image. From this analysis, we propose to represent an image as a set of *strings of histograms*. We carefully analyze several string-based approaches and introduce a *pyramidal strategy* to get an improved representation.

Second, we introduce an approximate *String Matching Distance* (SMD), specifically adapted to our strings of histograms in the context of image comparison. After reviewing several string comparison approaches as *Levenshtein edit-distance* [Levenshtein, 1966], sequence matching [Yeh and Cheng, 2008], video matching [Ballan et al., 2010], we propose a new definition of edit operation costs more suitable for string of histograms representing images. The new distance can be computed efficiently by Dynamic Programming. Its performance is evaluated on classification tasks and compared to the standard method and several related

methods. The new method outperforms other methods thanks to its ability to detect and ignore identical successive regions inside images.

Motived by the very promising performance of SMD, we then propose to extend it by introducing a new edit operation called merging, which allows to combine successive symbols to create a new one. In this approach, the number of sub-regions i.e. the grid divisions may vary according to the visual content. More precisely, although we first use a fix grid to decompose the image into regions, this grid is automatically modified during the string matching phase since successive regions from both strings can be merged together. The new distance then brings more flexibility and precision to the matching process.

In addition, we describe two algorithms to compute this merge-based distance. The first one is a *greedy approach* which can be computed efficiently by Dynamic Programing but can produce a non optimal edit script. To get the optimal matching, we propose a recursive approach. However, this new algorithm has an exponential complexity which is not suitable for practical purpose. We then present a new optimal recursive version which requires only a 4^{th} degree polynomial complexity. All the proposed distances are evaluated on several datasets and are shown to outperform comparable existing methods. It confirms the advantage of our idea of using merge operation in string-based edit distance for image comparison.

1.3 Thesis outline

The dissertation is organized as follows:

Chapter 2 reviews important works related to the thesis. First, we discuss about the image representation and advantages of using local based representations. Then, we present in details the Bag of Word framework. The main limitation of the BoW, which is that it ignores spatial information of local features, is pointed out. We pay special attention on Spatial Pyramid Representation (SPR) and its alternative representations which successfully incorporate spatial information into the BoW. Then, SPR drawbacks about spatial matching are discussed.

Chapter 3 introduces the new image representation as strings of region histograms. In parallel, we present the approximate String Matching Distance to

compare such string-based representations. Then, we explain carefully the computation algorithm based on dynamic programming. The new approach is validated on several well-known datasets, in order to study the impact of image string parameters and demonstrate advantages of the proposed distance.

Chapter 4 discusses about a new merge-based String Matching Distance. It is an edit distance which supports merge-operation allowing to group similar regions into a new one. To compute this distance, we describe two different algorithms. We compare all versions of String Matching Distance in terms of classification performance and in execution time. Then, we discuss about the pros and cons of each approach.

Chapter 5 summarizes the main ideas and suggests several directions of future work.

Chapter 2

Related Work

Abstract:

This chapter reviews relevant background knowledge and related works in the literature. The image classification problem is first discussed in section 2.1. Then, the next section 2.2 studies the image representation and gives details about the Bag of Word model in section 2.3. The Spatial Pyramid Representation and its alternative extensions are presented in section 2.4. Section 2.5 reviews approaches proposed for computing similarity between such representations and recalls notably the definition of several metrics for image classification. Section 2.7 sums up the chapter.

2.1 Introduction

Assuming we have a collection of already categorized images (training set) and a set of unknown ones (test set), the objective of content-based image classification is to assign unknown images to one or more semantic categories based on their content. Since we have no information about a given unknown image, the simplest way is to compare it to images from the training set and then assign it the label of the closest-one. This is the 1-nearest neighbor approach. In a more accurate way, we can use some learning approaches (e.g SVM). The distances between images are used as input of a learning classifier and the final class model is used to categorize unknown images. However, even with the help of a very strong machine learning technique such as a SVM classifier, image classification is still a very challenging problem as described in Chapter 1 since we need to consider various

image transformations, viewpoints, scales and also inter/intra class diversity. The question is: How to compare two images efficiently and correctly based on their contents? To answer this question, there are two points to tackle: first, the image representation, i.e. how to encode an image content into a robust and compact form; second, the image matching, i.e. how to compare two images with such image representation. In this chapter, we review relative issues on these questions.

2.2 Image representation

The purpose of image representation is to convert an image content into a compact and comparable form. A good image representation approach should encode all the related information about the visual content of the image. What information in the image is considered relevant depends on the task and dataset. For instance, to classify two scene images, color and texture are two visible features which can be used; but for two object images, shape information may be more suitable. Furthermore, depending on the information used, the image can be represented globally (using the whole image information, e.g color histogram...) or locally (using set of image regions). Each representation type has advantages and limitations.

The first image representations were global such as *color histogram* [Swain and Ballard, 1991], combination of *color and shape* [Jain and Vailaya, 1996], *texture* [Manjunath and Ma, 1996]. The advantages of global representations are that they are compact, fast to compute and invariant to layout of image parts [Krapac, 2011]. However, since they are constructed from all image pixels, any variation in the image content may probably affect the final image representation. This makes it difficult to obtain invariance to transformations, light changes or noise. When object are involved, as in object detection and recognition tasks, global representations are not robust to background cluster, occlusions and deformations. Therefore, global representations are used most of the time for scene classification purpose like indoor/outdoor or landscape/city classification. These approaches have more difficulties to handle a large number of classes.

To overcome the limitations of global image representations, the image is represented in a discriminative way, by a collection of local features. The successful development of local feature detectors and descriptors has had a tremendous impact on research in object recognition and image classification. Those features have made possible to develop robust and efficient recognition approaches that can

operate under a wide variety of viewing conditions and under partial occlusions [Grauman and Leibe, 2011]. The most well known local features based representation is the Bag of Word model. More details about this model will be discussed below.

2.3 The Bag of Word representation

The Bag of Word representation is one of the most famous techniques in computer vision. The method was first applied in the text domain [Harris, 1954]. For text document classification task, a document is represented as a set of single words. This representation is done without taking into account the grammar and even the word's order. Fundamentally, the document is categorized by using only information of the words's appearance and their frequency.

The Bag of Word model was further adapted to computer vision applications by treating each image as a document of *visual words*. The principle key of the model is the use of *local features* to generate the image visual words. In recent years, the approach has drawn a lot of researchers's attention due to its effective performance, computational efficiency, and simplicity. There is an increasing number of publications using the Bag of Word model everyday. This confirms that the Bag of Word representation is the most successful and suitable approach for object and scene classification.
The Bag of Word approach contains several steps. Figure 2.1 summaries its framework. According to the figure, the approach can be divided into four main steps:

- Local feature detection: Extracting interesting regions or points from image.
- Local feature description: Computing descriptors over these regions/points.
- Coding: Representing the distribution of local descriptors in a compact way. This step is usually based on vector quantization and consists in two phases: (i) Building a visual vocabulary from a random set of feature descriptors, (ii) Assigning the descriptors to the different visual words using a soft or hard strategy.
- Pooling: Constructing the final vector by either sum or max pooling.

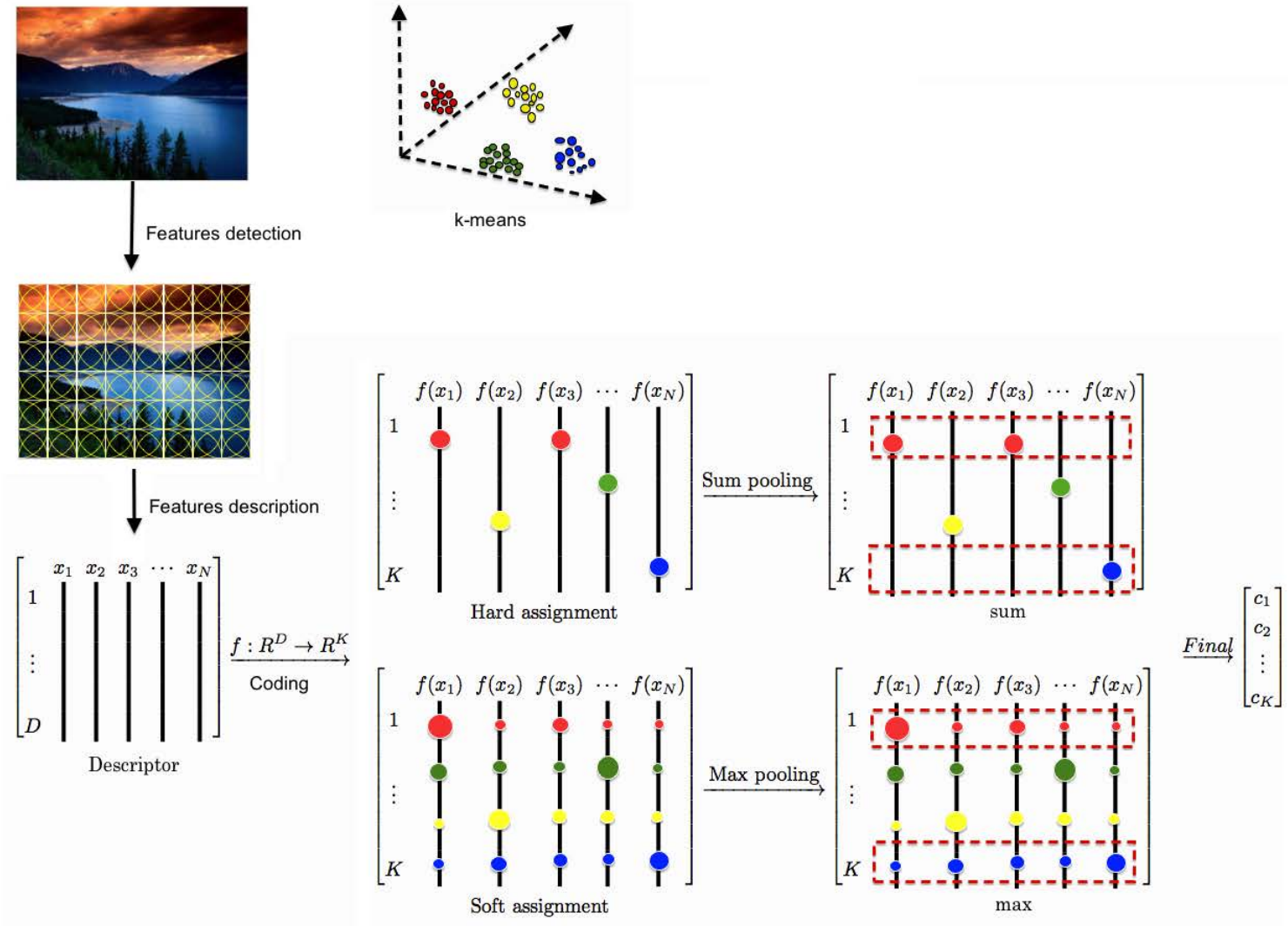


FIGURE 2.1: The Bag of Word image representation baseline: First, local features are extracted from the image either by interest point detector or dense sampling detector. These local features are then encoded as descriptors (e.g SIFT). At the coding step: first, a clustering approach such as *k-means* is applied on a random subset of descriptors taking from all images to generate the visual vocabulary. Then, each descriptor is allocated to one (hard strategy) or a set of visual words (soft strategy). Finally, a vector histogram is computed at pooling step

At each step, many alternative improvements on the standard Bag of Word have been proposed. In following next subsections, we will review them in detail.

2.3.1 Feature detection

The goal of feature detection is to identify some parts of an image that are interesting for a given application. These parts are called *local features* and correspond to specific structures in the image itself, ranging from simple structures such as points or edges to more complex structures such as objects. They are interesting because they can capture information about local neighborhood of interest points in an image and allow to cope with large changes in illumination condition and with image transformations, such as translation, rotation, scaling, and affine deformation [Tuytelaars and Mikolajczyk, 2008]. For that reason, a set of local features are usually used as a robust image representation or to compare two or more images. Generally, local features can be detected and extracted from the image by using feature detector. In the literature, we can separate two kinds of feature detectors: (i) Interest point detector and (ii) Dense sampling detector. Figure 2.2 illustrates two kinds of detector on a sunflower image.

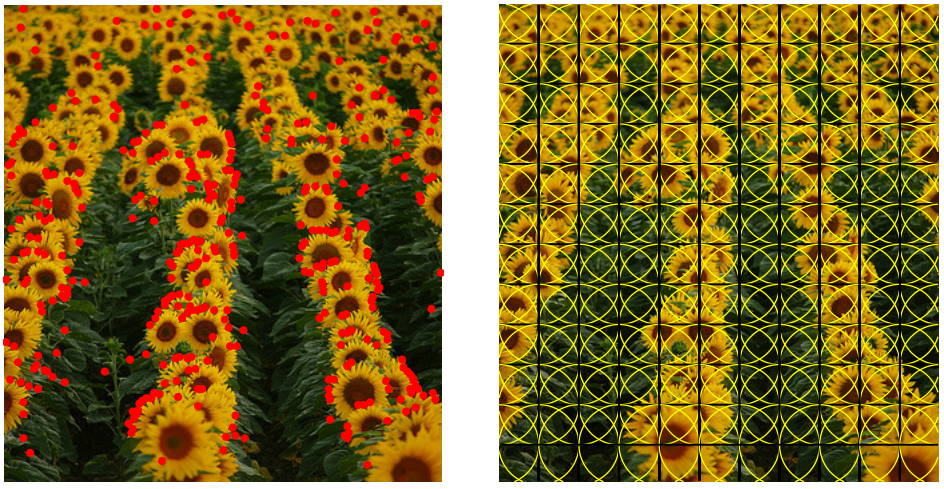


FIGURE 2.2: Example about interest point detector (LoG detector) and dense sampling detector

Interest point detector

Interest point detectors focus on interesting locations in images which can be points, corners, edges, blobs and regions. The earliest interest point detectors are *Harris detector* [Harris and Stephens, 1988] and *Hessian detector* [Lindeberg,

1998]. While Harris detector tries to detect corner-like structures, the Hessian detector intends to detect blobs or regions with strong texture variation. [Lindeberg, 1998] introduced the *Laplacian of Gaussian LoG* detector - a *blob-like* features detector which can cope with different scale invariant regions. [Lowe, 1999] had approximate Laplacian function by a derivation Gaussian function and proposed *Difference of Gaussian* (*DoG*) detector whose performance is same as LoG. In the work of [Schmid et al., 2000], the authors noticed that although those two Harris and Hessian detectors are very robust to image rotations, illuminations changes and noise, they can not cope with scale variations. Then in their later work [Mikolajczyk and Schmid, 2004], they proposed to combine the Laplacian of Gaussian detector with Harris and Hessian detectors and created *Harris-Laplace* detector and *Hessian-Laplace* detector which are scale invariant region detectors. Moreover, they extended those two detectors to yield affine invariant properties (*Harris-Affine, Hessian-Affine*) [Mikolajczyk et al., 2005]. Those detectors can detect the features under large viewpoint changes.
All previous detectors have been used in many computer vision applications specially in image matching and retrieval. More details about experimentation and comparison of those feature detectors can be found in [Mikolajczyk et al., 2005; Tuytelaars and Mikolajczyk, 2008].

Dense sampling detector

While an *interest point* detector considers only special locations in an image, the dense sampling detector, on the other hand, extracts local features at every point on a regular grid. It allows to detect features on contours, corners as well as in uniform regions. The dense sampling detector is believed to be well adapted for a classification task where the image background contains important information.

The most disadvantage of dense sampling approach is that it uses a fix image patch size which causes the inaccuracy of sampling scales and locations. In order to reduce impacts of these problems, *multi scales dense sampling* and overlapping between image patches are frequently applied. Consequently, having more image patches means paying more computational cost for following steps. Nevertheless, due to the computational constraint, it is suggested to use a small random selection of image patches [Nowak et al., 2006]. Recently, [Tuytelaars, 2010] proposed *Dense Interest point*, a hybrid technique and showed that the new detector improves both performance and computation.

2.3.2 Feature description

Once a set of local features has been detected from an image, some encoding algorithms are applied on these local image regions to convert them into local descriptors. The purpose of this step is to make local features robust against scale changes, affine transformations and partially invariant to illumination variations. This means, the descriptors of two local features with the same visual content should be identical even if they appear under different viewpoints and scales. Several feature descriptions have been invented such as SIFT [Lowe, 2004], SURF [Bay et al., 2006], HOG [Dalal and Triggs, 2005])... Among them, the SIFT descriptor is the most widely used one in computer vision, and the one we use in the following.

SIFT

Basically, the SIFT (*Scale Invariant Feature Transform*) descriptor is a spatial orientation histogram of the image gradient. In the original formulation proposed by Lowe [Lowe, 2004], the SIFT descriptor is a combination of DoG based detector and its corresponding features encoder (based on orientation normalization). In fact, the study of [Mikolajczyk et al., 2005] has confirmed that the SIFT descriptor can work with any kind of interest point detectors and also dense sampling method. In recent publications [Lazebnik et al., 2006; Yang et al., 2009; Wang et al., 2010; Boureau et al., 2011], the *dense-SIFT* approach has been showed to better perform on object and scene classification.

To compute the SIFT descriptor, first a grid 4×4 is applied around an interest point (e.g. one image patch). Each window is divided again into 4×4 cells and the gradient of magnitude and gradient of orientation are computed in each cell. A histogram of 8-bin gradient orientation is computed which leads to $4 \times 4 \times 8$ or 128 dimensions. By using the intensity gradient (magnitude and orientation) in the computation, the SIFT descriptor becomes invariant to image rotation and illumination change.

Extensions of SIFT

Several extensions of SIFT have been introduced in literature. The PCA-SIFT [Ke and Sukthankar, 2004] applied *Principal Component Analysis* (PCA) to reduce the dimension of normalized local gradient maps, thus, made new descriptor both faster and more distinctive than the regular SIFT descriptor. GLOH [Mikolajczyk and Schmid, 2005] extended SIFT by using a log-polar grid as opposed to

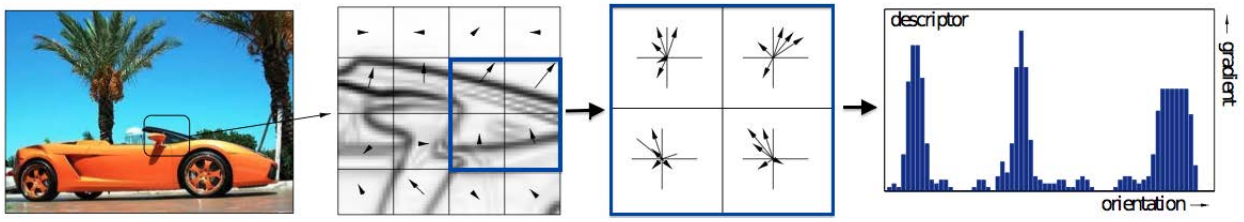


FIGURE 2.3: Computation of SIFT descriptor: At each interesting region of image, image gradient is computed for 4×4 grid. Then at each cell, the histogram of magnitude and orientation of gradient are used to obtain final key point descriptor. Image from [Marszalek, 2008]

a rectangular 4×4 grid and 16-bin orientation against 8-bin. The PCA method is then used to reduce the dimension of the descriptor. From their experimental results, the authors showed that the new descriptor, GLOH, well performed with structured scenes. The SURF descriptor (*Speeded Up Robust Features*) proposed by [Bay et al., 2006] is not an extended version of SIFT but is very closely related to SIFT descriptor. It uses an integer approximation to the determinant of Hessian blob detector instead of the Laplacian operator, and Haar wavelets instead of Different of Gaussian in an image pyramid.

2.3.3 Coding

Visual codebooks are proposed to represent an image in a compact way. The idea of a visual codebook is motivated by using a dictionary to represent the document in text categorization. But unlike the text domain, where the codebook is already given, in the computer vision system, the vocabulary is not yet available and need to be created from training data. Once the dictionary is created, each local descriptor is assigned to visual words and each image is represented as a *Bag of visual Word* (BoW). This process is called *coding*. It is an important step which has strong effect on classification performance.

We detail its two phases: *Codebook generation* and *Codeword assignment*.

Codebook generation:

At this phase, a large set of image descriptors is randomly selected from all training images. A clustering method is applied on this set in order to group similar image descriptors into a same cluster. Many clustering methods have been used

to construct a visual codebook such as *K-means* [Csurka et al., 2004; Sivic and Zisserman, 2003], *Gaussian Mixture Models* (GMM) [Dorkó and Schmid, 2005], *online clustering with mean-shift* [Jurie and Triggs, 2005], and *hierarchical clustering* [Nister and Stewenius, 2006]. Once the clustering is done, each center of cluster becomes a codeword.

All approaches mentioned above are un-supervised learning techniques. Their advantages are simplicity, lower risk of overfitting and computational efficiency. Nonetheless, the visual dictionary can be generated by supervised learning approaches [Moosmann et al., 2006; Perronnin, 2008; Lazebnik and Raginsky, 2009]. These approaches employ the class labels during the construction of codewords. They usually generate a more discriminative codebook which adapts to real-world applications. However, in the other hand, these approaches introduce more complexity. Typically, in the basic Bag of Word pipeline, authors prefer to use unsupervised approaches, like the K-means clustering method.

One more thing to discuss in the codebook generation phase is the codebook size or the number of clusters. This number defines the diversity of the visual dictionary and consequently the quality of our image representation. If it is too small, it is highly probable that quite different descriptors belong to the same cluster making the image representation less specific. In contrast, if the codebook size is too big, we can obtain a more precise representation. However, if this size grows too much, the overfitting phenomenon might occur and similar image patches might be described by different codewords. Typically, the codebook size is selected as the best one after several tests.

Codeword assignment:

Let denote $\mathbf{B} = (b_1, b_2, ..., b_K)$ as the codebook obtained after the previous codebook generation step. b_i denotes its *i-th* visual codeword and K is the vocabulary size. In the codeword assignment phase, each feature descriptor x is assigned to one or several visual words. There are several assignment methods used in the literature.

Hard Assignment Coding or *Hard Coding* or *Vector quantization:* is the easiest way to assign codewords to local descriptors. Each local descriptor is assigned only one visual word which is the closest nearest neighbor in the feature space. Normally, the Euclidean distance is used to compute distances between the descriptor and the centers of the cluster. Let $\boldsymbol{\alpha}$ be the coding coefficient vector of x with α_i being

the coefficient with respect to visual word b_i. Each coefficient α_i is computed as:

$$\alpha_i = \begin{cases} 1 & \text{if } i = \arg\min \|x - b_i\|_2 \\ 0 & \text{otherwise} \end{cases}$$

In spite of its simplicity, hard assignment approach suffers from large quantization errors.

Soft Assignment Coding: The biggest problem of hard assignment coding is the use of only one visual word to code each descriptor, ignoring the relevance of other candidates. This problem can be solved using *Soft Assignment Coding* [van Gemert et al., 2010]. The main idea of soft assignment is that each local descriptor is described by all codewords with corresponding weights. The weight are the coding coefficients and are computed as a Gaussian function of the distance between the descriptor and the visual codebook:

$$\alpha_i = \frac{exp(-\beta\|x - b_i\|_2)}{\sum_{i=1}^{K} exp(-\beta\|x - b_i\|_2)}$$

Lately, [Liu et al., 2011] noticed that assigning a local descriptor to all visual words is non-reasonable (even if coding coefficients of non relevant candidates are small). As a consequence, this soft coding scheme degrades the classification performance. The authors proposed a *Semi-Soft Assignment Coding* which each local feature is presented by only its top k nearest cluster centers in the feature space.

