VISUAL DETECTION AND RECOGNITION USING LOCAL FEATURES

BY

MERT DIKMEN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Thomas S. Huang, Chair
Professor Narendra Ahuja
Assistant Professor Derek W. Hoiem
Professor Sanjay J. Patel

ABSTRACT

Detection and recognition of objects in images is one of the most important problems in computer vision. In this thesis we adhere to a traditional bottom–up detection and recognition framework, where the objects are first localized with a sliding window detector before being identified. We make multiple contributions along this path. All of the contributions pertain to the central theme of local image features.

We demonstrate improved object detection performance with our proposed feature extraction process, which generalizes the traditional feature extraction methodology of pooling atomic appearance information (e.g., image gradients) around pixels in localized histograms. In addition, we propose a method to fuse two types of information sources in a locally discriminative manner by leveraging local class-dependent correlations.

For the recognition task, we adopt a state–of–the–art metric learning method and modify it to handle unknown identities.

Lastly, the computational improvements achieved through leveraging parallelism are brought together by the Vision Video Library (ViVid), which we release as open source to the research community.

To my parents

ACKNOWLEDGMENTS

First and foremost I owe my deepest gratitude to Professor Thomas S. Huang for his never ending support and encouragement through my achievements and setbacks. The lessons I will be taking from his mentoring go beyond his scientific insight and knowledge. It is a privilege to have him as a personal role model in all aspects of my life.

I would also like to thank my thesis committee, Professors Ahuja, Hoiem and Patel, for their valuable insight and guidance in steering this thesis in the right direction. Professor Hoiem's intuition was spot–on, identifying the most important issues in the new feature representation proposed in Chapter 2. Professor Ahuja's support has resulted in a fruitful collaboration with his student Emre Akbas, which has materialized as a valuable publication that is outlined in Chapter 4. Professor Patel's continued support for my involvement with the Universal Parallel Computing Research Center has opened doors for a completely new dimension to my development and allowed me to learn a skill set which is becoming more relevant with the introduction of new technologies.

A very important part of my experience in graduate school has been the interactions with my collaborators at IFP: Shen-Fu Tsai, Liang Liang Cao, Jianchao Yang, Andrey Del Pozo and especially Dennis J. Lin, who set up the foundations of the ViVid framework and helped selflessly whenever I needed.

Finally I am grateful to my personal friends Ayhan Gunaydin, Ozgur Mizrak, and Bernard Ghanem, who always offered their camaraderie, and last but not least, Arthi Jayaraman, who has been my most special friend throughout.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Unit
HOG	Histograms of Oriented Gradients
LBP	Local Binary Patterns
PAD	Patch Appearance Dictionary
SVM	Support Vector Machine
ViVid	Vision Video Processing Library

CHAPTER 1

INTRODUCTION

Representing visual information is one of the fundamental problems of computer vision. Generally the choice of representation is a significant factor affecting the practical performance of algorithms. In recent years, several types of image representations have emerged as popular choices in the computer vision research community; these include, Gabor filters [1], Local Binary Patterns (LBP [2]), and image gradients [3].

Perhaps the most well known image descriptor is the Scale Invariant Feature Transform (SIFT [3]), a two–step feature representation based on finding affine and scale invariant key-points in an image and locally pooling the gradient information around them. The practical success of SIFT has prompted many variants of the local gradient pooling approach. Histograms of Oriented Gradients (HOG [4]) descriptor is a variant of SIFT that does not use affine invariant interest points. Instead, HOG computes histograms of gradient orientations within each rectangle on a dense, regular grid in order to capture locality information, which results in a very good performance for detecting rigid objects. Gradient Location and Orientation Histogram (GLOH [5]) makes use of log polar bins to introduce better description of the content surrounding the interest region. Another variant is the PCA-SIFT [6] descriptor, which projects local gradient information of a square patch onto a low dimensional subspace spanned by the eigenvectors corresponding to the principal components of highest value.

The widespread adoption of gradient information has several justifications. Gradients are invariant to constant shifts in grayscale values around the pixel. Further, evidence from investigating the visual system of primates and humans revealed the presence of neurons that are highly sensitive to register gradients of specific orientations and locations in the visual field [7]. It is suggested that this local pooling of gradient information is an important contributor to affine invariant recognition properties of our visual systems

[8]. In this thesis we ask the question whether a set of local descriptors with more descriptive information than image gradients can be found. We build a representation, whose atomic building blocks are grayscale invariant and normalized patch exemplars. Similar to gradient filters, these patch exemplars have very small support and have zero mean, preserving the affine and constant illumination invariance properties of the gradient filters. We test our proposed representation in an object detection setting and report favorable results in challenging datasets.

It is probably impossible to find an image representation that suits all needs [9]. As demonstrated in large scale visual classification challenges such as ImageNet [10] and Pascal–VOC [11], combined utilization of different feature modalities yields superior results. In Chapter 3, within a sliding window classifier object detection setting, we investigate the correlations of descriptors that are from different modalities but are spatially close within the sliding window. We show that by finding simultaneous projection directions in which the projected feature descriptors obtain an ordering that respects the class label, we can produce an augmented feature representation that improves over the performance of naive concatenation of features with different modalities.

In Chapter 4, we approach the within–category recognition problem in the context of recognizing pedestrians. Through adopting a metric learning framework, we demonstrate good within–category recognition performance using local feature representations. One of our proposed innovations replaces the local penalty in the cost function of metric learning with a uniform penalty over the sample set. This allows us to learn a metric that is suitable for making rejection decisions on queries that do not have a match in the target dataset.

The proposed feature representation in Chapter 2 increases discriminative performance at the cost of additional computational load. Fortunately, the computational pattern of the feature representation consists of a lot of arithmetic operations with high data locality. This pattern maps to the