Sparse Coding: The sparse coding method also intends to improve hard coding by using more than one visual words to represent a local descriptor. The main idea of *sparse coding* is to find the codebook from which a training set of descriptors can be represented as a linear combination having the lowest number of non-zeros coefficients. In other words, this sparse coding tries to minimize the number of visual words assigned to each feature.

[Yang et al., 2009] are the first authors to propose this sparsity constraint in the BoW framework. The sparse vector α coding a given descriptor x according to codebook $\mathbf{B}$ is determined using the constrain:

$$\alpha_i = \arg\min \|x - \mathbf{B}\boldsymbol{\alpha}\|_2 + \lambda\|\boldsymbol{\alpha}\|_1$$

where the sparsity regularization term $\|\boldsymbol{\alpha}\|_1$ is selected as the l_1 norm. Here, the codebook $\mathbf{B}$ can be obtained by *k-means*, or for better performance, trained by

minimizing the average of $\|x - \mathbf{B}\boldsymbol{\alpha}\|$ over all samples [Boureau et al., 2011]. The main drawback of sparse coding is that two similar features can be encoded by two very different coefficients.

Locality constrained Linear Coding (LLC): introduced by [Wang et al., 2010], is an adaptation of sparse coding with locality constraints. Instead of using sparsity constraint (which tries to represent an input vector x using as least non-zeros basic vectors as possible), LLC approach employs a *locality constraint* (the features located nearby should be represented by the same basic vectors):

$$\alpha_i = \arg\min \|x_i - \mathbf{B}\boldsymbol{\alpha}\|_2 + \lambda \|d.\boldsymbol{\alpha}\|_1$$

where $d = (d_1, d_2, ..., d_K)$ is a penalty weight. Each element d_i is computed as a function of the distance between the local feature x and centre of clusters (i.e. visual codewords) in the feature space: $d_i = exp(\frac{\|x - b_i\|}{\sigma})$

In addition, the authors also introduced a fast approximated LLC using only the top k nearest codewords of x in the feature space in order to reduces computation time such that the approach can be compatible with real applications. Because of the locality constraint, LLC can capture the correlations between similar descriptors. It means that similar image patches, whose descriptors are closed in the feature space, will be represented by similar coding coefficients.

2.3.4 Pooling

After the coding step is the pooling step which aims to convert a set of visual words of an image into a single vector. The illustration of the pooling step is shown on Figure 2.1. The earliest pooling approach is *sum pooling* [Sivic and Zisserman, 2003; Csurka et al., 2004] where we count the number of occurrences of visual words in the image. In this case, an image is represented as a histogram of codewords. If we normalize this histogram to the total number of visual words in the image, sum pooling becomes *average pooling*.

Lately, [Yang et al., 2009] introduced *max pooling* with sparse coding. The final vector is computed as the maximum value of each dimension over the set of coding vectors. The max pooling method has been showed well suited to other sparse-based representations, e.g *semi-Soft assignment, LLC...* and regularly used together with a *linear classifier* to get better performances.

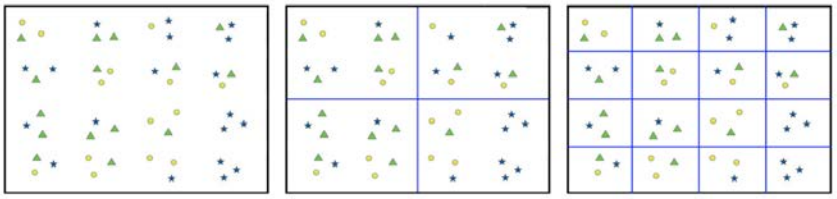
2.3.5 Discussion

The Bag of Word model is the state of the art approach for image classification because of its simplicity and robustness. With this representation, we can efficiently and easily apply a classification technique like Nearest Neighbors, SVM, random forest, etc...to classify an image. The Bag of Word model has been improved in several ways by enhancing each of its steps. For instance, in the coding step, a lot of approaches have been proposed to create an efficient vocabulary. In the assignment step *vector quantization* has been replaced by *Soft assignment* and recently by *Sparse Coding or LLC coding*. Single image features (e.g texture) have been replaced by multiple features (color, texture, shape) [Gehler and Nowozin, 2009].
Another research direction investigated in recent years is to overcome its orderless limitation. Indeed, the main drawback of the Bag of Word model is that it ignores the spatial layout of local features. Such spatial information is thought to play a vital role in modeling object categories. In this context, the *Spatial Pyramid Representation* [Lazebnik et al., 2006] is probably the most notable work.

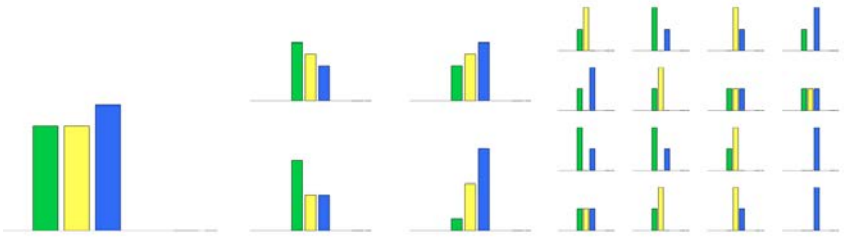
2.4 Spatial Pyramid Representation

2.4.1 Principle

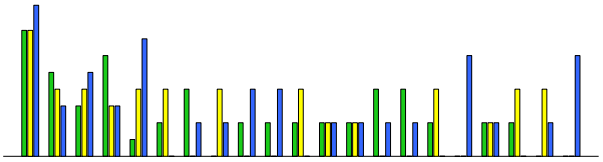
The basic assumption of *Spatial Pyramid Representation* (SPR) is that certain features tend to appear in certain spatial areas in images belonging to the same class. Therefore, the authors suggest to use a regular grid to locate image layout. Consequently, if two images are similar, it is highly probable to find the same local features at the same locations. A histogram of visual words is computed for each image sub-region at each resolution level. The spatial pyramid representation is a collection of local Bag of Word computed over cells defined by a multi-level recursive image decomposition. The authors used the number of matches over different image resolutions as a measure to compare two images. This idea is an extension of *Pyramid Matching kernel* [Grauman and Darrell, 2005], where the histogram intersection kernel is computed between two corresponding regions to count the number of matches. With a penalty weight, the matching at finer level of resolution has more effect compared with the matching at coarser level. The higher the total number of matches the more similar the distribution of local



(a) Tiling image into regions



(b) Computing local histogram for each region



(c) Concatenating all local histograms into single one

FIGURE 2.4: A toy example about Spatial Pyramid Representation proposed by [Lazebnik et al., 2006]: First, the image is divided into multiple regions at multiple scales, e.g. $1 \times 1, 2 \times 2, 4 \times 4$ regions. The local histogram is calculated at each cell and at each resolution. Finally, all local histograms are multiplied with corresponding weights for each level before concatenating into single histogram-the final image representation.

features is. For that reason, in case of objects and scenes which are well aligned in their images, the SPR is a good choice. The approach has several advantages:

- The image is partitioned into regions according to a predefined grid and thus no segmentation is required.
- Information about the location of local features is efficiently used in the pyramid matching step.
- Thanks to the weighting scheme a finer level have a higher weight. Indeed the matching computed at coarser level still contains the matching at a finer level.

Figure 2.4 illustrates in detail the implementation of SPR approach.

2.4.2 Extensions of SPR

By successfully incorporating spatial information into the Bag of Word model, the SPR achieved a significantly better performance over BoW and became the state of the art. Despite previous successes, the Spatial Pyramid Representation still has several limitations which come from the partitioning, the weighting scheme and the features matching.

Recently, several alternative approaches trying to improve the Spatial Pyramid Representation were proposed.

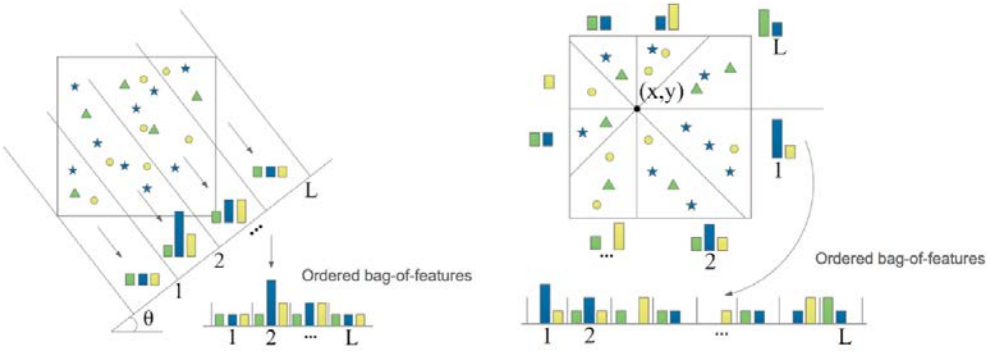


FIGURE 2.5: Spatial Bag of Features (SBoF) approach of [Cao et al., 2010]. The authors try to partition an image either with a predefined direction to deal with translation or into multiple circular sections to cope with rotation.

FIGURE 2.6: Pyramid rings approach of [Li et al., 2011]. The authors suggest to use a set of rings whose centers are the center of the image to locate the image layout.

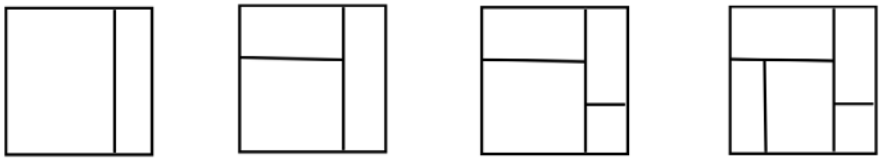


FIGURE 2.7: Learning adaptive partition grid of [Sharma et al., 2011] which for each image class, tries to find the optimal way to split the images recursively

[Marszaek and Schmid, 2006] noticed that, SPR uses three layers of grids 1×1, 2×2, 4×4 which makes the total length of the final concatenated histogram 21 times

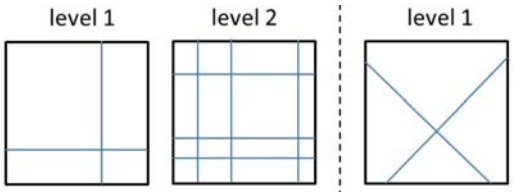


FIGURE 2.8: Randomized partition of [Jiang et al., 2012] where the image is divided randomly in many possible spatial partitions.

longer than the original Bag of Word. It is not convenient when increasing the number of training images or with a large dataset. The authors also remarked that in the general case, the visual image content has vertical distribution. Therefore, they suggested to use a 3×1 grid to replace a 4×4 one in order to reduce the final vector dimension while still capturing the visual information of the image plane.

[Cao et al., 2010] noted that the uniform partitioning is not invariant to scale, translation and rotation. Thus, they introduced a new framework, called *Spatial Bag of Features* (SBoF) (Figure 2.5) which either partitions the image recursively with a predefined direction or divides an image into multiple circular sections to cope with rotation problem. The drawback of this method is that it heavily depends on the choice of direction plane and centre location. Using all possible lines angle and center locations leads to an extremely high dimensional histogram representation for an image, which causes high computational cost. Motivated by this, [Li et al., 2011] suggested to replace the grid partitioning and circular sectors by a set of rings whose centers are the center of the image (see Figure 2.6). The approach is then less complicated compared with SBoF.

[Sharma et al., 2011] noticed that using an unique and uniform partition grid for all classes clearly reduces the classification performance. They proposed to learn the partition grid which adapts to different classes to obtain optimal way to split images so we can obtain a higher classification performance. They fixed the number of tilling then divided images recursively in each direction. For example in Figure 2.7, first, it tries to find the best way to divide the image into two regions. Then, from this two-regions grid, it finds the best ways to divide the image into 3 regions, and so on. For each image class, all patterns are tested to find the best, the most suitable one which is used as a final spatial layout to partition images of this class. However, it is not guarantee that the matching between local regions as used in SPR is still working.

Recently, [Jiang et al., 2012] proposed *Randomized Spatial Partition* (RSP) to characterize the image layout by randomizing partitions (Figure 2.8). The image is divided randomly in many possible spatial partitions (either rotation) so that it can discover the descriptive partition patterns that better represents the spatial configuration of the category. Moreover, different from [Sharma et al., 2011] approach where one grid per class is used, the RSP-based method combines several suitable partition patterns. This makes the image representation more robust as it is less sensitive to the spatial quantization error. However, since they use a lot of partition patterns per class to divide images (either rotation), the matching process and image comparison in this method is not clear and very questionable. Also, both methods have the main drawback that they are computationally expensive.

Although successful incorporating spatial information into BoW, those approaches still use *region to region matching*. Two regions of two images are matched because they are situated at the same location in image layout, not because they have the same visual content. So, this matching scheme introduces a lot of mismatches due to image transformations such that rotation, translation and scaling. We will discuss more about spatial matching in the next section.

2.5 Image comparison

Once an image is represented as a Bag of Word or a set of Local Bag of Words, the question that arises is how to compare two images.

Bag of Word matching:

While the image is represented as a vector of visual words, the matching between two images can be computed directly as the ground distance between their two histograms. The reason is, since we assign the visual words for the local features, we already assumed that: the local features are represented by the same words means they have same visual contents. Therefore, they can be matched. The bin-to-bin distance is the easiest way to measure this match.

The formulas of several frequently used bin-to-bin ground distance can be found bellow. Here, we keep the same notation in all formulas, where $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$ are two feature vectors in R^n space.

- l_1 *or Manhattan distance*

$$d_{l_1} = \sum_{i=1}^{n} |x_i - y_i|$$

- *L_2 or Euclidean distance*

$$d_{l_2} = \sqrt{\left(\sum_{i=1}^{n} |x_i - y_i|^2\right)} = \left(\sum_{i=1}^{n} |x_i - y_i|^2\right)^{1/2}$$

- *χ_2 Distance*

  The distance is computed on normalized vectors. Let us note $\bar{x} = \frac{x}{\sum_{i=1}^{n} x_i}$ and $\bar{y} = \frac{y}{\sum_{i=1}^{n} y_i}$ the normalized vectors of x and y. The χ_2 distance is:

$$d_{\chi_2} = \frac{1}{2} \frac{\sum_{i=1}^{n} (\bar{x}_i - \bar{y}_i)^2}{\bar{x}_i + \bar{y}_i}$$

- *Histogram Intersection Distance*

$$d_{inter} = \sum_{i=1}^{n} min(x_i, y_i)$$

Recently, several authors proposed to use *cross-bin* distances, such as *Earth Mover's Distance* [Rubner et al., 2000] to compare such visual feature histograms. Applying EMD, the images similarity is computed as the amount of changes necessary to transform one image feature into another.

- *Earth Mover's Distance*

  The idea of EMD is based on a solution to the well-known transportation problem. Suppose that the suppliers $P = \{(p_1, w_{p_1}), \ldots, (p_m, w_{p_m})\}$, each supplier p_i having a given amount of goods w_{p_i}, are required to supply the consumers $Q = \{(q_1, w_{q_1}), \ldots, (q_n, w_{q_n})\}$, each consumer q_i having a given limited capacity w_{q_j}. For each supplier-consumer pair, the cost of transporting a single unit of goods is given by d_{ij} (d is called *ground distance*). The transportation problem is then to find a least-expensive flow of goods F from the suppliers to the consumers that satisfies the consumers' demand. Because all customers can receive goods from all suppliers, the Earth Mover's Distance is a *cross-bin distance*. Let f_{ij} being the flows between p_i and q_j, that minimizes the overall cost. The EMD is:

$$EMD(P, Q) = (\sum_{i,j} f_{ij} d_{ij}) / (\sum_{i,j} f_{ij})$$

with some constraints:

$$\sum_j f_{ij} \leq p_i$$

$$\sum_i f_{ij} \leq q_j$$

$$\sum_{i,j} f_{ij} = min(\sum_i p_i, \sum_j q_j)$$

Set of local Bag of Words matching:

The Bag of Word matching procedure does not use any spatial information therefore it may introduce some bad-matches between local features of two images. The reason is that local features corresponding to different semantic regions can be assigned to the same codeword and be matched. For instance, the *sky* feature and the *sea* feature can be described by the same visual word and are matched but this match is wrong. Since the patches from different categories may have different spatial distributions, such as the *sky* feature is usually on the upper part, the *sea* feature is normally on the bottom of the images, it suggests to use location information of image features to correct the matching process.

Spatial Pyramid Representation based approaches such as [Lazebnik et al., 2006; Cao et al., 2010; Li et al., 2011] have successfully employed features location into the matching scheme. The idea of those approaches are: two features of two images are matched if they are located at same position in the image. When considering SPR based representation, we have to deal with the computation of distances between sets of local BoWs. Generally, the matching is done between features at the same cell-location in the image. In other words, it is a *cell to cell matching*. So although these methods performs better than BoW matching, they are very sensitive to image deformations.

Structure matching:

An image is not only represented by its local BoW but also by the relations among them. There is another direction which tries to model image as structured models of regions like *strings* [Smith and Li, 1999; Yeh and Cheng, 2008; Lu and Ip, 2009; Kim and Grauman, 2010; Hong-Thinh et al., 2014] or *graph* [Duchenne et al., 2011; Wu et al., 2013]. The matching problem then becomes a string matching (or graph matching) problem. For instance, in a string based representation, the image is divided into regions and each region is represented as a symbol of a string. [Yeh and Cheng, 2008] used a raster scanning to construct a 1-D string and used the edit

distance to compare two image strings. [Lu and Ip, 2009] represented each image as 2-D string and proposed *Spatial Mismatch Kernel* to compare those strings. In graph based approach, [Duchenne et al., 2011] described an image as a graph of nodes and edges are image regions and their relationships. They formulated the image graph matching as the optimization of an energy problem. Recently, [Wu et al., 2013] presented a *Spatial Graph* which attempts to use region relationships to improve the matching scheme. The advantage of these approaches is a flexible matching. With string matching, the matching can be done at any position on the string. Moreover, with the properties of edit distance, this matching scheme can also take into account location information of regions/symbol in strings. The graph matching can be seen as an extension of string matching. We will discuss this type of matching for images in more details in the next chapter.

2.6 Image classification and kernels

2.6.1 SVM classification

With the development of digital technology, number of images and number of classes increase significantly. As a consequence, using supervised learning techniques has been a recent trend in image classification due to its better performance compared to the K-NN approaches. The role of a supervised classifier is to learn how to separate the known label images in feature space then reliably predict the classes for unknown ones. A Support Vector Machine [Cortes and Vapnik, 1995] is one of the successful supervised learning approach for visual classification. It typically involves two steps. First, in the *training step*, a class model has to be chosen which can well separate training images in the feature space. In this step, all model parameters are tuned to minimize the error criterion using small subset of training data. When the classifier model is available, it is ready to classify new test images.

The classic classifier is the binary classifier as it works with only two classes (Figure 2.9) It is proposed to construct the maximized margin hyperplane separating the training examples into their two classes. In general, the training examples that are closest to the hyperplane are called *support vectors*. In case of linear classifier, the function to predict the output is:

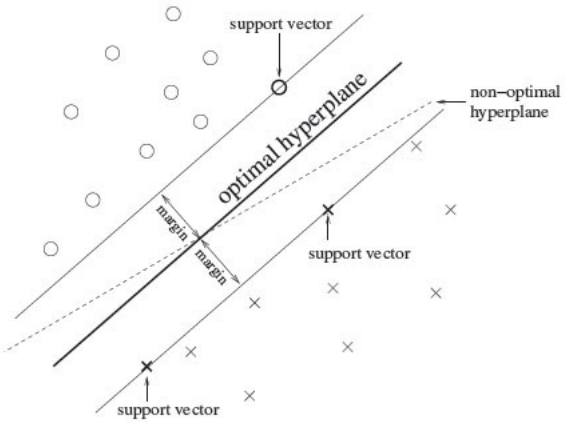


FIGURE 2.9: Example about binary SVM classifier which tries to find optimal hyperplane with maximum margin to classify feature data.

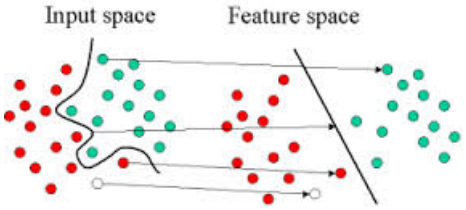


FIGURE 2.10: Example about using a transfer function to map data into a high-dimensional space where a linear classifier can be used.

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i < x_i, x > +b \tag{2.1}$$

where $\{x_i, y_i\}_1^n$ are training samples with $y_i = \{+1, -1\}$ is the label of x_i, α_i are non-zeros, b is bias and $< ., . >$ denotes the dot product.

However, the data does not always have a linear distribution. In case of non linear, a non-linear classifier can be obtained from a linear classifier by using a non-linear map transform function ψ into a high-dimensional space (see Figure 2.10). The decision function will be:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i < \psi(x_i), \psi(x) > +b \tag{2.2}$$

Furthermore, the above equation only needs to know the dot product $< \psi(x_i), \psi(x) >$. The explicit representation for ψ is not involved. The expensive computation of

$\psi(x_i)$, $\psi(x_j)$ and $< \psi(x_i), \psi(x_j) >$ in the transformed space are reduced significantly by defining a suitable kernel function k: $k(x_i, x_j) =< \psi(x_i), \psi(x_j) >$. It is called the *kernel trick* [Aizerman et al., 1964].

2.6.2 Mercer's theorem and kernel function

Mercer's theorem: For a training set $S = \{x_i\}_1^n$ and a function $k(u, v)$.
The kernel matrix (also called the Gram matrix) K_S is the matrix dimension $|S| \times |S|$ where $(K_S)_{ij} = k(x_i, x_j)$.

$k(u, v)$ is a valid kernel if and only if the corresponding kernel matrix is Positive Semi Definite for all training sets S:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} K(x_i, x_j)c_i c_j \geq 0$$

for all choices of real numbers $c_1 \dots c_n$.

Using a valid kernel function k, the decision function is rewritten as:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) > +b \tag{2.3}$$

Some common valid kernels used for computer vision tasks are given below:

- *Linear kernel:*
$$k(x, y) =< x, y >$$
- *χ_2 kernel:*
$$k(x, y) = 2\sum_i \frac{x_i y_i}{x_i + y_i}$$
- *Intersection kernel:*
$$k(x, y) = \sum_i \min(x_i, y_i)$$
- *Gaussian kernel:* $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2)$, for $\gamma > 0$.
- *Edit kernel:* ([Li and Jiang, 2005])
$$k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma D_{ed}(x, y))$$
where $D_{ed}(x, y)$ is the edit distance between the two sequences x and y, $\gamma > 0$

2.6.3 Multiclass SVM

Multiclass problems can be solved by dividing them into multiple binary classification problems. The common technique is to build binary classifiers which distinguish between (i) one class against the rest (so called *one vs all*) and chooses the class for which the corresponding classifier reports the highest confidence score; or (ii) between every pair of classes (it refers to *one vs one*) and chooses the class that is selected by most classifiers.

In the following, we will use the *one vs all* approach since it is reported as a simple yet effective approach in image processing.

2.7 Summary

In this chapter, we have reviewed the Bag of Word model and its main limitation which is the lack of spatial information. We have payed attention on the Spatial Pyramid Representation and its extensions which are very successful to incorporate spatial layout into the BoW. In spite of the fact that these methods have become the state of the art due to their simplicity, several limitations have been discused. Beside, we have also introduced the spatial matching problem for image comparison. Three types of matching has been discussed: the BoW matching, the local BoW matching and the matching using structured data such as string or graph to represent image content. Those methods introduce a flexible matching scheme, and for that reason our motivation in this thesis is to use a string model to incorporate region relationships and approximate region matching. The string edit distance is an effective tool to compare two strings and particularly strings representing image regions. Thus, in this dissertation we aim at combining advantages of using local BoW and string matching to improve image classification. More details about our propositions will be explained in the next chapters.

Chapter 3

Approximate matching for image classification

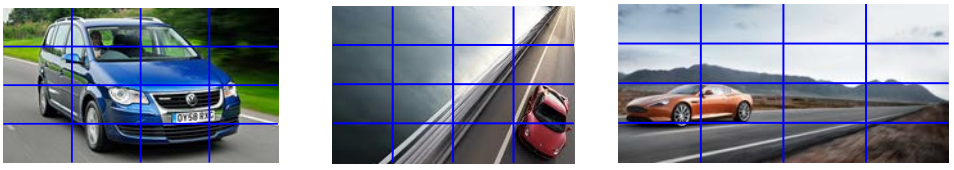
Abstract: *The state of the art representation model for image classification is based on the spatial pyramid representation (SPR). Its principle is to make a spatial partition of the image at different scales and to represent each region as a local Bag of Words (BoW). It assumes that images of the same class have a similar visual content across the spatial partition providing a region by region matching between images. However, this rigid matching is one of the main drawbacks of SPR which prevents this approach to cope with image transformations.*
We propose a new string based image representation together with a new edit-distance. Our goal is to introduce some flexibility in the matching and to integrate some neighborhood relationships between regions. Experiments on several datasets show that our approach outperforms the classical spatial pyramid representation and most existing concurrent methods for classification presented in recent years.

3.1 Introduction

Local feature histograms are widely employed to represent visual contents in various areas of computer vision. In particular, histograms of visual words based on SIFT features, in the well-known Bag of Words model, have proven to be very powerful for image and video classification or retrieval tasks [Sivic and Zisserman, 2003; Csurka et al., 2004]. However, this histogram representation is based on occurrences of image features, completely ignoring the spatial image layout.

In recent years, the significant work Spatial Pyramid Representation (SPR) [Lazebnik et al., 2006] was shown to be successful for classification of objects and scene images (as seen in Chapter 2). In SPR, an image is divided into regions by using predefined regular grids of different scales and by computing a BoW histogram for each region. The image matching is calculated region by region, form coarser to finer grids. Since this approach is very simple and effective, it has received great attention from researchers.

Different aspects of this model have been investigated for the purpose of improving performance leading to systems that reach state-of-the-art results in the domain [Bosch et al., 2007; Yang et al., 2009; Boureau et al., 2011]. Most of SPR-based methods perform well despite the fact that they still keep SPR-rigid matching between corresponding regions. In other words, two regions are matched even if their content is dissimilar, just because they are situated at the same location. Consequently, these approaches are sensitive to geometric transformations. Figure 3.1 shows a car at different viewpoints and scales. It can be noticed that, the SPR-rigid matching is not efficient while considering the matching between image pairs in this example.



(a) (b) (c)

FIGURE 3.1: Example about SPR-rigid matching problem: region to region matching does not work due to image transformations.

Our contribution is to propose a new approach to provide approximate matching between regions by means of : (i) a new image representation as strings of BoW and (ii) a new edit-distance to compare such strings. For each given pair of images, our distance not only takes into account the similarity between pairwise regions as in the standard SPR model, but also integrates information about similarity between neighboring regions. It allows to identify local alignments between subregions or groups of similar subregions in images. In our proposed approach, the number of subregions for different images may vary and is considered according to the visual content, which brings flexibility to the matching process. We validate our approach on several datasets.

The outline of this chapter is organized as follows. We start in Section 3.2 by reviewing several related works on spatial matching. In section 3.3, we describe our representation of an image as strings of histogram. Section 3.4 introduces the edit-distance adapted to strings of histograms and derive an edit kernel. The experiments and results of our edit-distance on image classification tasks are presented on Section 3.5. Finally, we give some conclusions in Section 3.6.

3.2 Related works

Several works have tried to solve this rigid matching problem in the context of image classification or retrieval.

- *Spatial Bag of Features* [Cao et al., 2010]: In this approach, the authors located features position by projecting the descriptors along either lines or circles (see Figure 2.5). Each time, a local histogram is computed for each cell and a final long concatenated histogram is generated. In order to cope with image translation and rotation, a re-arranged procedure is done on every bin of the concatenated histogram by starting from the position with the maximum frequency. The matching is then applied on two re-arranged concatenated histograms. However, this rearrangement scheme may not correspond to the true image transformation and the approach is complex because it requires to consider all directions, spatial bins and center points of circular sectors.

- *Reordered Spatial Pyramid Matching* [Li et al., 2011]: The authors use the local histogram representation: an image is divided into sub regions by a regular grid and a local BoW is computed for each region. Those histograms are then set together in a matrix form, which each row of matrix is one local histogram. After that, this matrix is sorted (by columns) for every bins based on their frequencies. The matching between two images is done on their sorted matrix of histograms. However, this sorted procedure destroys location information of visual words, therefore the matching may be irrelevant.

- *Finding optimal alignment* [Van Kaick and Mori, 2006; Xu et al., 2008; Viitaniemi and Laaksonen, 2010; Yan et al., 2013]: After dividing the images into regions, several authors suggested to relax the rigid matching between two images by using all pairwise regions matching .

[Van Kaick and Mori, 2006] used *Hungarian algorithms* to compute optimal alignment between two images. Each match between two regions of the two images is penalized by a weight (e.g distance between two regions); and the algorithm finds the minimum total weight to match all the regions of the first image to all the regions of the second one. This total weight is used as a similarity measure between two images. The method has the drawback to require the calculation of all pairwise region distances before searching for the best alignment.

[Viitaniemi and Laaksonen, 2010] proposed to use *Integrated Region Matching* [Li et al., 2000] to loosen the constraint of geometrical rigid matching. In this approach, one region of the first image can match several regions in the second image. The distance of two images are the total matching cost between each region pairs. This matching cost is computed by a distance between two regions. Moreover, they proposed two weight parameters, one to decide the reliability of the matching, one to decide the effect of this matching on computing the similarity of two images. Finding the values of these parameters should depend on content of regions and it is not an easy task. [Viitaniemi and Laaksonen, 2010] fixed both parameters as 1. However, this approach performs slightly worse than using rigid matching.

[Xu et al., 2008] introduced *Partially Aligned Pyramid Matching* using the Earth Mover Distance to compare two sets of concatenated local histograms representing two images. This approach can only detect near duplicate images.

Later, [Yan et al., 2013] proposed *Semantic Spatial Matching* approach which assigns each region to one semantic label. The authors kept the SPR image partition scheme, dividing an image into 4×4 identical cells and computing local BoW for each cell. All local histograms from the training images were collected and clustered to create semantic labels. This process is similar to a visual codebook generation. The image was then represented as a histogram of semantic labels. Two regions are matched if they have the same semantic labels and this match does not depend on the regions location. This approach therefore relax the problem of rigid matching, but still has problems with image transformations. Since a fixed grid is used for partitioning, the rotation or the translation of an image will result in a change of the semantic labels of its regions. Moreover, the semantic label generation step can introduce quantization errors.

- *String representations:* Several approaches try to relax rigid matching by approximate matching using string representations [Smith and Li, 1999; Ros et al., 2005; Yeh and Cheng, 2008; Lu and Ip, 2009; Kim and Grauman, 2010]. Compared to other approaches, the string-based approach has advantages of providing the matching of symbols at different positions, keeping into account the order of these symbols. It also enables to work with sequence of regions of different lengths.

  In the early work of [Smith and Li, 1999], the authors proposed to use strings to capture the relative locations of color regions. Their system generated strings of regions by using series of vertical scans on a segmented image. A string of regions is then converted into a *Composite Region Template* descriptor which allows classification using region order information.

  [Ros et al., 2005] proposed to represent each image as a string of interest regions or salient regions and to use string-based edit-distance to compare the content of two images. However, in this system, the string is created from a graph of descriptor points, so it is not robust to image variations and requires expensive computation.

  In the work of [Yeh and Cheng, 2008], the image is first divided into 4×4 regions and each region is described by a local BoW. A raster scan is applied to create a 1-D string of regions where each symbol is one local BoW. The Levenshtein distance [Levenshtein, 1966] is used to measure the similarity between two images. However, in this method, two successive symbols in the string may not correspond to neighboring regions of the image due to the raster scan and the 1-D string representation.

  [Lu and Ip, 2009] proposed a *Spatial Mismatch Kernel* in order to compare two images. In this approach, each image is represented as a combination of two 1-D strings (based on row-wise and column-wise scans) of visual keywords. The distance between two images is defined as a total *mismatch string distance* [Leslie et al., 2002] (i.e. number of similar sub-strings within m mismatch) between two strings (row based or column based) of the two images.

  Lately, [Kim and Grauman, 2010] introduced an *Assymetric Region Matching* approach, computing image similarity between a segmented image and a non-segmented one. The first image is decomposed into regions using a given segmentation method. Each region is mapped to a set of local multi scale dense SIFT descriptors. The authors used two 1-D strings of SIFT-blocks to represent a region (based on row-wise and column-wise scans). In this string

based approach, the string is only used to represent segmented regions and each symbol of the string is a multi-scale SIFT-block. At the matching step, each symbol in the string is matched to several candidates of non segmented image by using the distance of two SIFT-blocks at multiple scales. Given the candidate matches for all symbols, the optimal correspondence between regions and image is computed by dynamic programming. The limitations of the approach are first the requirement of a segmentation process and second the lack of symmetry of the distance between two images (since it depends on the selection of a segmented image and a non-segmented one).

In this chapter, we combine the advantage of string and local histograms representations to propose a new method to represent an image as strings of histograms. In order to measure the similarity of two images, we introduce a new string edit-distance called *String Matching Distance* (SMD) which is adapted to the context of string of regions comparison .

3.3 Image representation

In this section, we discuss how to represent an image as ordered strings of regions preserving spatial relationships of those regions. We focus on the local histogram based representation where an image is first tiled into regions and each local region is described as a histogram of visual words.

Image Partition

There are several ways to decompose an image into regions. Some authors used image segmentation as for example: [Smith and Li, 1999; Van Kaick and Mori, 2006; Kim and Grauman, 2010]. Since, image segmentation is still a difficult task, which introduces errors and needs expensive computation, it is reasonable to replace this step by the division of the image into regions by using a tilling grid. Our grid is defined from an orthogonal basis $(\mathbf{v_1}, \mathbf{v_2})$ aligned in the directions that may best represent the image content. An image is split into B bands having the same width along direction $\mathbf{v_2}$. Then, each band is subdivided into N subregions of same size, along direction $\mathbf{v_1}$. Consequently, the image is divided into $B \times N$ regions. Figure 3.2 illustrates the division of an image into 2 Bands of 3 regions.

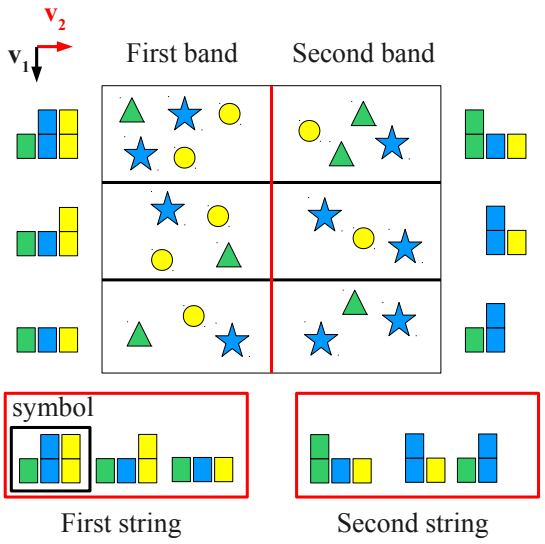


FIGURE 3.2: Example of an image representation as two strings of histograms.

In this work, we only consider the case of vertical and horizontal axes, which is the most simple way to partition an image, since our objective here is to point out the advantages of string representation. However, the method can be generalized to any other basis. Note that, the two directions $\mathbf{v_1}$ and $\mathbf{v_2}$ do not play the same role: $\mathbf{v_1}$ is chosen to be the direction that best represents to the image content.

Actually, in images, there exists a natural sequencing of objects or entities within objects. It is possible to find a principal direction along which the projection of local features may convey information about the image content or capture the essence of the form of an object. Intuitively, as suggested in [Cao et al., 2010], in natural scenes, vertical or horizontal directions can plausibly describe relationships among local features. For instance, the sky is above trees, and trees are above grass. For urban scenes, in [Iovan et al., 2012], the authors propose similarly to replace the SPR grid with divisions along the vertical axis to better take into account the composition of this kind of images. For object images, as proposed in [Tirilly et al., 2008], the major axis of an object can be obtained from the first principal component in a principal component analysis. Distribution of local features along this major axis is similar whatever the orientation or scale of the object is. In the string based approach, [Smith and Li, 1999] suggested to use series of vertical scans along the image to create region strings since in many photographs, the vertical order provides a better characterization of the content.

The graphs of Figure 3.4 highlight the influence of bad matching and direction on partitioning images in classification task on 15 Scenes dataset. The classification

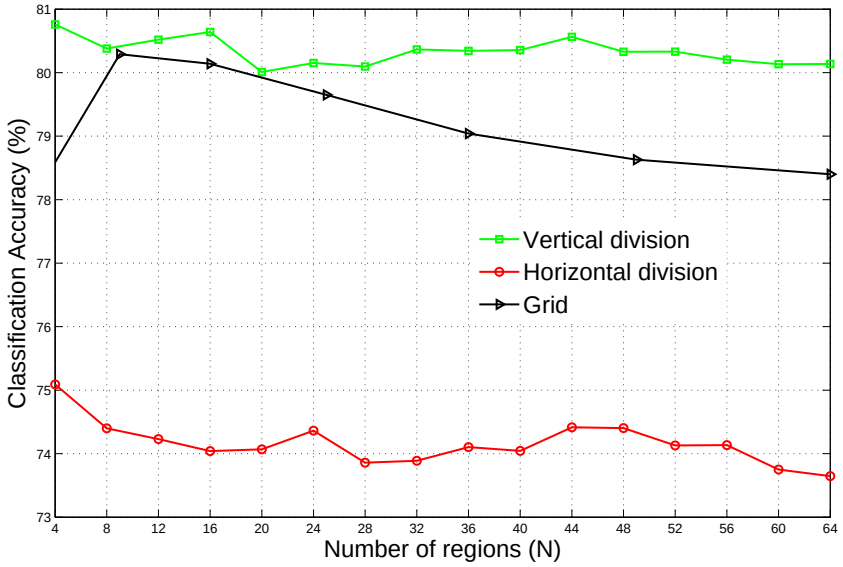


FIGURE 3.3: Classification accuracy (%) versus number of local regions for the 15 Scenes dataset ([Lazebnik et al., 2006]) using pairwise rigid matching and different partitioning schemes: grid, vertical divisions or horizontal ones.

accuracy is plotted with respect to the number of local regions, using either a grid partitioning (i.e. 1×1, 2×2,...,8×8) or divisions along one axis, vertical or horizontal (for each direction: first the image is divided into 1 band, 2 bands or 4 bands; and then tiling each band from 1 to 16 small regions). Each region is described with a SIFT-BoW obtained following the protocol of [Lazebnik et al., 2006] and a vocabulary of 100 words. The classification accuracy was computed with intersection kernel SVM and 10-fold cross-validation.

For the grid division case, we observe that increasing the number of regions (from 4 to 8) first improves the classification accuracy, but when the number of regions is higher, the accuracy decreases. For the vertical division or horizontal division, the accuracy slightly decrease with repeat to the number of regions. It may be explained by the fact that the number of mismatches is all the greater that the number of regions increases. Moreover, using a vertical directional partitioning gives higher results than a grid partitioning for this dataset composed mainly of natural scenes. In the experimental section, we will further study the influence of the scanning direction and the effect of the two parameters number of bands and number of regions.

Scanning

Once the image has been divided into regions, our next task is to represent it as

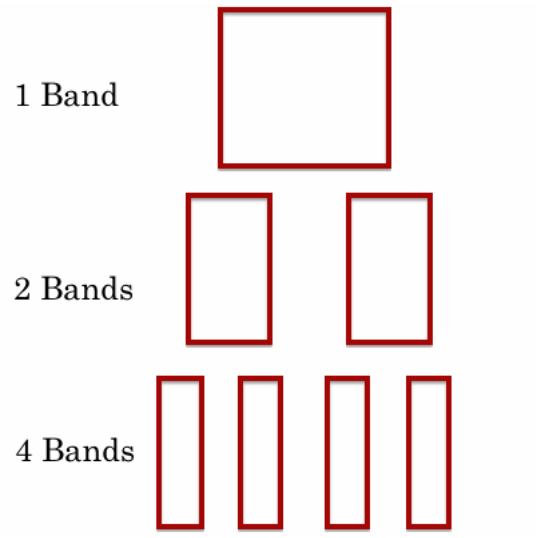


FIGURE 3.4: Pyramid scheme with L=2. It is combination of using 1 Band, 2 Bands and 4 Bands for representing the image.

strings. In the work of [Yeh and Cheng, 2008], they used a row-wise raster scan to construct a string of regions. However, this approach has a major drawback: two successive symbols in the sequence may correspond to non adjacent regions in the image when a new scan line is starting. Even in a zig-zag scan, the spatial relationship type is changing along the string sequence. Therefore, we decide to use not only one but several strings of regions to represent an image. Our proposal is to build a new string for each band of our partition in order to create a set of strings having N symbols each. An image is then described as a set of B strings (for example in Figure 3.2).

Pyramidal strategy

The *pyramid matching* scheme which is proposed by [Grauman and Darrell, 2005; Lazebnik et al., 2006] has shown excellent performance on classification tasks. Motivated by this, we intend to apply multi-resolution regions in our representation. It can be noticed that changing the *number of bands* will change the size of regions but not the length of the strings (i.e. number of symbols). Therefore, we follow the setup of Spatial Pyramid Representation using L levels of resolutions. We keep the *number of regions* fixed and vary the *number of bands* as a power of 2, e.g $B = 2^0, 2^1 \dots 2^L$.

The matching is done between strings in each level separately. Moreover, different weights are applied for different resolution levels as proposed in SPR so that the matching at finer level has stronger effect than the matching at coarser level. For

instance, when we use the pyramid with $L = 2$ levels; level $l = 0$ has a weight $1/2^L = 1/4$, level $l = 1$ has weight $1/2^{L-l} = 1/4$ We also can employ several learning techniques which are proposed to acquire optimal weighting, e.g *Global Level Weights* of [Bosch et al., 2008], *weight map* of [Harada et al., 2011], to have better image classification performance.

3.4 A new edit-distance for strings of histograms

In this section, we present our *String Matching Distance* - a string distance suitable with our local histogram string based representation. This distance is an edit-distance tailored to compensate mismatches limiting performances of rigid matching approaches, as explained previously. We first look back on the standard edit-distance and then introduce the new distance.

3.4.1 The standard edit-distance

The standard edit-distance or *Levenshtein* distance [Levenshtein, 1966] is a distance between two strings of symbols taken from the same alphabet Σ. It is classically used to measure the similarity between two textual words not necessarily having the same size. It is based on three elementary *edit operations*: *insertion, deletion and substitution* of a symbol. A sequence of edit operations transforming string $\mathcal{X}$ into string $\mathcal{Y}$ is called an *edit script*. The edit-distance between two strings $\mathcal{X}$ and $\mathcal{Y}$ is defined as the minimum edit script cost which transforms $\mathcal{X}$ into $\mathcal{Y}$. The permitted edit operations with their associated cost functions are as follows:

- insertion of a symbol y_j of $\mathcal{Y}$ into $\mathcal{X}$ with a cost $c_{ins}(y_j)$
- deletion of a symbol x_i of $\mathcal{X}$ with a cost $c_{del}(x_i)$
- substitution of a symbol x_i of $\mathcal{X}$ with the symbol y_j of $\mathcal{Y}$ with a cost $c_{sub}(x_i, y_j)$. If $x_i = y_j$, no substitution is needed; so $c_{sub}(x_i, x_i) = 0$

In its simple form, it uses a unit cost for all edit operations thus corresponds to the minimum number of operations turning one string into another. For instance, the edit-distance between *abb* and *aa* is 2 since we need two operations in order to convert *abb* to *aa*: substitution of *b* into *a*, and deletion of *b*. In the same way, to

C	λ	a	b
λ	0	2	10
a	2	0	4
b	10	4	0

TABLE 3.1: Example of edit cost matrix C. $\Sigma = (a, b)$, λ is Null histogram.

convert *Sunday* into *Saturday*, we need two insertions a and t and one substitution n to r. So the edit-distance between the two strings *Sunday* and *Saturday* is 3 which is the minimum number of edit operations needed.

More generally, each operation can have a specific non-negative cost. These costs are defined by a *cost matrix* $\mathcal{C}$, for example as in Table 3.1. In this case, the cost for deleting/inserting a symbol a is 2, the cost for deleting/inserting a symbol b is 10 and the cost for substituting a symbol a with a symbol b is 4. The edit-distance between *abb* and *aa* is 10, which is the cost for the deletion of a and two times the substitution of b into a.
It can be noticed that, this time we have used a different scenario when converting *abb* into *aa* because of a different cost matrix. It suggests that, we can learn the cost matrix from the input data to obtain the optimal performance.

Computing this distance can be formulated as an optimization problem and can be carried out with dynamic programming. To compute the edit-distance between string $\mathcal{X}(M) = \{x_1 x_2 \dots x_M\}$ and string $\mathcal{Y}(N) = \{y_1 y_2 \dots y_N\}$, the algorithm consists in computing a distance matrix $D_{(M,N)}$, where $D_{(i,j)}$ represents the minimum cost of transforming the first i symbols of string $\mathcal{X}$ into the first j symbols of string $\mathcal{Y}$. In other words, $D_{(i,j)}$ is the edit-distance between two sub-strings $\mathcal{X}(i)$ and $\mathcal{Y}(j)$, with allowable edit operations mentioned above.

The algorithm 1 describes the dynamic programming based algorithm to compute the standard edit-distance given the cost matrix and the two text strings, $\mathcal{X}$ of length M and string $\mathcal{Y}$ of length N. At each step, the computation of one cell uses only the information of its three previous neighboring cells. For example, to compute cell $D_{(i,j)}$ we need the value of cells $D_{(i-1,j)}$, $D_{(i,j-1)}$ and $D_{(i-1,j-1)}$. Depending on the costs of deletion, insertion and substitution, we can decide which edit operation will be used by choosing $D(i, j)$ as $\min(D_{(i-1,j)} + c_{del}(x_i), D_{(i,j-1)} + c_{ins}(y_j), D_{(i-1,j-1)} + c_{sub}(x_i, y_j))$. After computing the edit-distance, we can also obtain the edit script which is the best way to convert the first string into the second one. Table 3.2 shows an example of distance computation and the corresponding edit script.

		S	**a**	**t**	**u**	**r**	**d**	**a**	**y**
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

TABLE 3.2: Example of computation the distance matrix D between two text strings *Saturday* and *Sunday*.

The computational complexity is proportional to the product of the length of the two strings, *i.e.* $\mathcal{O}(N \times M)$. The value $D_{(M,N)}$ of the last cell of matrix D is the edit-distance between the two strings. It is the minimum cost of all possible edit scripts to convert string $\mathcal{X}$ into $\mathcal{Y}$.

Algorithm 1 Standard Edit-distance Algorithm

Require: Two strings $\mathcal{X}$(M) and $\mathcal{Y}$(N)

Initial:
$D(0,0) \leftarrow 0$
for $i = 1$ *to* M **do**
  $D_{(i,0)} = D_{(i-1,0)} + c_{del}(x_i)$
end for
for $j = 1$ *to* N **do**
  $D_{(0,j)} = D_{(0,j-1)} + c_{ins}(y_j)$
end for
for $i = 1$ *to* M **do**
  for $j = 1$ *to* N **do**
    $D_{(i,j)} = \min(D_{(i-1,j)} + c_{del}(x_i), D_{(i,j-1)} + c_{ins}(y_j), D_{(i-1,j-1)} + c_{sub}(x_i, y_j))$
  end for
end for
return $D_{(M,N)}$

Table 3.2 is an example of using Algorithm 1 to compute the D matrix between the two text strings *Sunday* and *Saturday*. With $c_{del} = c_{ins} = 1$, $c_{sub}(x, y) = 1$ and $c_{sub}(x, x) = 0$.

3.4.2 A new string matching distance

In the previous section, we already presented our string-based representation where each symbol is a histogram of visual words. Our purpose is to use approximate matching between strings in order to correct mismatches of rigid matching. By definition, the edit-distance aims to find the optimal alignment between two strings, and thus allows to correct local or global misalignments due to translation of viewpoint or modifications between two images. The only question is how to adapt this edit-distance to the proposed string representation.

In fact, when computing the edit-distance on the text domain, the alphabet Σ is limited, hence it is possible to define a cost matrix between characters. In our case, each symbol is one finite histogram of visual words. Even with small codebooks, our string alphabet Σ is infinite. It is impossible to define a cost matrix $\mathcal{C}$ for the edit operations. Thus, in order to apply the edit-distance with our string-based image representation, we have to find a new way to compute the cost of edit operations.

A possible method is to setup a pre-defined cost and to choose a distance to compare histograms qualified as ground distance [Yeh and Cheng, 2008; Ballan et al., 2010; Ros et al., 2005]. In order to measure the similarity of two video sequences, [Ballan et al., 2010] proposed to represent them as strings where each symbol is one histogram of keyframe and to use the edit-distance to compare those strings. The authors used an unit cost 1 for all edit operations. To decide which operations to use, a ground distance between two symbols is compared to a threshold to judge whether two frames are similar enough to apply a substitution. If not, a deletion or insertion is employed.In their experiments, they found that the best metric is χ_2 distance with the corresponding threshold of 0.13.
In the work of [Ros et al., 2005], an image is represented as a string of salient regions. They used l_2 distance between two signatures of salient regions to compute the substitution cost. The cost for deletion or insertion of a symbol is the l_2 distance between this symbol signature and a *Null* vector.
Having a similar idea as [Ros et al., 2005], however [Yeh and Cheng, 2008] used χ_2 as ground distance. In this case, the insertion/deletion cost turned to be 1/2 which is the χ_2 distance between any histogram and a Null histogram.

To go further, we propose to adapt insertion and deletion costs to the local context of symbols. Our goal is to virtually adjust the grid partitioning during the image comparison and compensate for mismatches that occur with homogeneous parts

of a scene or object split in different regions. We plan to use deletion and insertion to detect the repetition of regions inside one image. More precisely, during the alignment of the two strings, if one symbol is more similar to its following than to the corresponding one in the other string, it will be removed. Formally, this rule comes to define costs functions as:

$$\begin{cases} c_{sub}(x_i, y_j) & = d(x_i, y_j) \\ c_{del}(x_i) & = d(x_i, x_{i+1}) \\ c_{ins}(y_j) & = d(y_j, y_{j+1}) \end{cases} \tag{3.1}$$

where d is the ground distance. It can be the Euclidean distance, χ_2 distance or any vector distance.

Let us remark that, to ensure the symmetry of the edit distance, we must have $c_{ins}(x) = c_{del}(x)$. Then, for a given symbol x both insertion and deletion operations can be seen as a single edit operation, i.e. the deletion operation but applied either in the first or in the second string. In other words, the insertion of symbol x of string $\mathcal{Y}$ into string $\mathcal{X}$, can be seen as the deletion of symbol x in string $\mathcal{Y}$.

We call our proposed distance *String Matching Distance* (SMD). The string distance $SMD(\mathcal{X}, \mathcal{Y})$ between two strings of histogram $\mathcal{X}$ and $\mathcal{Y}$ can be computed by dynamic programming based on Algorithm 1 and with the edit costs defined in Equation 3.1.

3.4.3 Examples

In this part, we present several examples to illustrate our proposed string-based distance.

Example 1: Let's compute the *String Matching Distance* between 2 text strings *aab* and *abb* with $\Sigma = \{a, b\}$. The edit operation costs are defined as in Equation 3.1; with $d(a, b) = d(b, a) = 1$, $d(a, \lambda) = d(b, \lambda) = 1$ and $d(a, a) = d(b, b) = 0$. The computation of distance matrix D is shown on Table 3.3. In parallel, we also present the edit script matrix, which is same size as D. Each cell of the edit script matrix contains the pointer to point out the cell where its value come from or which the previous edit operation has been applied. Following the script, we can explain step by step the computation of SMD: first, there are two successive identical symbols a in in the first string *aab*, so we delete the first symbol a in

aab. The cost is $c_{del}(a) = d(a, a) = 0$. After that, we need to compute the distance between *ab* and *abb*. Since, there is a symbol a in both two strings, one substitution is enough, which costs $c_{sub}(a, a) = d(a, a) = 0$. After this step, the two new strings are b and bb. There are two identical symbols b in the second string; so a deletion of the first symbol b in the second string is applied, which costs $c_{del}(b) = d(b, b) = 0$. The remaining two new strings b and b are the same, so a substitution is used which costs $c_{sub}(b, b) = 0$. Consequently, the distance between *aab* and *abb* is 0. A basic property of our distance is to remove sequences of identical (or similar) symbols.

		a	**b**	**b**
	0	1	1	2
a	0	0	0	1
a	1	0	0	1
b	2	1	0	0

		a	**b**	**b**
	0	←	←	←
a	↑	↖	←	←
a	↑	↖	←	↖
b	↑	↖	←	↖

TABLE 3.3: Example of computation of SMD between two strings *aab* and *abb*. The first table is the distance matrix D which is computed by dynamic programming. Each value of D, $D_{(i,j)}$ is minimum edit cost to convert sub string $\mathcal{X}(i)$ into $\mathcal{Y}(j)$. The second table is edit scripts matrix, which has same size as D. It is used to point out which previous edit operation has been applied to obtain the value of the corresponding cell in D. We use the notation: ← for a *Deletion* on string $\mathcal{Y}$, ↑ for a *Deletion* on string $\mathcal{X}$ and ↖ for a *Substitution*.

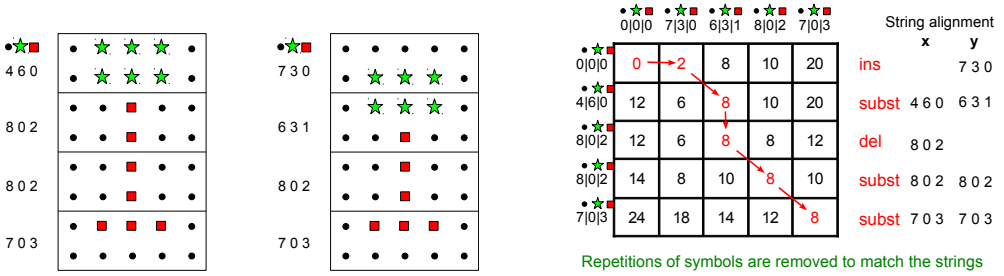


FIGURE 3.5: A toy example to illustrate the matching using SMD for the single band case.

Example 2: We illustrate our SMD on a real histogram-based string with a toy example (Figure 3.5). In the two toy images, the descriptors are computed on 5×8 grids. There visual words (e.g circle, square, star) are utilized to represent the images. Each image is described as one image strings of 1 Band and 4 regions. In the left-side of each toy image, we show the local histograms. We use l_1 as ground distance. All local histograms need to be normalized with the l_1-Norm. However, in this example, all the local histograms have the same l_1-Norm of 10, so no additional normalization is required. Each local histogram is notated as a triplet of three numbers separated by '|'. The SMD distance of two toy images

is the distance between two strings of histogram: $\{4|6|0, 8|0|2, 8|0|2, 7|0|3\}$ and $\{7|3|0, 6|3|1, 8|0|2, 7|0|3\}$. The distance matrix gives the minimum distances $D_{(i,j)}$ and arrows show the sequence with the minimum cost, detailed on the right. To well understand the values, we detail the calculations of three cells. First, cell $D_{(0,1)}$ equal to 2 gives the insertion cost of symbol $7|3|0$, *i.e.* $d(7|3|0, 6|3|1)$, while cell $D_{(1,0)}$ is the deletion cost of symbol $4|6|0$, i.e. $d(4|6|0, 8|0|2)$. The value of $D_{(1,1)}$ is the minimum of $D_{(0,1)} + d(4|6|0, 8|0|2)$, $D_{(1,0)} + d(7|3|0, 6|3|1)$ and $D_{(0,0)} + d(4|6|0, 7|3|0)$, i.e. $min\{14, 14, 6\} = 6$. So $D_{(1,1)} = 6$ and a substitution is needed. As for the computation of $D_{(1,1)}$, each minimum distance takes into account the similarity between neighboring regions and direct pairwise similarity between corresponding regions, allowing to remove repetitions of symbols when necessary to adapt to the other string. In our toy example, the resulting edit sequence comes to consider the two similar regions $8|0|2$ as a unique one that matches the similar one in the second image.

Example 3: A real-case example is given in Figure 3.6. Two images are from *Caltech101* dataset [Fei-Fei et al., 2004; Lazebnik et al., 2006]. We use SIFT descriptor on a densely sampled grid with a patch size of 16×16 and a period of 8 pixels. The vocabulary $K = 100$ words is build by K-means clustering on a random subset of 100000 descriptors taken from all training images. The hard assignment is used in coding step. We use l_1 as ground distance and each histogram is normalized according to l_1-Norm. In this example, the two images are divided into two bands of four regions. We show the local histogram matchings between two first bands of the two images: the SMD matching and the pairwise regions matching (PMD). The string matching script obtained for the first band (x_2 with y_1, x_3 with y_2, x_4 with y_4) shows a better alignment than direct pairwise region matching. Insertions and deletions enable to deal with a change of position of the head of the bird. In both cases of SMD and PMD, we have drawn the final string alignment correspond to the real computed edit scripts. Value of SMD is smaller than of PMD which confirms that the two images are more similar due to better alignment between regions of SMD matching.

These examples prove the interest of our approach to better deal with possible changes in object size, position or shape in the direction of the string.

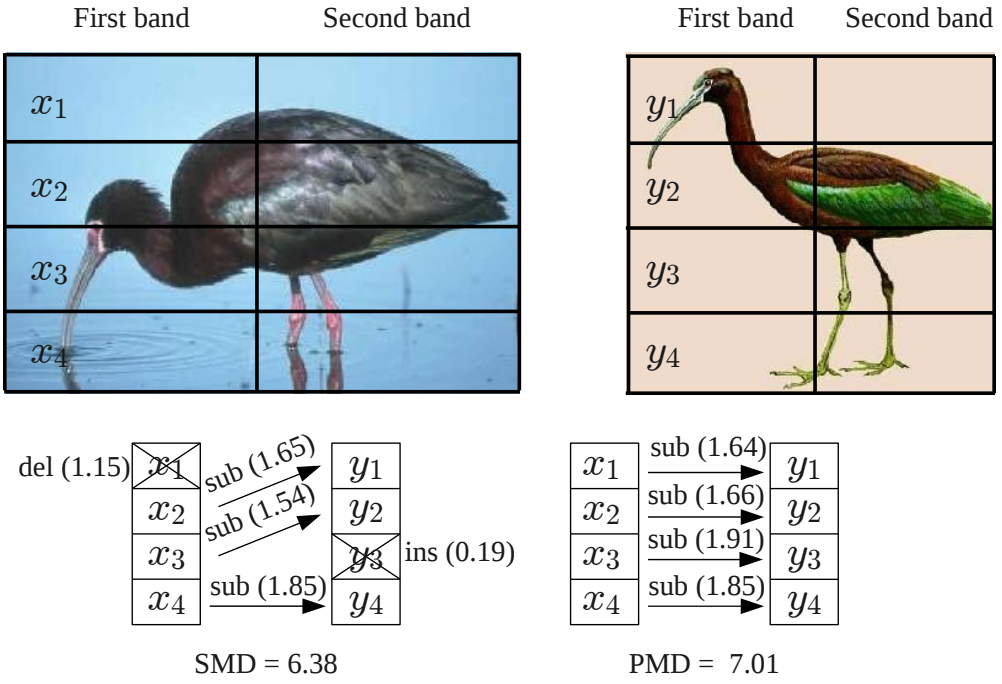


FIGURE 3.6: Real example of SMD matching. Two images are divided into two bands of fours regions. The matching is done on each band separately. Let denote x_i, y_i with $i = 1, \ldots, 4$ are the local histograms of regions in the first bands of the two images. The final edit script (or strings alignment) and the corresponding distance which is the minimum all possible edit scripts are shown on the figure. The SMD distance between two first strings of two images are computed as $SMD = c_{del}(x_1) + c_{sub}(x_2, y_1) + c_{sub}(x_3, y_2) + c_{del}(y_3) + c_{sub}(x_4, y_4) = 1.15 + 1.65 + 1.54 + 0.19 + 1.85$. In parallel, the Pairwise Matching Distance (PMD) which uses rigid matching between local region, is computed as total distance between x_1 and y_1, ...,x_4 with y_4. In other words, it is the total substitution cost between them. $PMD = c_{sub}(x_1, y_1) + c_{sub}(x_2, y_2) + c_{sub}(x_3, y_3) + c_{sub}(x_4, y_4) = 7.01$.

3.4.4 Weighted edit operations

One advantage of our SMD is that edit costs are automatically determined during the string matching. Contrary to other methods, no threshold or fixed cost have to be fixed. However, it is interesting to provide a way to adapt the distance with respect to the dataset. If we try to interpret the computation according to our cost functions and Algorithm 1, we notice that:

- The substitution operation is applied when c_{sub} is small enough compared to c_{del} and c_{ins}. It means that the two regions are very similar. In other words, there is a matching between the two regions in the two images.

- Deletion operations and insertion operations are used when the next neighboring region is very similar to the current one. So we can ignore or delete this region without losing information.

Then, a way to control edit operations is to try to control the balance between substitutions and insertion/deletion operations. Thus, we propose to add a weighting coefficient w controlling the ratio between substitutions and insertion/deletion. The corresponding cost functions are then rewritten as:

$$\begin{cases} c_{sub}(x_i, y_j) & = d(x_i, y_j) \\ c_{del}(x_i) & = w.d(x_i, x_{i+1}) \\ c_{ins}(y_j) & = w.d(y_j, y_{j+1}) \end{cases} \tag{3.2}$$

A high w value penalizes insertion and deletion operations and thus allows more pairwise matchings between two images. In this case, our distance becomes close to rigid matching pairwise distance. Conversely, a low w value enables more insertion and deletion operations which means there is more intra image matching. In the Section 3.5 dedicated to experiments we will study the influence of this parameter.

3.4.5 Image comparison kernel

In recent years, Support Vector Machines (SVM) and related kernel methods have become very popular tools for solving classification problems. For this reason, it is better to define a kernel from our distance. The classical edit-distance kernel proposed by [Li and Jiang, 2005] has the following form:

$$k(\mathcal{X}, \mathcal{Y}) = e^{-\gamma d_{ed}(\mathcal{X}, \mathcal{Y})} \tag{3.3}$$

where $d_{ed}(x, y)$ denotes the edit-distance between two strings $\mathcal{X}$ and $\mathcal{Y}$. $\gamma > 0$ is a coefficient to scale the kernel value for numerical stability. In fact, γ plays a very important role in making the kernel matrix positive definite. Applying with our proposed SMD, Equation 3.3 becomes:

$$K_{SMD}(\mathcal{I}, \mathcal{J}) = e^{-\gamma d_{SMD}(\mathcal{I}, \mathcal{J})} \tag{3.4}$$

where $d_{SMD}(\mathcal{I}, \mathcal{J})$ is the String Matching Distance between two images $\mathcal{I}$ and $\mathcal{J}$. The computation of $d_{SMD}(\mathcal{I}, \mathcal{J})$ depends on the number of bands and number of pyramid levels.

If the images are divided into B bands:

$$d_{SMD}(\mathcal{I}, \mathcal{J}) = \sum_{b=1}^{B} d_{SMD}(\mathcal{I}_b, \mathcal{J}_b) \tag{3.5}$$

where $\mathcal{I}_b$,$\mathcal{J}_b$ are histogram strings of band b of images $\mathcal{I}$ and $\mathcal{J}$.

In case of using a pyramid scheme of L levels:

$$d_{SMD}(\mathcal{I}, \mathcal{J}) = \sum_{l=1}^{L} \sum_{b=1}^{B} d_{SMD}(\mathcal{I}_{b,l}, \mathcal{J}_{b,l}) \tag{3.6}$$

$\mathcal{I}_{b,l}$ denotes the string of level l and band b.

In order to be a valid kernel: this edit kernel must fulfill the Mercer conditions. To do so, for all image pairs $\mathcal{I}$ and $\mathcal{J}$, the Gram matrix $K_{SMD}(\mathcal{I}, \mathcal{J})$ is positive semidefinite. In the experiment part, we have performed some tuning on γ to ensure that the Gram matrix between each random set of training images is positive definite.

3.4.6 Computational complexity

To compute the distance between two image-strings of length N, we need to fill out the $D(N \times N)$ matrix. Each value of this matrix is determined by computing insertion, deletion and substitution costs. These costs are calculated by the ground distance between two histograms of length K (K is the vocabulary size). Therefore, to compute the three edit costs, it requires $3 \times K$ operations. However, the complexity of computation at each iteration is still $O(K)$. Hence, the complexity of computing SMD is $\mathcal{O}(K \times N \times N)$. If the images are represented as B strings of N regions. The similarity between them is done by summing the distance between the corresponding bands of the two images. In other words, we need to compute B times the SMD distances. Therefore, the complexity of comparing two images becomes $\mathcal{O}(B \times K \times N^2)$

3.5 Experiments

This section reports experimental results. The motivation of these experiments is twofold. First we aim to study the influence of the parameters of our image

representation model on the classification accuracy. We validate this part on two popular datasets: 15 Scenes and Caltech 101. Second our goal is to compare the classification performance of our String Matching Distance (SMD) against the rigid Pairwise Matching Distance (PMD) , Spatial Pyramid Representation (SPR) approach and several concurrent methods. This last part is completed with three other datasets: Pascal2007, Graz-01 and Corel10.
We begin this section by describing the datasets used in the experiments. Then we show experimental settings and finally the results.

3.5.1 Datasets

In this work, we use 15 Scenes, Caltech 101, Pascal2007, Graz-01 and Corel10 datasets for experiments.
Caltech101 [Fei-Fei et al., 2004; Lazebnik et al., 2006]: This is a well-known object dataset, consisting of 9144 images from 101 object classes and one background class. The number of images in each category varies from 31 (inline-skate) to 800 (airplanes). This dataset has been used by a lot of researchers in order to evaluate the classification performance of their proposed systems.

15 Scenes: This dataset is a scene dataset, first introduced in [Lazebnik et al., 2006]. The dataset contains 4485 images of 15 classes, from both outdoor scenes *(coast, suburb, forest, mountain, open country, street, highway, tall building, inside city, industrial)* and indoor scenes *(bedroom, store, living room, kitchen, office)*. The number of images in each category varies from 200 to 400 images. Examples about this dataset are shown on Figure 3.7.

Pascal2007: [Everingham et al., 2007] The dataset consists of 9963 images from 20 different object classes. Those object classes are categorized as person (*person*), vehicle (*aeroplane, bicycle, boat, bus, car, motorbike, train*), animal (*bird, cat, cow, dog, horse, sheep*) and indoor objects (*bottle, chair, dinning-table, potted-plant, sofa, tv/monitor*). It is an extremely challenging dataset since the images contain objects of different scales, view points, illuminations and poses. Moreover, an image may belong to more than one class. Some example images are shown in Figure 3.8.

Corel10: This dataset is used in [Lu and Ip, 2009] with 10 selected classes: *skiing, beach, buildings, tigers, owls, elephants, flowers, horses, mountains and food* from the Corel dataset. Each class has 100 images which have sizes equal to 384×256 or 256×384. Examples of this dataset is shown on Figure 3.9

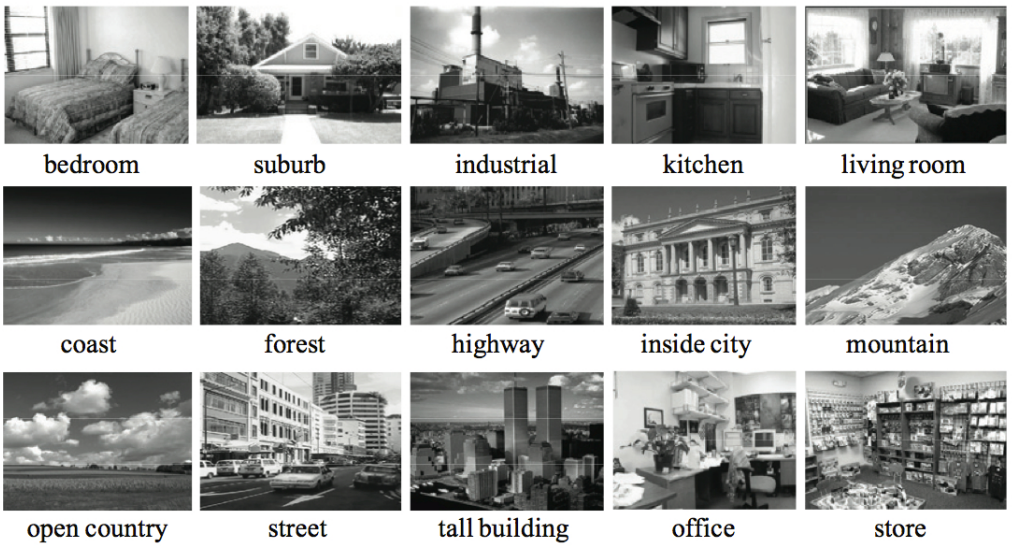


FIGURE 3.7: Examples from 15 scene dataset.

FIGURE 3.8: Examples from Pascal 2007 dataset.

Graz-01: This dataset is a small dataset, which contains 1103 images of only 3 classes: *People*(460 images), *Bike*(373 images) and *Background* (270 images). However, it is also a very challenging dataset due to high intra- and inter- class variations and objects with different scales or poses.

FIGURE 3.9: Examples of Corel10 dataset [Lu and Ip, 2009].

3.5.2 Experimental Settings

For all datasets, we use dense sampling SIFT descriptor calculated on 16×16 image blocks which overlap every 8 pixels. To create the vocabulary, the K-means clustering is applied on a subset of the descriptors. For Caltech 101 and 15 Scene, three codebook sizes are chosen: $K = 100$, 200 and 400. For other datasets, only $K = 100$ vocabulary is used. In the coding step, both hard assignment with sum pooling and sparse coding with max pooling are employed. However, only hard coding is used in the first part in order to evaluate the impact of the string parameters.

For classification, we apply a SVM classifier using the *libSVM* toolbox [Chang and Lin, 2011]. We first calculate the distances and then compute the corresponding kernel matrices using the formulas of edit kernels presented in Section 3.4.5. We do a 10-fold cross validations on random train/test subsets. For fair comparison with other approaches, we keep the same train/test setup as reported in the original papers. For Caltech 101, 15 Scene and Graz 01 dataset, we follow the setting of [Lazebnik et al., 2006]. That is:

- With Caltech 101: we randomly select 30 images per class for training and maximum 50 images per class for testing.
- With 15 Scene: we randomly select 100 images per class for training and test on the rest.

- With Graz-01: we train on 100 positive images (bike or person) and 100 negative images (where 50 images is from background, 50 images from the other class).

For Pascal 2007 dataset, we keep the training/test sets given in VOC2007 challenge [Everingham et al., 2007]: we train on 5011 images and test on 4952 images. We also use the development kit [1] of VOC-challenge in order to evaluate the classification performance.
For Corel10 dataset, following [Lu and Ip, 2009], we randomly select 50 images per class for training and the rest images are used for testing.

An other problem is the tuning of SVM parameters γ and c. Note that, the value of γ is chosen to ensure the kernel matrix (i.e Gram matrix) to be semidefinite positive and in the mean time, to obtain the best classification performance. The tuning is done on a subset of training images. We divide the training set into five parts. Each time, we train on one part and test on the rest. We apply a grid search to find the best value of γ. We compute the Gram matrix with the so-obtained value of γ, and verify that all of its *eigenvalues* are positive. If yes, we keep this γ for classification. If no, we redo the previous work. This way provides the best γ but requires computation time. There is another possible way, which is to set γ to the inverse of the pairwise mean distances as in [Avila et al., 2013]. Since we already have the matrix of distances between all pairs of training images, it is easy to calculate γ according to this approach. Following this procedure generally leads to a near optimal classification performance.
We also varied the values of c from 1 to 10, and found that values of c equal 3, 4 or 5 almost led to the same best performance. As a result, we fixed $c = 4$ for further computations.

3.5.3 Results

3.5.3.1 Influence of the string parameters

In our string based representation model, several parameters have to be set to compute classification results: the ground distance, the scanning direction, the number of bands B, the number of regions N and the codebook size K, the weighting parameter for edit operation w and the number of pyramid levels L.

[1] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/devkit

In this section, we study the influence of these parameters. The experiments are validated on two datasets: 15 Scenes and Caltech 101.

Choosing the ground distance:

We study the effect of the ground distance on SMD classification performance. Number of bands and codebook size are fixed respectively to $B = 1$ and $K = 100$. The l_1, l_2 and χ_2 are considered as ground distances. The classification accuracies are shown on Figure 3.10, where the accuracies obtained by l_1 and χ_2 distances are nearly equivalent; and are much higher than the results obtained with l_2. Compared with χ_2, l_1 can be computed more simply. Therefore, we chose l_1 as the ground distance and used it for all following experiments.

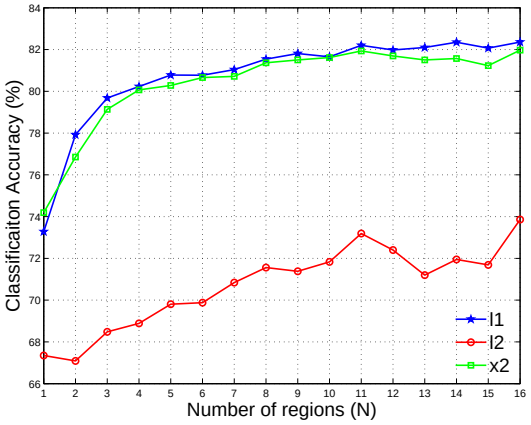


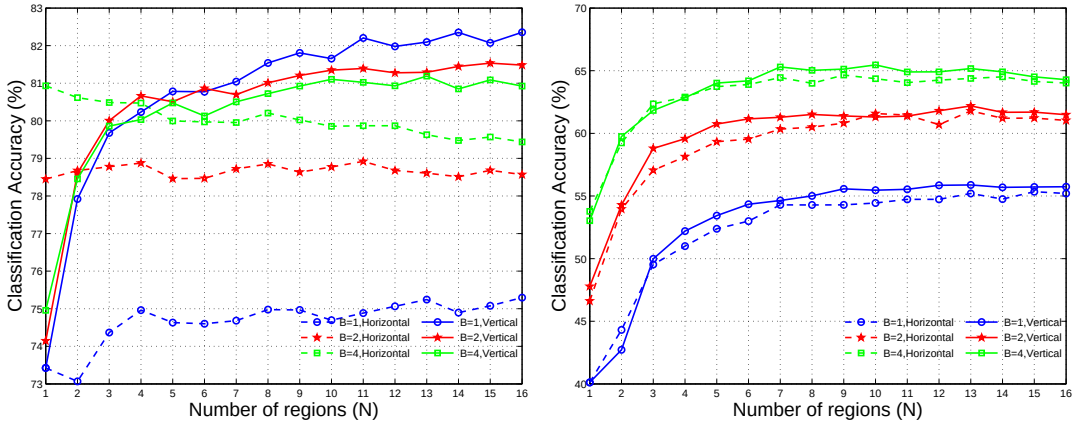
FIGURE 3.10: Classification accuracy (%) obtained by SMD distance with different choices of the ground distance (i.e. l_1, l_2, χ_2) on 15 Scene dataset. The number of regions N is varied from 1 to 16. Number of bands and codebook size are fixed to $B = 1$ and $K = 100$.

Scanning direction:

The objective of this experiment is to examine the impact of the scanning direction (*i.e.* vertical or horizontal) on SMD implementation. The tests are evaluated on 15 Scene and Caltech 101 datasets with respect to the number of regions N varying from 1 to 16. The number of bands is set to $B = 1$, $B = 2$ and $B = 4$. We fix the codebook size to $K = 100$ words. The classification accuracy for the two datasets are shown on Figure 3.11.

For the two datasets, the obtained results showed different trends but the vertical direction normally provides a better characterization of the image structure than the horizontal one. For the 15 Scenes dataset, all vertical-cases graphs are above horizontal cases ones. For the Caltech 101 dataset, the differences between the

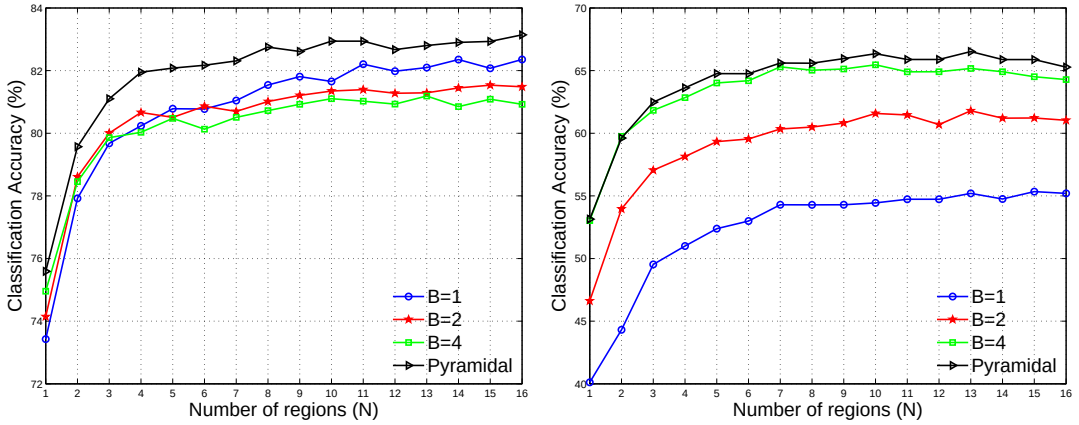
horizontal and the vertical cases are not that significant. It can be explained by the specificity of this collection where the objects of interest take up most of the image and are approximately centered. Comparing objects along one or the other direction does not really matter. Since on the overall the vertical direction performs better, we keep this direction in all subsequent experiments.



(a) 15 Scene (b) Caltech101

FIGURE 3.11: Classification accuracy (%) obtained by SMD distance on 15 Scene and Caltech 101 datasets using vertical scanning direction (solid lines) and horizontal scanning direction (dash lines). The number of bands B is varied in 1,2 or 4. The codebook size K is fixed to 100 words.

Number of bands and pyramidal strategy:



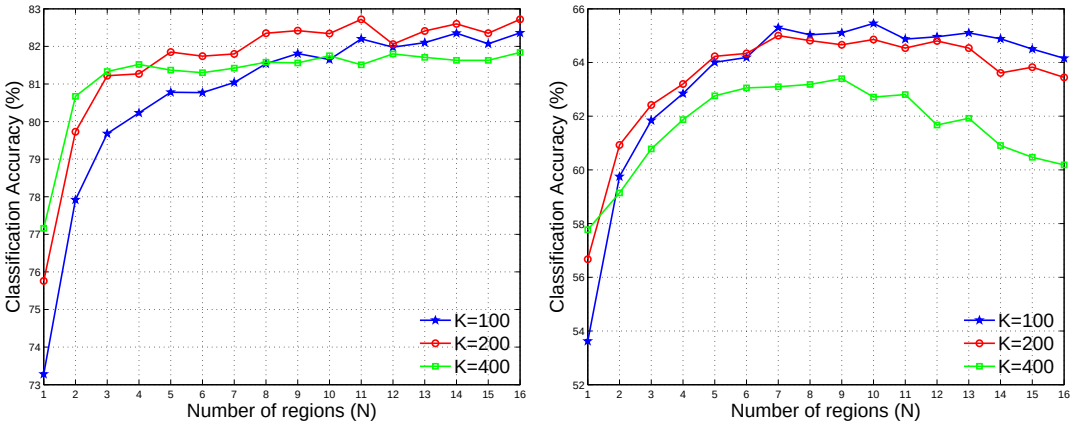
(a) 15 Scene (b) Caltech101

FIGURE 3.12: Classification accuracy (%) obtained by SMD with variations of number of bands B and with pyramidal strategy. The codebook size is $K = 100$ for both datasets.

The purposes of this experiment are to investigate influence of the parameter (B)-number of bands and to prove advantages of using pyramid scheme into SMD classification performance. We keep the same setup as previous tests: $K = 100$, $N = 1, \ldots, 16$, and we scan the image vertically to create the strings. The number of bands B is set to 1,2 and 4. We also evaluate the pyramidal strategy with $L = 2$, which means the combination of one band, two bands and four bands. Figure 3.12 indicates the classification performance of SMD on the two datasets. Again, the results depend on the collection. Indeed, considering the 1, 2 and 4 bands cases, the results behave inversely. For 15 Scenes, results decrease as the number of bands increases, *i.e.* one band is enough to get the best result, while for Caltech 101, it is preferable to use four bands. As previously, it is inherent to the type of images. Observing a natural scene from the top to the bottom allows to identify the content. Using two parallel vertical bands does not convey much information. It even introduces confusion because of redundancy between bands, leading to worst results. For objects, a finer look at the different parts is necessary to identify them correctly.
It is worth noticing that for both datasets, a 2-level pyramid approach clearly outperforms single level split cases. This strategy is suitable to get the best results.

Vocabulary size:

To investigate the effect of the vocabulary size, we fix the number of bands to the optimal values obtained previously, *i.e.* $B = 4$ for Caltech 101 and $B = 1$ for 15 Scenes. Figure 3.13 shows the classification results for three vocabulary sizes 100, 200 and 400 as a function of N. For Caltech 101, the best results are obtained with the smallest vocabulary ($K = 100$) and we note that the accuracy is decreasing over N for $K = 200$ or $K = 400$. For 15 Scenes, the influence of the vocabulary size is low and the results are slightly better for $K = 200$, but they are very close to $K = 100$ for a large N.
Obtaining the best results for small vocabularies is unusual in the BoW context. We think that this effect is due to a too low number of descriptors. Histograms are not computed accurately when the size of the vocabulary increases and as the number of regions grows. However, with small vocabularies, increasing the number of regions increases significantly the accuracy. This is clear advantage of our string-based approach to provide high accuracy with compact vocabularies and the compact representation.



(a) 15 Scene (b) Caltech101

FIGURE 3.13: Classification accuracy (%) obtained by SMD distance on 15 Scene and Caltech 101 datasets using different vocabulary sizes (*i.e.* 100, 200, 400). The number of bands B is fixed $B = 1$ for 15 Scene and $B = 4$ for Caltech 101.

Number of regions:

The effect of number of regions N on the distance implementation can be seen in Figures 3.10, 3.11, 3.12 and 3.13. Considering the influence of the number of regions N, the global evolution of all curves is similar: the accuracy is almost monotonically increasing with a stabilization for $N = 8$ for Caltech 101 and $N = 10$ for 15 Scenes. From this value, the results remain roughly constant or slightly better. The highest accuracy is 66.53% achieved with $N = 13$ for Caltech 101, and 83.14% with $N = 16$ for 15 Scenes. Since performances were quite similar with lower number of regions, it is preferable to use $N = 9$ to $N = 12$ to reduce the computation time.

Weight parameter:

In the section 3.4.4, we have already discussed about weighted edit operations. In this experiment, we aim to estimate the influence of the weight parameter and to find its optimal value. The weight is varied from 0.5 to 1.3. SMD is computed with $K = 100$ for both datasets. With the 15 Scene dataset, we use $B = 1$ and $N = 16$. In case of Caltech 101 dataset, B is fixed to 4 and $N = 10$ for saving computation time. The results are shown on Figure 3.14. It can be noticed that: the weight parameter has effect on the performance of SMD, and for both 15 Scenes and Caltech 101 datasets, the best weighting value is around 0.8. It means our proposed distance performs better with more substitutions.

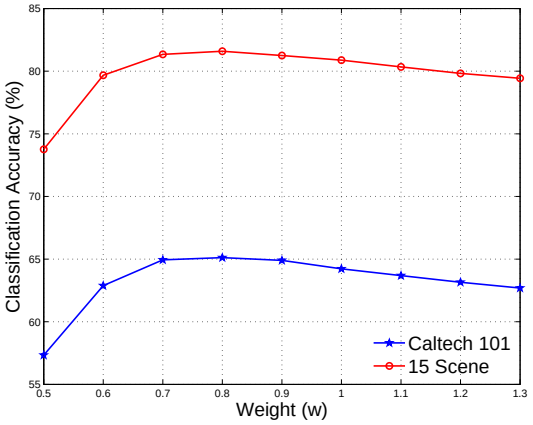
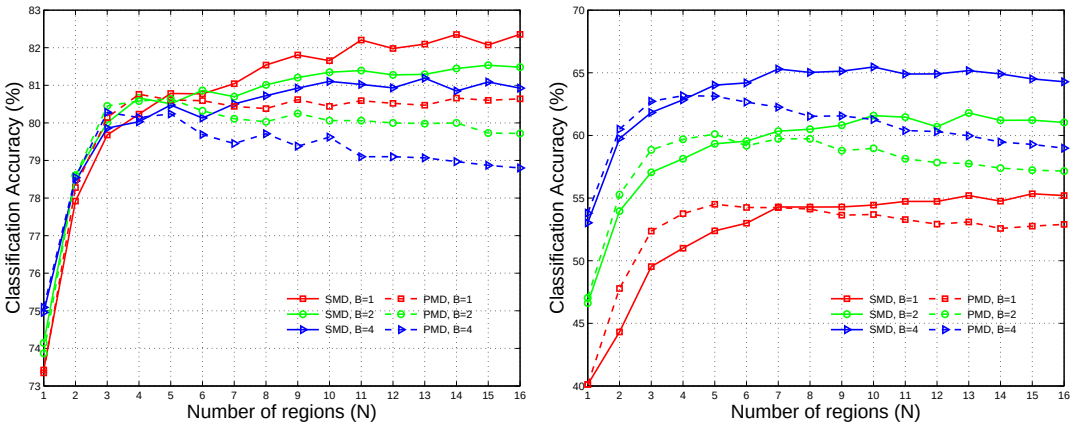


FIGURE 3.14: Influence of weighting parameter on classification performance of SMD. We fix $B = 1$ and $N = 16$ for 15 Scene dataset and $B = 4$ and $N = 10$ for Caltech 101. The codebook size is fixed to $K = 100$ for both two datasets.

3.5.3.2 String matching vs pairwise matching



(a) 15 Scene (b) Caltech 101

FIGURE 3.15: Comparing the classification performance of SMD (solid line) versus PMD (dash line) with different values of B. The codebook size is fixed $K = 100$ for two datasets.

These experiments are to verify the performance improvement of our string matching approach (SMD) over a rigid Pairwise Matching Distance (PMD). Figure 3.15 presents the results. First, it is obvious that for both datasets, SMD is always better than PMD from $N > 5$ and any given splitting. As seen in Section 3.3, the greater the number of regions, the greater the number of local mismatches, leading to a decrease of performance of a pairwise matching approach. With SMD, for large N, the accuracy stabilizes (Caltech 101) or slightly increases (15 Scenes). It proves that SMD naturally compensates local mismatches.

3.5.3.3 Comparison with existing methods

Results on 15 Scene dataset:

In Table 3.4, the proposed method is first compared with the concurrent techniques that use the same setup which is: a single SIFT descriptor, dense sampling on 16×16 image blocks and at every 8 pixels. The results of SMD performance on both hard coding (with sum pooling) and sparse coding (with max pooling) are reported. In both cases, the SMD is computed with pyramid scheme $L = 2$, number of regions $N = 16$, codebook size $K = 100$ and weight $w = 0.8$.
We first compare our method with the BoW baseline (K=100, intersection kernel) and with the original Spatial Pyramid Representation (pyramid with $L = 2$) of [Lazebnik et al., 2006].

Coding	Approaches	Accuracy (%)
Hard-coding	BoW baseline	73.28 [100]
	SPR [Lazebnik et al., 2006]	81.40 ± 0.50 [200]
	Sequence string matching [Yeh and Cheng, 2008]	80.93 ± 0.64 [200]
	Bipartite graph matching [Belongie et al., 2002]	78.17 ± 0.47 [200]
	SPR+co-occurence [Yang and Newsam, 2011]	82.51 ± 0.43 [200]
	Learning optimal spatial partition [Sharma et al., 2011]	80.10 ± 0.60 [100]
	SMD	**83.14 ± 0.67 [100]**
Sparse-coding	BoW baseline	63.48 ± 0.79 [100]
	ScSPM [Yang et al., 2009]	80.28 ± 0.93 [1024]
	ScSPM [Boureau et al., 2010]	83.10 ± 0.60 [1024]
	KSR-SPM [Gao et al., 2010]	83.68 ± 0.61 [1024]
	SMD	**84.59 ± 0.7 [100]**

TABLE 3.4: Classification accuracy on the 15 Scene dataset using different approaches and coding methods. Results of SMD are obtained with codebook size $K = 100$, pyramid scheme $L = 2$, $N = 16$ and edit operation weight $w = 0.8$ for both hard coding and sparse coding. For other methods, the results are obtained with the codebook size given in brackets.

We are also interested in the sequence string matching proposed by [Yeh and Cheng, 2008]. They used a raster scan to represent the image as one string, and a fixed cost for deletion/insertion operations. Furthermore, we compare our matching with the bipartite graph matching of [Belongie et al., 2002] which finds the

best alignment (minimizing the cost matching) between all pairs of regions of two images but subject to the constraint that the matching is one-to-one (this result is from [Yeh and Cheng, 2008]).
In addition, we report the results of two other approaches which successfully incorporate spatial information into BoW: a combination of SPR with information about the co-occurence visual words proposed by [Yang and Newsam, 2011] and an adaptive spatial partitioning proposed by [Sharma et al., 2011] which try to learn the best way to divide images into sub-regions. All the approaches above are computed with hard coding.
The second part of the Table 3.4 shows the results on sparse coding. Our results is compared to Sc-SPM and *Kernel Sparse Representation* (KSR-SPM) methods. The Sc-SPM approach is a Spatial Pyramid Representation using sparse coding. We show on here the results of Sc-SPM performed by both [Yang et al., 2009] and [Boureau et al., 2010]. The KSR-SPM approach [Gao et al., 2010] is the combination of SPR with a sparse kernel representation technique. The results in Table 3.4 shows that our method definitely outperforms all other approaches for both hard coding and sparse coding. It is important to note that the best result is obtained with the smallest vocabulary of 100 words.

Results of Caltech 101 dataset

The classification performance of SMD on Caltech 101 dataset with both hard coding and sparse coding are shown on Table 3.5. Again, we compare our approach with SPR of [Lazebnik et al., 2006] for both hard coding and sparse coding. In addition, we pay attention on SVM-KNN of [Zhang et al., 2006]. In this method, authors used a hybrid classifier which is a combination of SVM and KNN to improve the final classification performance. For each test image, first the KNN classifier applies to find the K-closest images from the training set. If all these K images have the same label, this label is assigned to the test image. If not, then a SVM classifier is applied but only on these K training images. Moreover, in case of using hard coding, we also compare our method with SPR+ Global Level Weight (GLW) optimization [Bosch et al., 2008]. In this method, the weight for each level pyramid is learned to get the optimal classification accuracy. Let us notice that this approach used multi-scale dense SIFT descriptor compared to our single scale dense SIFT descriptor.

From the Table 3.5, we see clearly that our method is compatible with SPR+GLW but we use fixed weights for the pyramid levels. Also, in our method, we use only single scale for SIFT, with smaller vocabulary. In comparison with other

approaches, our method outperforms all of them for both hard coding and sparse coding. The best performance is obtained when using SMD with sparse coding and the average classification accuracy is 73.48%.

Coding	Approaches	Accuracy (%)
Hard-coding	BoW baseline	40.12 ± 0.5 [100]
	SPR [Lazebnik et al., 2006]	64.40 ± 0.8 [200]
	SVM-KNN [Zhang et al., 2006]	66.23 ± 0.48 [-]
	SPR+GLW [Bosch et al., 2008]	66.5 ± 0.70 [1500]
	SMD	**66.52 ± 0.51 [100]**
Sparse-coding	BoW baseline	55.48 ± 1.30 [100]
	ScSPM [Yang et al., 2009]	73.20 ± 0.50 [1024]
	ScSPM [Boureau et al., 2010]	71.50 ± 1.1 [1024]
	SMD	**73.48 ± 1.2 [100]**

TABLE 3.5: Classification accuracy on Caltech 101 dataset using different approaches and coding methods. The result of SMD are obtained with codebook size $K = 100$, pyramid scheme $L = 2$, $N = 13$ and edit operation weight $w = 0.8$ for both hard coding and sparse coding. For other methods, the results are obtained with the codebook size (if available) given in bracket.

Results on Graz-01 dataset

We also applied our approach to Graz-01dataset. For evaluation, we keep the same setup as [Lazebnik et al., 2006] with dense-SIFT and hard-assignment coding. We chose to train on 100 positive images (bike or person) and 100 negative images (where 50 images are from the background class) and test on the rest. We reports the average of equal error rates over ten runs. As can be seen from the Table 3.6, the results are very impressive: our SMD outperforms SPR by 6%, and the combination of SPR+co-occurence of visual words [Yang and Newsam, 2011] by about 1 % for both two classes.

Class	SPR	SPR+ co-cocurrence	**SMD**
Bike	86.5 ± 2.5	91.0 ± 4.8	**92.1 ±2.2**
Person	82.3 ± 3.1	87.2 ± 3.8	**88.2 ±3.5**

TABLE 3.6: The comparison about classification performance (Average of equal error rate) of our proposed with SPR and SPR+co-occurence on Graz-01 dataset. All methods use hard coding

Results on Corel10 dataset

Our purpose of this experiment is to compare our string-based distance with Spatial Mismatch Kernel of [Lu and Ip, 2009]. We keep the same setup as the authors, that is we randomly select 50 images as training data and use the rest as test data. We show in the Table 3.7, the results obtained with SVM one vs all [2] and 10 fold cross validation on hard coding. Moreover, our results with hard coding are shown compatible with the results of KSR-SPR of [Gao et al., 2010] using sparse coding. Our method outperforms *Spatial Mismatch kernel* by about 6.4 % and SPR in both cases of using hard coding (3.1 %) and sparse coding (2.9 %). We believe that, the performance of SMD can be improved by using sparse coding (as seen with Caltech 101 and 15 Scene datasets.)

Coding	Approaches	Accuracy (%)
Hard-coding	BoW baseline	80.20 ± 3.3
	SPR [Lazebnik et al., 2006]	85.98 ± 1.4
	Spatial Mismatch Kernel [Lu and Ip, 2009]	82.7 ± 3.0
	SMD	**89.1** ± 0.88
Sparse-coding	Sc-SPR	86.20 ± 1.01
	KSR-SPR [Gao et al., 2010]	**89.43 ± 1.27**

TABLE 3.7: Classification accuracy on Corel10 dataset using different approaches and coding methods. The result of SMD are obtained with hardcoding, codebook size $K = 100$, $B = 1$, $N = 9$ and weight $w = 0.8$.

Results on Pascal2007 dataset

With Pascal 2007 dataset, we report the Average Precision (AP) values.We compare our result with SPR approach which is computed by ourself. However, for fair comparison, we do not report the results of other approaches since we use a very naive configuration, that is: single SIFT descriptor on single scale (16x16 pixels, overlap 8); small codebook size $K = 100$ and only edit kernel with SVM. Almost all other approaches use multi-scale descriptors with multiple image features (texture, color, shape), with multiple kernels learning, and typically the codebook size is larger, about 1000 words in order to obtain an optimal performance.

[2] [Lu and Ip, 2009] reported results with SVM one vs one

Method	Aero	Bicyc	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
SPR	58.16	30.55	29.14	51.10	20.79	39.36	63.35	29.83	41.25	23.41
SMD	**60.33**	**46.99**	**29.52**	**58.98**	16.39	**52.84**	**68.68**	**39.59**	**48.05**	**33.73**

Method	Table	Dog	Horse	Mbike	Person	Plant	Sheep	Sofa	Train	TV	**Mean**
SPR	36.80	31.72	65.96	41.73	72.44	18.12	26.85	33.39	55.01	29.04	39.90
SMD	**44.11**	**35.51**	**70.60**	**50.98**	**75.79**	17.56	**32.51**	**42.46**	**67.72**	**39.95**	**46.62**

TABLE 3.8: Image classification results (AP) on Pascal 2007 dataset. The codebook size is fixed to $K = 100$ for all approaches. The results of SMD is obtained with $N = 16$; pyramid with $L = 2$ and $w = 0.8$.

Our approach has very successful performance compared with the standard SPR. SMD shows better classification for 18/20 classes. With several classes, the difference is significant such as *Bicycle* 16.5 %, *Bus* 13.5 %, *Cow* 10.3%, *Sofa* 9%, *Train* 12.7 % and *TV* 11 %. In average, our SMD is superior to SPR by about (6.7 %).

3.6 Conclusion

In this chapter, we first described a novel image representation as strings of histograms which encode spatial information, each histogram being a BoW model of a subregion. Then, we introduced a new edit-distance able to automatically identify local alignments between subregions and sequences of similar subregions. This characteristic makes our method more robust to translation or scale variations of objects in images than SPR-based approaches that compare rigidly corresponding parts of images. The experiments confirm that our model is able to take into account spatial relationships between local BoW and leads to a clear improvement of performance in the context of scene and image classification compared to the classical spatial pyramid representation. It is worth noticing that to the best of our knowledge, it is the first time that results better than SPR are reported with the standard BoW coding and a lower dimension for the representation. Moreover, the proposed approach obtain similar or better accuracies than other recent methods trying to infuse spatial relationships into the original BoW model with the great advantage of using a small codebook and a compact representation.

Chapter 4

Merge-based edit-distance for strings of histograms

Abstract: *The previous chapter described the String Matching Distance which uses string-based representation and string matching to compare two images. In this chapter, we propose two extensions of SMD to further improve the recognition performance. Our motivation is to study other possible options of extended edit-distance and to find better ways to compare our image string of histograms. We extend the edit-distance by using merge operations. The performance of these extended distances are evaluated on 15 Scene, Caltech 101 and Pascal 2007 datasets.*

4.1 Motivation

The previous chapter has focused on the String Matching Distance algorithm which uses string-based representation and string matching approaches to measure image similarity. In SMD algorithm, two new definitions of deletion and insertion operations were provided. In main idea of the approach is to delete a region when its content is almost identical to its next neighboring one.The deletion/insertion of a region is therefore applied when the content of this region is more similar to its successive region than the corresponding one in the other image.

Figure 4.1(a) illustrates this procedure: the two images are divided into 4 regions. In the first image, the first region (sky) has similar visual content to the second region, then it is deleted. The second one also looks like the third one therefore it is removed. In the second image, the first region has the same visual content

as the second region thus it can be ignored. Likewise, the third region is removed because it is very similar to the fourth region. Finally, the image matching results associating in the third region of the first image and the second region of the second image, as well as the fourth region of the first image and the fourth region of the second one.

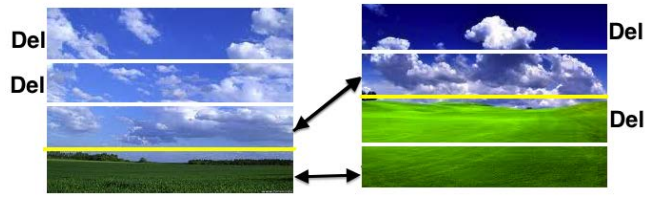
These definitions have the advantage to detect and set aside identical regions inside an image. Hence, the matching between regions of two images is more robust than the rigid matching. However, it also introduces several disadvantages. If a deletion/insertion of a region is applied, this region is totally ignored in all future computations of edit distance and only its next region is kept in the string for remaining calculations. Because two consecutive regions may not be completely identical, using the second region instead of the first one in future computation of the distance is arbitrary. Moreover, ignoring the deleted region leads to loss of information.

To deal with these problems, we think it is better to keep both regions and use their information for future computation. Inspired by this motivation, we replace the deletion operation by a new edit operation, *merge operation*, which combines two or more regions inside an image to create a new one.

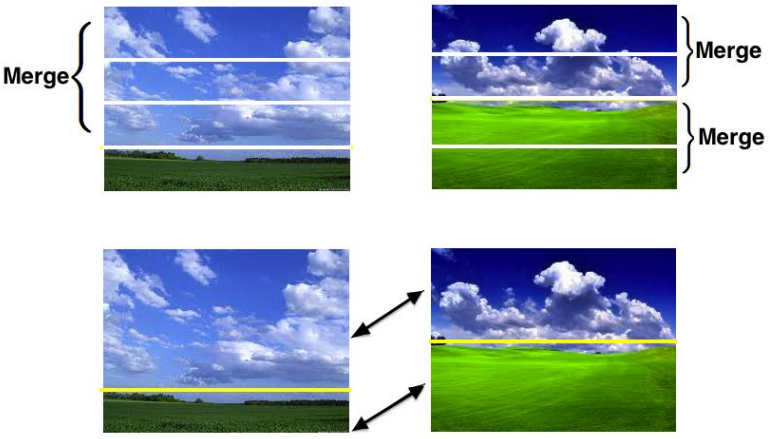
An example of this merge operation is demonstrated in Figure 4.1(b). Here, the three first regions of the first image are combined to create a single one due to the similar visual content. In the same way, the first and second regions, the third and the fourth regions of the second image are grouped together. The matching of two images is then between these combined-regions. It is more precise approach compared to SMD (Figure 4.1(a)) because it keeps all the image content information.

4.2 Related work

The merge operation has been introduced with string distance on several applications. [Khurshid et al., 2009] introduced the merge-split operations in order to detect/recognize words in scanned document retrieval applications. Typically, the low quality scanning of documents may introduce some noise which causes difficulty to detect or to recognize the characters. For this reason, the author proposed the merge operation which allows the combination of two symbols to make a new one, and the split operation which allows a symbol to be divided into two symbols.



(a) Matching of regions in SMD algorithm



(b) Matching of regions with new merge operation

FIGURE 4.1: Example about SMD-matching and merge region based matching.

[Christodoulakis and Brey, 2009] also introduced combination-split edit-distance in OCR pattern matching. In order to cope with *inexactness* pattern matching, the authors introduced two more edit operations: the combination operation that combines two or more symbols from the first string to match with a single symbol from the second string. Equivalently, a single symbol of one string can be split and matched with a sequence of symbols in the other. The decision of which combination will match to which symbol depends on applications. For example in this paper, the authors group a sequence of symbols that look similar to one symbol and treat it as this symbol (likes $rn = m$, $ii = n$, ...).

The merge operation was also introduced in computer vision applications. [Tsai and Yu, 1985] presented a merge operation with *Attribute String Distance* for Shape recognition task. A new operation can be used to combine a number of consecutive boundary primitives in one shape and to match with those in another shape.

In applications of content based video retrieval, [Adjeroh et al., 1999] proposed *v-string* to measure similarity of two video sequences. The distance between two video sequences is described as a problem of sequence to sequence matching. Since

videos always contain temporal information (such as order of frames), repetitions, several video edit operations (such as special effect transitions), video functionalities (such as fast-forward, slow motion, frame skipping) edit-distance with only three basic edit operations (insertion, deletion, substitution) is not robust enough to compute distance between two video sequences. The authors then introduced the v-string edit-distance with merge operation (i.e combination of two or several similar frames to a single frame) and fission operation (i.e separation of one frame into two or more similar frames). These two operations are performed offline before computing the video sequence distance.

In our case, we define the merge operation as a combination of two or more similar regions from one image. With the new operation, we propose a *merge-based* string matching approach, called *m-SMD*, which allows regions to be grouped together, separately in each image, depending on the image visual content. This distance automatically identifies local alignments between sub-regions or groups of similar sub-regions in the images. With the proposed operation, the number of sub-regions for different images may vary and is adjusted according to the visual content, which brings more flexibility to the matching process.
Compared to *partition learning* approach ([Sharma et al., 2011]) and *randomized partition* ([Jiang et al., 2012]) our approach presents the advantage to get a partition grid for each pair of images in order to have a better matching. Also, no learning process is needed. The image division grid is obtained along with the distance computation.

The rest of this chapter is organized as follows: Section 4.3 explains how to compute the new edit-distance with the merge operation. We introduce two frameworks to compute the merge-based approach in Section 4.4: a greedy m-SMD algorithm and a recursive m-SMD algorithm in the sub sections 4.4.1 and 4.4.2. Section 4.5 describes the experiments and discusses about results obtained with scene and object datasets. Finally, we sum up the chapter in Section 4.6.

4.3 Adding new merge operation into SMD

The underlying idea of this section is to define a new edit-distance, *m-SMD*, which supports a merge operation between regions and which is adapted to our string of local histograms image representation.

4.3.1 Principle

The principle of m-SMD is to allow regions to be merged depending of their visual content. When two regions are merged, the resulting histogram is the combination of the two initial ones. Then, depending on merge operations made within each string, one single region of one image may be matched to a set of merged regions in the other image, resulting in a better matching between the two images.

In order to determine which regions to merge, we formulate it as a problem of deleting a region in string matching. For example, two strings $\mathcal{X} = \{x_1, x_2, \dots x_M\}$ and $\mathcal{Y} = \{y_1, y_2, \dots y_N\}$ have M and N symbols. To delete symbol x_i, we combine this symbol with its next symbol x_{i+1}. The cost of this operation is computed as the ground distance $d(x_i, x_{i+1})$ between the two histograms symbols . One new symbol is then generated, as the combination of the two symbols x_i and x_{i+1}. This new symbol is denoted as $\bar{x}_{i \to i+1}$. So $\bar{x}_{i \to i+1} = \mathit{merge}(x_i, x_{i+1})$.

With this definition, there is neither deletion nor insertion operation in the new distance. However, we can also remark that the merge operation of symbols x_i and x_{i+1} can be seen as the deletion of symbol x_i followed by a modification of symbol x_{i+1} which is assigned to the result of the merge operation. Then, two successive symbols x_i and x_{i+1} are likely to be merged if their distance is lower than the distance between the first symbol x_i and the symbol y_j in the other string. In summary, the costs functions and symbol updating are defined as:

$$\begin{cases} c_{sub}(x_i, y_j) & = d(x_i, y_j) \\ c_{merge}(x_i \to x_{i+1}) & = d(x_i, x_{i+1}); \textit{Update: } \bar{x}_{i \to i+1} = \mathit{merge}(x_i, x_{i+1}) \\ c_{merge}(y_j \to y_{j+1}) & = d(y_j, y_{j+1}); \textit{Update: } \bar{y}_{j \to j+1} = \mathit{merge}(y_j, y_{j+1}) \end{cases} \tag{4.1}$$

We also can define the weighting scheme for the edit operations similar to that of SMD as described in Chapter 3:

$$\begin{cases} c_{sub}(x_i, y_j) & = d(x_i, y_j) \\ c_{merge}(x_i \to x_{i+1}) & = w.d(x_i, x_{i+1}); \textit{Update: } \bar{x}_{i \to i+1} = \mathit{merge}(x_i, x_{i+1}) \\ c_{merge}(y_j \to y_{j+1}) & = w.d(y_j, y_{j+1}); \textit{Update: } \bar{y}_{j \to j+1} = \mathit{merge}(y_j, y_{j+1}) \end{cases} \tag{4.2}$$

where w is a weight parameter which controls the ratio between substitutions and insertion/deletion operations.

4.3.2 Merge operation

Merge operation of sequence of symbols:

In the definition, the merge operation is done on two successive symbols. However, when successive merge operations occurs, symbol x_{i+k} can be merged with sequence of k symbols $x_i, x_{i+1}, \ldots, x_{i+k-1}$. The result of the merging of the $k+1$ symbols $x_i, x_{i+1}, \ldots, x_{i+k}$ is denoted as $\bar{x}_{i \to i+k}$ and we have: $\bar{x}_{i \to i+k} = merge(\bar{x}_{i \to i+k-1}, x_{i+k})$.

The total cost for merging can be computed as the cost to merge first $(k-1)$ symbols with x_{i+k}. So:

$$\begin{aligned} c_{merge}(x_i \to x_{i+k}) &= c_{merge}(x_i \to x_{i+k-1}) + d(\bar{x}_{i \to i+k-1}, x_{i+k}) \\ &= c_{merge}(x_i \to x_{i+k-2}) + d(\bar{x}_{i \to i+k-2}, x_{i+k-1}) + d(\bar{x}_{i \to i+k-1}, x_{i+k}) \\ &= \sum_{l=1}^{k} d(\bar{x}_{i \to i+l-1}, x_{i+l}) \end{aligned}$$

where $c_{merge}(x_i \to x_{i+k})$ is the cost to merge k successive symbols $x_i, x_{i+1}, \ldots, x_{i+k}$.

We discuss about how to determine this new merged symbol in below.

Merge operation with normalization:

Since the merge operation means combination of two successive regions, merge operation of two symbols returns a new symbol as the averaged of the two histograms. The question that arises is what is the result of merging a sequence regions?

Because all original regions in an image string have the same role, a combination of a N merged regions with a new region should be considered as the merge of $N+1$ regions. Therefore, it is necessary to perform normalization of the regions before and after combining together.
Each region is represented as (H_n) where H is its histogram and n is the number of merging (i.e. number of regions used to compute H). The original regions have $n = 1$. Each time two symbols are merged, it is necessary to save both the sum of their two histograms and sum of number merging. The normalized histogram of a

new symbol is the result of the division the sum of the histograms with the total number of merging the original regions. For instance, the normalized histogram of the merged region (H_n) is $\frac{H}{n}$.

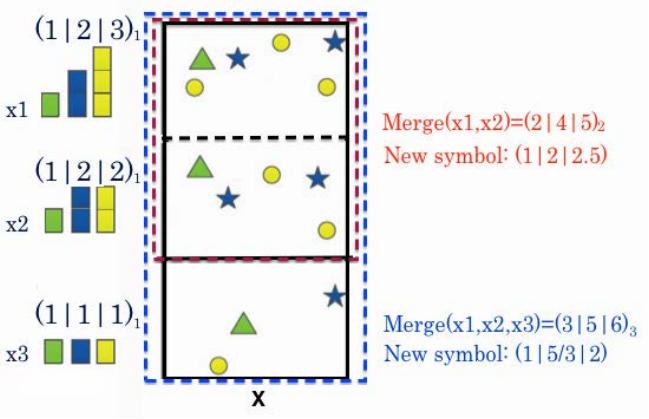


FIGURE 4.2: Example about using merge operation with normalization.

Example 1: A simple toy example of using the merge operation with normalization is illustrated on Figure 4.2, where the toy image is divided into three regions. This image is represented as local BoWs, using three visual codewords (star, triangle and circle). Image string $\mathcal{X} = \{x_1, x_2, x_3\}$ is described by the symbols histograms which are shown beside regions. A histogram is denoted as a triplet of three values separated by character "|". We choose l_1 as ground distance. The merging of region 1 and region 2 returns a new region $(2|4|5)_2$. The normalized histogram of the new symbol is $\frac{(2|4|5)}{2} = (1|2|2.5)$. The subsequent merging of region 3 returns a new region which is the sum of $(2|4|5)_2$ with $(1|1|1)_1$ and equals $(3|5|6)_3$. The normalized histogram of this new symbol is $\frac{(3|5|6)}{3} = (1|\frac{5}{3}|2)$

Example 2: In this example, we plan to illustrate the use of the merge operation in computation of the distance. The two toys images of Figure 4.3, are divided into three regions. These images are represented as strings of local BoWs with three visual codewords. We still choose l_1 as ground distance. To fill the cell $D_{(1,1)}$ of the distance matrix, we compute three different costs:

$$\begin{cases} Merge(x_1, x_2) : c_{merge}(x_1 \rightarrow x_2) & = d(x1, x2) = d_{l1}((1|2|3)/1, (1|2|2)/1) = 1 \\ Merge(y_1, y_2) : c_{merge}(y_1 \rightarrow y_2) & = d(y1, y2) = d_{l1}((2|1|1)/1, (0|2|1)/1) = 3 \\ Substitution(x_1, y_1) : c_{sub}(x1, y1) & = d(x1, y1) = d_{l1}((1|2|3)/1, (2|1|1)/1) = 4 \end{cases}$$

so a merge operation is applied on string $\mathcal{X}$ and after merging, a new symbol $\bar{x}_{1\rightarrow 2}$ is generated, as the average of the combined region $(1|2|3)_1 + (1|2|2)_1 = (2|2|5)_2$. The normalized histogram of new symbol is $\bar{x}_{1\rightarrow 2} = \frac{(2|2|5)}{2} = (1|2|\frac{5}{2})$.

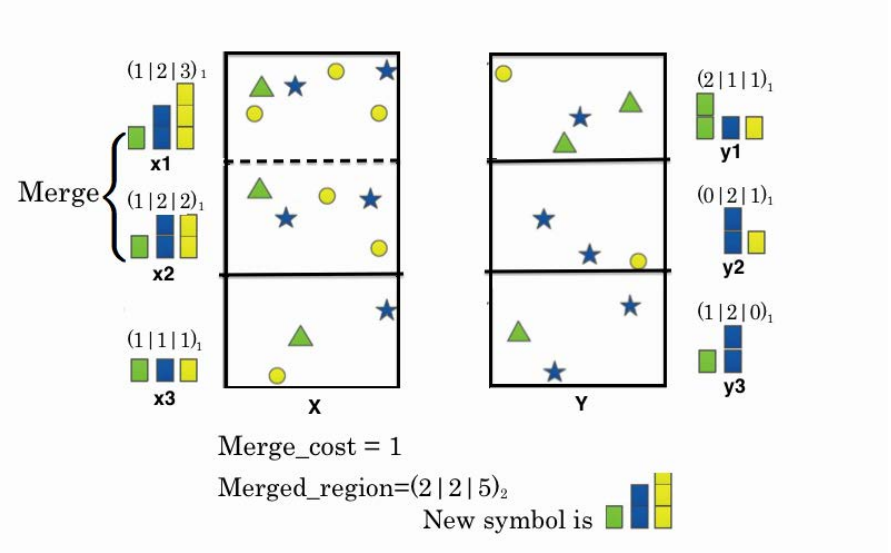


FIGURE 4.3: Example about using merge operation in computation of the distance.

4.4 New merge-based edit distance

The previous section has discussed the principle of the proposed merge-based String Matching Distance, m-SMD. In this part, we present in details the algorithm to compute this distance.

4.4.1 Greedy merge-based SMD algorithm

Compared to SMD, m-SMD requires to modify the symbols each time the merge operation is applied. More precisely, each merge operation generates a new symbol obtained by combining the histogram of the current symbol with the next one. As a consequence, the original strings are modified. If we use dynamic programming, as presented in Algorithm 1, to compute this distance; at the step (i, j), the update equation must revise as:

$$D_{(i,j)} = \min \begin{cases} D_{(i-1,j-1)} + c_{sub}(\bar{x}_i, \bar{y}_j) \\ D_{(i-1,j)} + c_{merge}(\bar{x}_i \rightarrow \bar{x}_{i+1}) \\ D_{(i,j-1)} + c_{merge}(\bar{y}_j \rightarrow \bar{y}_{j+1}) \end{cases} \tag{4.3}$$

where $\bar{\mathcal{X}} = \{\bar{x}_i\}$ and $\bar{\mathcal{Y}} = \{\bar{y}_j\}$ are strings obtained after the previous iteration. They can be the original strings (if only substitution is used) or modified ones (if any merge operations has been used).

To compute the distance, it is necessary to update these strings at each step. Note that it should not change the initial strings because other edit scripts not using this merge operation may be obtained in the dynamic programming algorithm. It is, therefore, essential to store two new matrices S_x and S_y respectively containing, for each cell (i, j), the next symbol of string $\mathcal{X}(M)$ and string $\mathcal{Y}(N)$ after applying each edit operation (See the example shows in Table 4.1 below).

Algorithm 2 describes in pseudo code how to compute m-SMD by dynamic programming. At each iteration, depending on which edit operation is employed, the following symbols are updated in *MergeUpdate function*.

Example :

An example of computation of the m-SMD using dynamic programming between two strings of histograms of three words $\mathcal{X} = \{(1|2|3), (1|2|3), (4|1|1), (4|2|0)\}$ and $\mathcal{Y} = \{(3|1|2), (3|1|2), (3|1|2), (4|0|2)\}$ is shown on Table 4.1.

D $\mathcal{Y}$ / $\mathcal{X}$	**(0|0|0)**	**(3|1|2)**	**(3|1|2)**	**(3|1|2)**	**(4|0|2)**
(0|0|0)	S_y $(3\|1\|2)_1$ **0** S_x $(1\|2\|3)_1$	$(6\|2\|4)_2$ **0** $(1\|2\|3)_1$	$(9\|3\|6)_3$ **0** $(1\|2\|3)_1$	$(13\|3\|8)_4$ **2** $(1\|2\|3)_1$	$(13\|3\|8)_4$ **8** $(1\|2\|3)_1$
(1|2|3)	$(3\|1\|2)_1$ **0** $(2\|4\|6)_2$	$(6\|2\|4)_2$ **0** $(2\|4\|6)_2$	$(9\|3\|6)_3$ **0** $(2\|4\|6)_2$	$(13\|3\|8)_4$ **2** $(2\|4\|6)_2$	$(0\|0\|0)_1$ **6.5** $(1\|2\|3)_1$
(1|2|3)	$(3\|1\|2)_1$ **6** $(6\|5\|7)_3$	$(3\|1\|2)_1$ **4** $(4\|1\|1)_1$	$(3\|1\|2)_1$ **4** $(4\|1\|1)_1$	$(4\|0\|2)_1$ **4** $(4\|1\|1)_1$	$(0\|0\|0)_1$ **6.5** $(4\|1\|1)_1$
(4|1|1)	$(3\|1\|2)_1$ **32/3** $(10\|7\|7)_4$	$(3\|1\|2)_1$ **6** $(8\|3\|1)_2$	$(3\|1\|2)_1$ **6** $(4\|2\|0)_1$	$(4\|0\|2)_1$ **6** $(4\|2\|0)_1$	$(0\|0\|0)_1$ **6** $(4\|2\|0)_1$
(4|2|0)	$(3\|1\|2)_1$ **50/3** $(10\|7\|7)_4$	$(3\|1\|2)_1$ **12** $(8\|3\|1)_2$	$(3\|1\|2)_1$ **9** $(0\|0\|0)_1$	$(4\|0\|2)_1$ **10** $(0\|0\|0)_1$	$(0\|0\|0)_1$ **10** $(0\|0\|0)_1$

TABLE 4.1: The computation of m-SMD using dynamic programming as described in Algorithm 2.

Algorithm 2 m-SMD: merge-based String Matching Distance Algorithm using Dynamic Programming

Input: Two strings $\mathcal{X}$(M) and $\mathcal{Y}$(N)
Initial:
$D(0,0) \leftarrow 0$
$S_x(0,0) \leftarrow x_1$
$S_y(0,0) \leftarrow y_1$
for $i = 1$ *to* M **do**
  $D_{(i,0)} = D_{(i-1,0)} + c_{merge}(S_x(i-1,0) \rightarrow x_{i+1})$
  $mergeUpdate\ (S_x, S_y, i, 0, 1)$
end for
for $j = 1$ *to* N **do**
  $D_{(0,j)} = D_{(0,j-1)} + c_{merge}(S_y(0,y_j) \rightarrow y_{j+1})$
  $mergeUpdate\ (S_x, S_y, 0, j, 2)$
end for
Loop:
for $i = 1$ *to* M **do**
  for $j = 1$ *to* N **do**
    $d_1 = D_{(i-1,j)} + c_{merge}(S_x(i-1,j) \rightarrow x_{i+1})$
    $d_2 = D_{(i,j-1)} + c_{merge}(S_y(i,j-1) \rightarrow y_{j+1})$
    $d_3 = D_{(i-1,j-1)} + c_{sub}(S_x(i-1,j-1), S_y(i-1,j-1)$
    $Index\ k \leftarrow \arg_{k=(1,2,3)} \min d_k$
    $D_{(i,j)} = d_k$
    $mergeUpdate\ (S_x, S_y, i, j, k)$
  end for
end for
return $D(M,N)$

mergeUpdate Algorithm:
Input: S_x, S_y, i, j, k
if $k = 1$ **then**
  $S_x(i,j) \leftarrow S_x(i-1) + x_{i+1}$
  $S_y(i,j) \leftarrow y_{j+1}$
else if $k = 2$ **then**
  $S_x(i,j) \leftarrow x_{i+1}$
  $S_y(i,j) \leftarrow S_y(j-1) + y_{j+1}$
else $\{k = 3\}$
  $S_x(i,j) \leftarrow x_{i+1}$
  $S_y(i,j) \leftarrow y_{j+1}$
end if

The first column and the first row of the table are the original strings $\mathcal{X}$ and $\mathcal{Y}$. We use a blue color for the matrix S_x and a red color for S_y. The values which are saved in S_x and S_y are (H_n) where H is the combination histograms, n is the total number of merging. Note that, the computation of the edit operation cost needs to use the normalized histogram, which is $\frac{H}{n}$.
In the middle of each cell is the value of distance matrix D.

To fill out the matrix, we start with initial values: $D_{(0,0)} = 0$, $S_x(0,0) = x_1 = (1|2|3)_1$ and $S_y(0,0) = y_1 = (3|1|2)_1$. The value of $D_{(1,0)}$ is obtained by merging x_1 with x_2 and is computed as:

$$\begin{aligned} D_{(1,0)} &= D_{(0,0)} + c_{merge}(S_x(0,0), x_2) \\ &= 0 + d_{l_1}((1|2|3)/1, (1|2|3)) = 0(\textit{line 7 in Algorithm 2}). \end{aligned}$$

After merging, we have to update the symbols and save them to the matrices S_x, S_y (*line 8)*,

$$\begin{aligned} S_x(1,0) &\leftarrow S_x(0,0) + x_2 = (1|2|3)_1, (1|2|3)_1 = (2|4|6)_2 \\ S_y(1,0) &\leftarrow y_1 = (3|1|2)_1 \end{aligned}$$

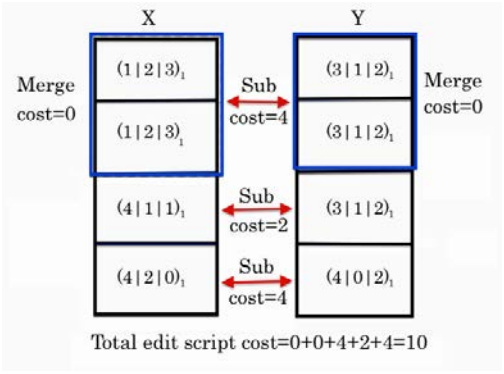
The information of $D_{(1,0)}, S_x(1,0)$ and $S_y(1,0)$ are saved into the cell. This information will be used in future computations with edit scripts related to this cell. Using the loop from *line 6* to *line 9* we can fill the second column of the table. In the same way, we can fill the second row by applying the loop from *line 10* to *line 13*. Now, to fill the rest of the table, we apply the loop from *line 15* to *line 24*. For instance, we can calculate the values of $D(2,3)$ (cell (4,5)) of the table) by using the information of the cells $D(1,2)$, $D(1,3)$ and $D(2,2)$:

$$\begin{cases} d_1 = & D_{(1,3)} + c_{merge}(S_x(1,3) \rightarrow x_3) = 2 + d_{l_1}(\frac{(2|4|6)}{2}, \frac{(4|1|1)}{1}) = 8 \\ d_2 = & D_{(2,2)} + c_{merge}(S_y(2,2) \rightarrow y_4) = 4 + d_{l_1}(\frac{(3|1|2)}{1}, \frac{(4|0|2)}{1}) = 6 \\ d_3 = & D_{(1,2)} + c_{sub}(S_x(1,2), S_y(1,2)) = 0 + d_{l_1}(\frac{(2|4|6)}{2}, \frac{(9|3|6)}{3}) = 4 \end{cases}$$

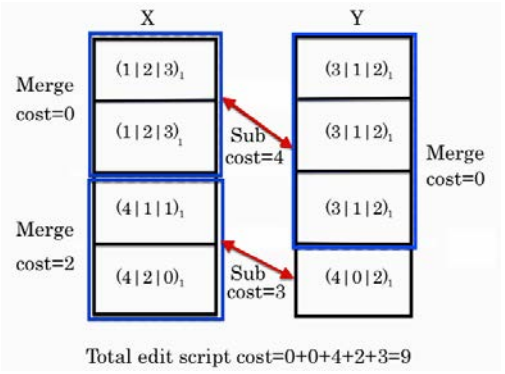
$\min(d_1, d_2, d_3)$ is d_3 so $D_{(2,3)} = d_3 = 4$ and $k = 3$ such that *mergeUpdate* will be:

$$S_x(2,3) \leftarrow x_3 = (4|1|1)_1$$
$$S_y(2,3) \leftarrow y_4 = (4|0|2)_1$$

The information of $D_{(2,3)}, S_x(2,3)$ and $S_y(2,3)$ are then saved into this cell. Continuing the computation, we obtain the distance of the two image strings $\mathcal{X}$ and $\mathcal{Y}$ is 10, which is the value of the last cell of the table. We also obtain the edit script, in this case, is highlighted on the Table 4.1. Following the edit script, it can be plotted the matching alignment between two image strings as in Figure 4.4(a). Note that in some cases, we can have several possibilities to choose among d_1, d_2, d_3 (*line 20* in Algorithm 2). For example, to fill in cell $D_{(2,2)}$, we can equally choose a merging from cell $D_{(2,1)}$ or a substitution from cell $D_{(1,1)}$. The choice of keeping the merging or the substitution could affect subsequent computation leading to a non optimal edit script as shown in Figure 4.4. This is why this algorithm is qualified as greedy.



(a) String alignment and final cost which obtained by greedy method.



(b) A better edit script

FIGURE 4.4: An example shows the greedy method is not optimal/true edit distance, since it does not produce a minimum edit script to convert one string to the another string.

Computational complexity:

The calculation of m-SMD between two image strings of length M and N is described in Algorithm 2. Like the computation of SMD, it is required to fill out the matrix $D(M \times N)$. At each step, we have to determine not only three edit operations cost (as SMD) but also need to compute the mergeUpdate new symbol. With the local histogram of size K and a l_1 as ground distance, it requires $\mathcal{O}(K)$ operations for computing each edit cost or update symbol. However, using algorithm theory, the complexity of each step is still $\mathcal{O}(K)$. So finally, it needs

$\mathcal{O}(K \times M \times N)$ operations to compute m-SMD between two image strings of length M and N.
If the images are divided into B bands and N regions, the computation must be repeated B times. Compared to SMD, m-SMD needs more computation time because of the merge update step. However, both algorithms have the same complexity, which is $\mathcal{O}(K \times B \times N^2)$.

Discussion:

The edit-distance between two strings has been defined as the minimum edit script cost to convert one string into the other one. However, in Algorithm 2, due to the changes of the symbols during the computation and as a consequence, multiple possibilities to update symbols, there is no guarantee that the final distance is the minimum edit script cost. For instance, in the example above, we can have an other edit-script which introduces a smaller edit cost, as shown on Figure 4.4(b).

4.4.2 Recursive merge-based SMD

To obtain the optimal distance (i.e. the minimum cost of edit scrip), it is necessary to check all the possible edit scripts to convert the first string into the second one. In this case, the recursive solution is a good choice since it basically checks all the possibilities and computes the minimum edit script cost to convert one string into the other. The recursive algorithm is described as below:

Algorithm 3 m-SMD : Recursive merge-based String Matching Distance Algorithm

Input: Two strings $\mathcal{X}(M) = \{x_1, x_2, ..., x_M\}$ and $\mathcal{Y}(N) = \{y_1, y_2, ..., y_N\}$
if $M = 0$ **then**
  return *cost-to-Delete* $\mathcal{Y}(N)$
end if
if $N = 0$ **then**
  return *cost-to-Delete* $\mathcal{X}(M)$
end if
$d_1 = c_{merge}(x_1, x_2) + \textbf{m-SMD}(\bar{\mathcal{X}}(M-1), \mathcal{Y}(N))$
$d_2 = c_{merge}(y_1, y_2) + \textbf{m-SMD}(\mathcal{X}(M), \bar{\mathcal{Y}}(N-1))$
$d_3 = c_{sub}(x_1, y_1) + \textbf{m-SMD}(\mathcal{X}(M-1), \mathcal{Y}(M-1))$
return $\min(d_1, d_2, d_3)$

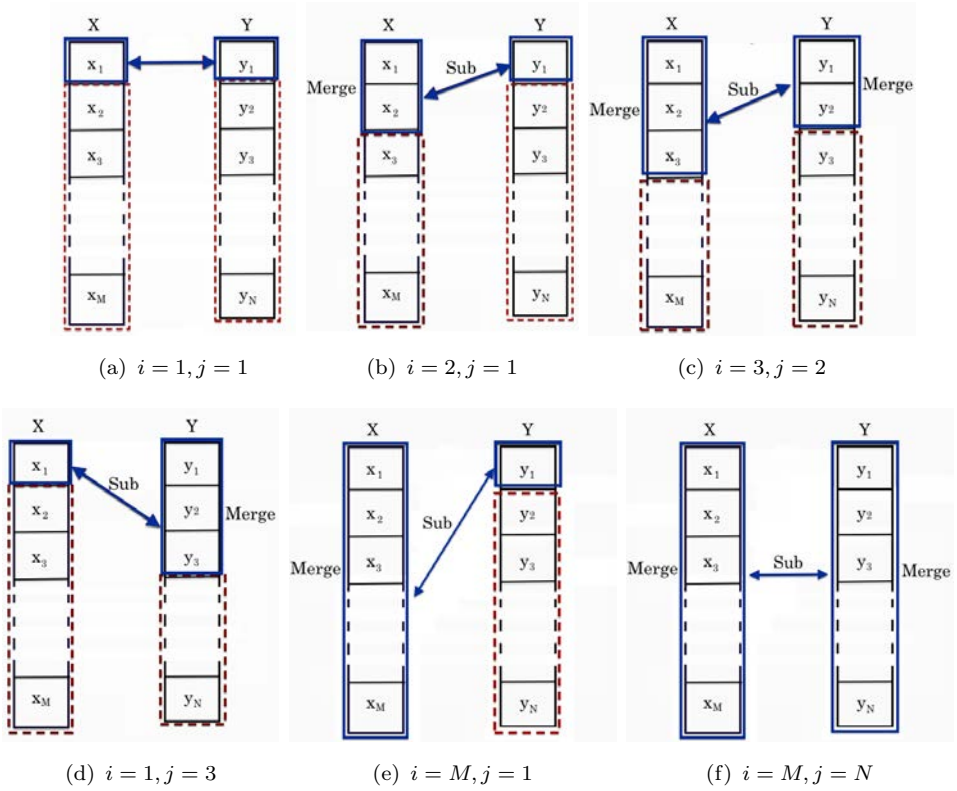
where $\bar{\mathcal{X}}$ is the updated-string after doing a merge operation on the two first symbols x_1, x_2 of string $\mathcal{X}$; $\bar{\mathcal{Y}}$ is the updated-string after doing a merge operation on the two first symbols y_1, y_2 of string $\mathcal{Y}$.
It can be noticed that, after each recursive call, the total length of strings is reduced. Depending on the edit operation, merging on string $\mathcal{X}$, merging on string $\mathcal{Y}$ or substitution, we call the function again but with smaller strings. If a merge operation is used, it is necessary to update the strings into $\bar{\mathcal{X}}$ or $\bar{\mathcal{Y}}$ and to employ the function on these new smaller updated strings. The recursive call stops till the two strings are empty. However, the main drawback of this algorithm is its high complexity - which is $\mathcal{O}(3^{max(N,M)})$. For practical purpose, it is essential to reduce the computation time. Since the method may repeatedly calculate several times the distance between sub strings, the possible solution is to re-use this information. We notice that:

- Successive merge operations on the first and the second string can be done in any order leading to identical merged symbols and identical total merge cost.
- After a substitution, the two remaining strings do not contain any merged symbols. It means, that the two remaining strings are identical to the original sub-strings.

Thus, the general scenario for matching two strings $\mathcal{X}$ and $\mathcal{Y}$ is:

- Merging the first i symbols of $\mathcal{X}$, which costs $c_{merge}(x_1 \rightarrow x_i)$. $i = 1$ means that no merge operations are employed on string $\mathcal{X}$. The resulting combined symbol is $\bar{x}_{1\rightarrow i}$.
- Merging the first j symbols of $\mathcal{Y}$, which costs $c_{merge}(y_1 \rightarrow y_j)$. $j = 1$ means that no merge operations are employed on string $\mathcal{Y}$. The resulting combined symbol is $\bar{y}_{1\rightarrow j}$.
- Substituting $\bar{x}_{1\rightarrow i}$ with $\bar{y}_{1\rightarrow j}$, which costs $c_{sub}(\bar{x}_{1\rightarrow i}, \bar{y}_{1\rightarrow j})$
- Re-computing the matching distance between the two sub-strings of $\mathcal{X}$, $\mathcal{Y}$: $\{x_{i+1}, \dots x_M\}$ and $\{y_{j+1}, \dots y_N\}$

Several examples of the matching scenario are illustrated on Figure 4.5.



(a) $i = 1, j = 1$ (b) $i = 2, j = 1$ (c) $i = 3, j = 2$

(d) $i = 1, j = 3$ (e) $i = M, j = 1$ (f) $i = M, j = N$

FIGURE 4.5: The matching scenarios: After matching two symbols or sequence of symbols (blue parts), we have to recompute the distance for two sub-strings (red parts).

The total edit script cost for this scenario is:

$$\begin{aligned} \textit{Total-cost} =& c_{merge}(x_1 \to x_i) + c_{merge}(y_1 \to y_j) + c_{sub}(\bar{x}_{1\to i}, \bar{y}_{1\to j}) + \\ &+ \textbf{m-SMD}(\{x_{i+1}, \dots x_M\}, \{y_{j+1}, \dots y_N\}) \end{aligned}$$

Since, the edit distance is the minimum cost of all possible edit scripts, we have the equation:

$$\begin{aligned} \textbf{m-SMD}(\mathcal{X}_{(1\to M)}, \mathcal{Y}_{(1\to N)}) = \min_{i=\overline{1,M} j=\overline{1,N}} & (c_{merge}(x_1 \to x_i) + c_{merge}(y_1 \to y_j) + \\ & + c_{sub}(\bar{x}_{1\to i}, \bar{y}_{1\to j}) + \textbf{m-SMD}(\mathcal{X}_{(i+1\to M)}, \mathcal{Y}_{(j+1\to N)}) \end{aligned}$$

Using this formula, it is clear that we can re-use the information about the distance of the sub-strings; so it can reduce a lot the computation time.

Computation complexity:

The evaluation of the algorithm complexity requires to count the total number of possible merge operations on strings $\mathcal{X}$ and $\mathcal{Y}$, as well as the total number of possible substitutions.
For string $\mathcal{X}_{(1\to N)} = \{x_1, x_2, \dots x_N\}$ of length N, we count $N-1$ possible merge operations involving x_1: $merge(x_1, x_2)$, $merge(x_1, x_2, x_3)$, ... , $merge(x_1, \dots, x_N)$.
In the same way, for sub-string $\mathcal{X}_{(2\to M)} = \{x_2, \dots x_N\}$, there are $N-2$ possible merge operations involving x_2.
Therefore, the total number of possible merge operations on the string $\mathcal{X}_{(1\to N)}$ is $(N-1) + (N-2) + (N-3) + \dots + 0$ and is therefore equal to $N \times (N-1)/2$.
With $N \times (N-1)/2$ possible symbols, there are $N^2 \times (N-1)^2/4$ possible ways to match symbols at most.
In fact, the number of matches is smaller than this. We have indeed to take into account an ordering constraint: a match always takes place from the beginning of the string, and when one match is done, the next match will concern a smaller sub-string. Therefore, the complexity to compute all possible substitutions is $\mathcal{O}(N^4)$ and to compute all possible merge symbols is $\mathcal{O}(N^2)$.
At the end, the new method has only a polynomial complexity of order 4.
We could also think about reducing the total number of operations by applying a merging constraint. In this case, we limit the maximum number of possible combined symbols (i.e. maximum length of sequence of merging symbols). For instance, if we fix the number of possible matches on a string of length N to a, $a \ll N$, then the maximum total possible number of merge operations on this string is only $N \times a$. As a consequence, the maximum number of possible substitutions is $N^2 \times a^2$. Thus, the algorithm complexity is reduced to $\mathcal{O}(N^2)$.

4.5 Experiments

In these experiments, we aim to compare the performance of SMD, greedy m-SMD and recursive m-SMD with respect to accuracy and efficiency. Three datasets, 15 Scene, Caltech 101 and Pascal 2007 are used for evaluation. Local features of all the images are extracted by using a dense SIFT descriptor on a regular grid 16x16 pixels and with a step size of 8 pixels. The k-means clustering approach is applied on a subset of descriptors to create the visual codebook. The codebook size is set to $K = 100$. We report the classification accuracy using both hard coding and

sparse coding for Caltech 101 and 15 Scene dataset. For Pascal 2007, only results using hard coding are shown.

Classification performance on 15 Scene dataset:

The SMD, greedy m-SMD and recursive m-SMD distances are computed with $N = 16$ and the weight is fixed to $w = 0.8$. We report both cases: using single level ($B = 1$) and pyramid case with two levels ($L = 2$). In order to evaluate the classification performance, we use the same train/test setup and kernel edit distance as described in Chapter 3 which are: 100 images per class for training and the rest for testing . We also recompute the baseline. In the single level case, the baseline corresponds to the BoW performance and in the pyramid case, it corresponds to the SPR framework. The results are shown on Table 4.2.

As shown, the recursive merge-based m-SMD approach outperforms both the original SMD and greedy m-SMD approaches in all cases. It confirms the effectiveness of the combination of similar regions within images. However, the difference of classification performance between SMD and recursive m-SMD is small (from 0.02 % to 0.6%).

Coding	Approaches	Single level	Pyramid
Hard-coding	baseline	73.28 ± 0.55	75.48 ± 0.58
	SMD	82.36 ± 0.48	83.14 ± 0.67
	greedy m-SMD	82.49 ± 0.68	83.07 ± 0.66
	recursive m-SMD	**82.65 ± 0.65**	**83.16 ± 0.77**
Sparse-coding	baseline	63.49 ± 0.78	78.79 ± 0.59
	SMD	83.36 ± 0.64	84.59 ± 0.70
	greedy m-SMD	83.21 ± 0.39	85.16 ± 0.59
	recursive m-SMD	**83.84 ± 0.47**	**85.33 ± 0.76**

TABLE 4.2: Classification accuracy for the 15 Scene dataset using different approaches and coding methods. The results of SMD, greedy m-SMD, recursive m-SMD are obtained with $N = 16$, $B = 1$ for single level, L=2 for pyramid, K=100 and w=0.8

The behavior of greedy m-SMD is not so clear. In some cases, as the single level case with hard coding or the pyramid with sparse coding, it seems to work better than SMD. But in other cases, as pyramid with hard coding or single level with sparse coding, the greedy version is slightly lower than SMD. All string-based distances, SMD, greedy m-SMD and recursive m-SMD outperform the baseline. The best performance, 85.33 %, is obtained with recursive version with sparse coding and pyramid scheme.

Classification performance on Caltech 101 dataset:

Coding	Approaches	Single level	Pyramid
Hard-coding	baseline	52.39 ± 0.34	58.66 ± 0.42
	SMD	65.46 ± 0.56	66.52 ± 0.51
	greedy m-SMD	64.71 ± 0.68	66.12 ± 1.07
	recursive m-SMD	**65.78 ± 0.98**	**66.90 ± 0.86**
Sparse-coding	baseline	55.05 ± 0.71	57.10 ± 0.93
	SMD	73.05 ± 0.64	73.48 ± 1.2
	greedy m-SMD	71.75 ± 0.97	73.37 ± 0.92
	recursive m-SMD	**73.24 ± 1.09**	**73.92 ± 0.91**

TABLE 4.3: Classification accuracy for Caltech 101 dataset using different approaches and coding methods. The results of SMD, greedy m-SMD, recursive m-SMD are obtained with $N = 13$; $B = 4$ for single level; L=2 for pyramid, K=100 and w=0.8

Table 4.3 shows results on Caltech 101 dataset with hard assignment coding and sparse coding methods. The distances are computed with $N = 16$, $K = 100$ and weight w=0.8. In the single level case, B is fixed to 4. We use $L = 2$ for pyramid scheme. We evaluate the classification performance using SVM one vs all, with the edit kernel discussed in Chapter 3. We randomly select 30 images per class for training and maximum 50 images per class for testing. The experiment is repeated 10 times and we report the average accuracy.

From this table we can see that: (i) The string-based approaches improve the baseline for all cases. (ii) The recursive merge-based distance is the best one. (iii) The greedy framework sometimes produces lower results than SMD. (iv) Again, the best classification accuracy (73.92 %) is obtained with recursive framework using the pyramid scheme and sparse coding.

Classification performance on Pascal 2007 dataset:

The Pascal 2007 dataset [Everingham et al., 2007] consists of 9963 images of 20 classes. It is an extremely challenging one due to variation of object scales and poses. The classification performance is evaluated using Average Precision (AP), following the setup for training/testing set of the VOC 2007 challenge. The training set is composed of 5011 images and the test set contains 4952 images. The classification performance of the three distances and the baseline (SPR) are shown on Table 4.4

Object class	SPR	SMD	greedy m-SMD	recursive m-SMD
Aeroplane	58.16	60.33	62.58	**62.62**
Bicycle	30.55	46.99	47.13	**48.31**
Bird	29.14	**29.52**	19.41	26.67
Boat	51.10	58.98	57.97	**60.50**
Bottle	**20.79**	16.39	19.03	18.48
Bus	39.36	52.84	54.34	**55.13**
Car	63.35	68.68	69.94	**70.42**
Cat	29.83	39.59	39.36	**40.29**
Chair	41.25	48.05	48.05	**48.90**
Cow	23.41	33.73	33.78	**34.38**
Table	36.80	44.11	45.00	**45.80**
Dog	31.72	**35.51**	33.23	35.36
Horse	65.96	70.60	71.59	**72.83**
Motorbike	41.73	50.98	53.29	**53.64**
Person	72.44	75.79	76.46	**77.97**
Plant	18.12	17.56	23.26	**23.13**
Sheep	26.85	32.51	25.99	**33.08**
Sofa	33.39	42.47	44.00	**44.73**
Train	55.01	67.72	67.94	**69.07**
TV	29.04	39.95	40.17	**41.29**
Mean	39.90	46.62	46.63	**48.13**

TABLE 4.4: Image classification results (AP) on Pascal 2007 dataset using different frameworks. The codebook size is fixed to $K = 100$ for all approaches. The results of SMD, greedy m-SMD, recursive m-SMD are obtained with $N = 16$; pyramid with $L = 2$ and $w = 0.8$.

The string based approaches have shown significant improvement over the SPR baseline (from +6.7 % in case of SMD to 8.23 % in case of recursive m-SMD). The recursive merged-based framework is the winner since it has the best performance for 17 classes out of 20. The merged regions idea seems very well suited to the *Plant* class, since it improves by about 5.5 % the classification performance. Also with the *Aeroplane, Bottle, Bus, Horse, Motorbike, Person and Sofa* classes, the gap between the two methods is more than 2 %. Again, the performance of the greedy framework is still very confusing. For several classes, it works equally or slightly better than SMD but for some classes, for example *Bird, Sheep, Dog* it

fails to handle the classification. However, in average, it produces lightly the same performance as SMD. The best AP is 48.13 % obtained with recursive m-SMD, which is higher than the results obtained with SMD or greedy m-SMD by about 1.5 %.

Complexity:

The purpose of the following experiments is to examine the computation time of the three string of histogram distances with respect to different parameters. We take randomly 1000 pairs of images from the 15 Scene dataset and calculate the execution time needed to compute the 1000 distances. The implementation is in C++ and the experiments are run on a $2700MHz$ PC under Linux operating system. Experiments are repeated 10 times and we report the average value.

Figure 4.6 illustrates the execution times obtained with SMD and greedy m-SMD varying the number of regions N from 1 to 16. Note that the *x-axis* is N^2. Here, the codebook size is fixed to $K = 100$ and the number of bands to $B = 1$. From this figure, we notice that, when the number of regions is small ($N < 9$), the time to compute either SMD or greedy m-SMD is nearly the same. However, when N is high, more time is needed to compute the greedy distance. However, the gap between the two lines is still small. In addition, both curves are almost linear functions of the square of the number of regions. It confirms that the complexity of SMD and greedy m-SMD is $\mathcal{O}(N^2)$.

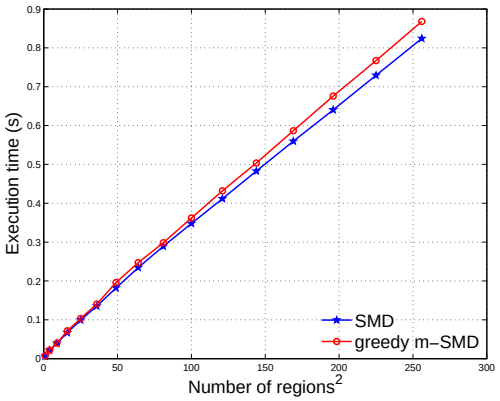


FIGURE 4.6: Implementation times (s) to compute 1000 SMD or greedy m-SMD distances versus square of Number of regions. The images are taken from 15 Scene dataset. Here, $B = 1$, $K = 100$.

We also study the complexity of recursive m-SMD in Figure 4.7 and Figure 4.8 compares the computation time of the three proposed string distances. The codebook size is fixed to $K = 100$ words and the number of bands is fixed to $B = 1$. First, to verify that the complexity of recursive m-SMD is $\mathcal{O}(N^4)$, we plot the

computation time versus the power four of N in Figure 4.7. We obtain a linear curve, which provides an experimental proof of the algorithm complexity given in section 4.4.2.

In Figure 4.8, we notice that: when N is small, the gap between recursive m-SMD and the two other lines is not too significant. Nevertheless, it dramatically increases when N becomes higher.

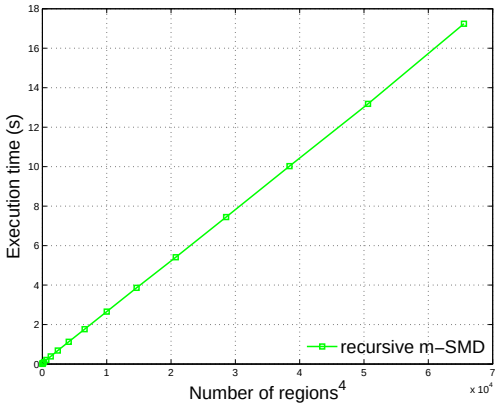


FIGURE 4.7: Implementation times (s) to compute 1000 recursive m-SMD distances versus the power four of the number of regions. Images are taken from the 15 Scene dataset. Here, $B = 1$, $K = 100$.

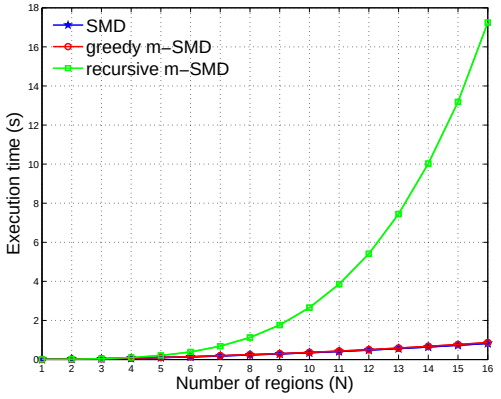


FIGURE 4.8: Implementation times (s) to compute 1000 SMD, greedy m-SMD or recursive m-SMD distances versus the number of regions.

4.6 Conclusion

This chapter has presented a new merge-based string matching edit-distance, along with two algorithms to compute it: a greedy one which uses Dynamic Programming and an optimal one which is recursive. Thanks to the new merge edit operation, the recursive merge-based distance has shown improvement over SMD

distance discussed in the previous chapter. The results on Pascal 2007 dataset show that for several object classes, the idea of merging regions can improve significantly the image classification. However, the computational cost of the exact version is still high ($\mathcal{O}(N^4)$). As explained in section 4.4.2, this complexity can be reduced to $\mathcal{O}(N^2)$ by constraining the total number of merging operations but further experiments must be done to analyze the impact of the constraint on classification accuracy. Without considering this constraint, the complexity of the recursive version is still too high against the positive gain obtained over SMD (about 0.6 % for 15 Scene, 0.44 % for Caltech 101, 1.5 % for Pascal 2007). If we want to employ the recursive merge-based SMD framework for practical purpose, such as classifying a large dataset, it would be necessary to find a solution to reduce the computation time. This problem has not been solved in this thesis yet. Hopefully, it could be solved using GPU in the near future.

Chapter 5

Conclusions

This thesis has addressed the problem of rigid matching in Spatial Pyramid Matching based approaches. Our objective was to introduce a string matching model in order to improve image representation and comparison in the context of image classification. We have proposed to transform the local BoW representation into strings of local BoW and to introduce a new class of string of local histogram edit distances to measure image similarity. Below the summary of the key contributions of our work.

- Our first contribution was to introduce a new string-based image representation model. In this model, the image is divided into regions, each region is represented as a local histogram and treated as a symbol of a string. In addition, we proposed a *Pyramid like strategy* which includes a multi-resolution scheme into our string representation. Our string model has the advantage of incorporating order information between regions which can be further used during image matching. The new representation was shown to be more effective than the classical Pyramid matching representation when using region by region matching.

- In order to take into account the sequential aspect of strings of local histograms, our second contribution was to propose a new edit distance, denoted String Matching Distance (SMD). Unlike previous works of Yeh and Cheng [2008]; Ballan et al. [2010] which use fix edit operation costs, we went a step further with our SMD by proposing a new definition for deletion and insertion costs. The image matching problem was formulated as finding the optimal edit script to align two given image strings. This new definition

helps to detect and set aside identical successive regions within images and therefore improves the quality of the image matching. In addition, a substitution weight was incorporated into the cost definition to control the balance between substitutions and insertion/deletion operations. Moreover, the new distance can be efficiently calculated using Dynamic Programming.

- The SMD distance was shown to outperform rigid pairwise region matching. It is because of its ability to detect and ignore identical successive regions within images. We explored this idea and proposed to replace insertions and deletions by region merging. Our third contribution was then to introduce an extended version of SMD distance, called merge-based SMD, which indeed supports a *merge operation*. Thanks to this new operation, the number of sub regions and the grid divisions are adapted during the matching according to the visual content of images. This brings more flexibility in the image matching process.

- In the merge-based SMD distance, a new symbol is generated after each merging and, as a consequence, the symbols are modified during the string alignment. Then, the direct extension of SMD algorithm with merge operations does not produce the optimal edit script. The corresponding dynamic programming based algorithm is thus qualified as a greedy version. Our fourth contribution was to introduce a new version which evaluates all the possible edit scripts and returns the optimal one. An efficient recursive algorithm was proposed to compute the result in a 4th order polynomial complexity. The experimental evaluation demonstrated that the optimal merge-based approach is more efficient than both the greedy version and the insertion/deletion based approaches.

- In the context of image classification, all the versions of the proposed string-based distances showed significant improvements over classical methods. They were evaluated using both hard and sparse coding. We believe that these distances can be integrated in any other coding methods (as LLC [Wang et al., 2010], semi soft coding [Liu et al., 2011]) or with several higher-order statistics models such as Fisher Vectors (FV) [Perronnin et al., 2010], Vectors of Locally Aggregated Descriptors (VLAD) [Jégou et al., 2010] or Super-Vector Coding (SVC) [Zhou et al., 2010] to get better classification performances.

Furthermore, there are still several open questions which can be studied in future research.

- **Improve the complexity:** One of the main drawbacks of the merge-based edit distance is its computational complexity. We can remark that the $O(n^4)$ complexity can be reduced to $O(n^2)$ if we limit the number of successive symbol merging. A possible extension of the recursive algorithm could then be to fix the maximum number of successive merge operations and to consider the corresponding optimal edit script. An evaluation of this approach must be done to see if it is relevant and to determine the optimal number of maximum successive merge operations.

- **Tree or graph based edit distances:** As discussed in section 3.3 a promising approach is to provide a 2D extension of our work to better account for the 2D nature of images. Although some 2D-string extensions or tree-based approaches have been proposed, the direct adaptation of our work to these extensions is still an open question.

- **Using multi image features and multi kernel learning:** Due to time limitation, we just tested our algorithms on very simple features (dense SIFT feature) to evaluate our new propositions. It is possible to extend the methods with denser features or multiple image features (such as color and shape) with multiple kernel learning ([Vedaldi et al., 2009]) in order to obtain better image classification accuracy.

- **Applications to other domains:** On the other hand, it is very promising to evaluate the proposed distances in other domains such as video classification [Adjeroh et al., 1999; Ballan et al., 2010], pattern recognition in document scanning and text retrieval [Christodoulakis and Brey, 2009] or shape recognition [Tsai and Yu, 1985; Yeh and Cheng, 2008].

- **Validation on large datasets:** The last but not the least, it is of practical importance to evaluate the proposed approaches on large scales datasets such as Image Net ([Deng et al., 2009]).

Bibliography

Adjeroh, D. A., Lee, M.-C., and King, I. (1999). A distance measure for video sequences. *Computer Vision and Image Understanding*, 75(1):25–45.

Aizerman, A., Braverman, E. M., and Rozoner, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837.

Avila, S., Thome, N., Cord, M., Valle, E., and De A AraúJo, A. (2013). Pooling in image representation: The visual codeword point of view. *Computer Vision and Image Understanding*, 117(5):453–465.

Ballan, L., Bertini, M., Del Bimbo, A., and Serra, G. (2010). Video event classification using string kernels. *Multimedia Tools and Applications*, 48(1):69–87.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522.

Bosch, A., Zisserman, A., and Munoz, X. (2007). Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. ACM.

Bosch, A., Zisserman, A., and Muoz, X. (2008). Scene classification using a hybrid generative/discriminative approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(4):712–727.

Boureau, Y.-L., Bach, F., LeCun, Y., and Ponce, J. (2010). Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2559–2566. IEEE.

Boureau, Y.-L., Le Roux, N., Bach, F., Ponce, J., and LeCun, Y. (2011). Ask the locals: multi-way local pooling for image recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2651–2658. IEEE.

Cao, Y., Wang, C., Li, Z., Zhang, L., and Zhang, L. (2010). Spatial-bag-of-features. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3352–3359. IEEE.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Christodoulakis, M. and Brey, G. (2009). Edit distance with combinations and splits and its applications in ocr name matching. *International Journal of Foundations of Computer Science*, 20(06):1047–1068.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Dorkó, G. and Schmid, C. (2005). Object Class Recognition Using Discriminative Local Features. Rapport de recherche RR-5497, INRIA.

Duchenne, O., Joulin, A., and Ponce, J. (2011). A graph-matching kernel for object categorization. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1792–1799. IEEE.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, pages 178–178.

Gao, S., Tsang, I. W.-H., and Chia, L.-T. (2010). Kernel sparse representation for image classification and face recognition. In *Computer Vision–ECCV 2010*, pages 1–14. Springer.

Gehler, P. and Nowozin, S. (2009). On feature combination for multiclass object classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 221–228. IEEE.

Gokalp, D. and Aksoy, S. (2007). Scene classification using bag-of-regions representations. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Grauman, K. and Darrell, T. (2005). The pyramid match kernel: Discriminative classification with sets of image features. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1458–1465. IEEE.

Grauman, K. and Leibe, B. (2011). *Visual object recognition*. Number 11 in ISSN. Morgan & Claypool Publishers.

Harada, T., Ushiku, Y., Yamashita, Y., and Kuniyoshi, Y. (2011). Discriminative spatial pyramid. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1617–1624. IEEE.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK.

Harris, Z. S. (1954). Distributional structure. *Word*, pages 146–162.

Hong-Thinh, N., Cecile, B., and Christophe, D. (2014). Approximate image matching using strings of bag of visual words representation. In *9th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP-2014)*. SCITEPRESS.

Iovan, C., Picard, D., Thome, N., and Cord, M. (2012). Classification of urban scenes from geo-referenced images in urban street-view context. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 339–344. IEEE.

Jain, A. K. and Vailaya, A. (1996). Image retrieval using color and shape. *Pattern recognition*, 29(8):1233–1244.

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE.

Jiang, Y., Yuan, J., and Yu, G. (2012). Randomized spatial partition for scene recognition. In *Computer Vision–ECCV 2012*, pages 730–743. Springer.

Jurie, F. and Triggs, B. (2005). Creating efficient codebooks for visual recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 604–610. IEEE.

Ke, Y. and Sukthankar, R. (2004). Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE.

Khurshid, K., Faure, C., and Vincent, N. (2009). A novel approach for word spotting using merge-split edit distance. In *Computer Analysis of Images and Patterns*, pages 213–220. Springer.

Kim, J. and Grauman, K. (2010). Asymmetric region-to-image matching for comparing images with generic object categories. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2344–2351. IEEE.

Krapac, J. (2011). *Image Representations for Ranking and Classification*. PhD thesis, PhD thesis, Caen University. 77.

Lazebnik, S. and Raginsky, M. (2009). Supervised learning of quantizer codebooks by information loss minimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(7):1294–1309.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE.

Leslie, C., Eskin, E., Weston, J., and Noble, W. S. (2002). Mismatch string kernels for svm protein classification. In *NIPS*, volume 15, pages 1441–1448.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

Li, H. and Jiang, T. (2005). A class of edit kernels for svms to predict translation initiation sites in eukaryotic mrnas. *Journal of Computational Biology*, 12(6):702–718.

Li, J., Wang, J. Z., and Wiederhold, G. (2000). Irm: integrated region matching for image retrieval. In *Proceedings of the eighth ACM international conference on Multimedia*, pages 147–156. ACM.

Li, X., Song, Y., Lu, Y., and Tian, Q. (2011). Spatial pooling for transformation invariant image representation. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1509–1512. ACM.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116.

Liu, L., Wang, L., and Liu, X. (2011). In defense of soft-assignment coding. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2486–2493. IEEE.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Lu, Z. and Ip, H. H.-S. (2009). Image categorization with spatial mismatch kernels. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 397–404. IEEE.

Manjunath, B. S. and Ma, W.-Y. (1996). Texture features for browsing and retrieval of image data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(8):837–842.

Marszaek, M. and Schmid, C. (2006). Spatial weighting for bag-of-features. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2118–2125. IEEE.

Marszalek, M. (2008). Past the limits of bag-of-features.

Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86.

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

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72.

Moosmann, F., Triggs, B., Jurie, F., et al. (2006). Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, volume 2, page 4.

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168. IEEE.

Nowak, E., Jurie, F., and Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *Computer Vision–ECCV 2006*, pages 490–503. Springer.

Perronnin, F. (2008). Universal and adapted vocabularies for generic visual categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1243–1256.

Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer.

Ros, J., Laurent, C., Jolion, J.-M., and Simand, I. (2005). Comparing string representations and distances in a natural images classification task. In *Graph-Based Representations in Pattern Recognition*, pages 72–81. Springer.

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.

Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172.

Sharma, G., Jurie, F., et al. (2011). Learning discriminative spatial representation for image classification. In *British Machine Vision Conference (BMVC)*.

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

Smith, J. R. and Li, C.-S. (1999). Image classification and querying using composite region templates. *Computer Vision and Image Understanding*, 75(1):165–174.

Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International journal of computer vision*, 7(1):11–32.

Tirilly, P., Claveau, V., and Gros, P. (2008). Language modeling for bag-of-visual words image categorization. In *Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 249–258. ACM.

Tsai, W.-H. and Yu, S.-S. (1985). Attributed string matching with merging for shape recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 7(4):453–462.

Tuytelaars, T. (2010). Dense interest points. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2281–2288. IEEE.

Tuytelaars, T. and Mikolajczyk, K. (2008). Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280.

van Gemert, J. C., Veenman, C. J., Smeulders, A. W., and Geusebroek, J.-M. (2010). Visual word ambiguity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1271–1283.

Van Kaick, O. and Mori, G. (2006). Automatic classification of outdoor images by region matching. In *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, pages 9–9. IEEE.

Vedaldi, A., Gulshan, V., Varma, M., and Zisserman, A. (2009). Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613. IEEE.

Viitaniemi, V. and Laaksonen, J. (2010). Region matching techniques for spatial bag of visual words based image category recognition. In *Artificial Neural Networks–ICANN 2010*, pages 531–540. Springer.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. IEEE.

Wu, Z., Huang, Y., Wang, L., and Tan, T. (2013). Spatial graph for image classification. In *Computer Vision–ACCV 2012*, pages 716–729. Springer.

Xu, D., Cham, T.-J., Yan, S., and Chang, S.-F. (2008). Near duplicate image identification with patially aligned pyramid matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE.

Yan, Y., Tian, X., Yang, L., Lu, Y., and Li, H. (2013). Semantic-spatial matching for image classification. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6. IEEE.

Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE.

Yang, Y. and Newsam, S. (2011). Spatial pyramid co-occurrence for image classification. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1465–1472. IEEE.

Yeh, M.-C. and Cheng, K.-T. (2008). A string matching approach for visual retrieval and classification. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 52–58. ACM.

Zhang, H., Berg, A. C., Maire, M., and Malik, J. (2006). Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2126–2136. IEEE.

Zhou, X., Yu, K., Zhang, T., and Huang, T. S. (2010). Image classification using super-vector coding of local image descriptors. In *Computer Vision–ECCV 2010*, pages 141–154. Springer.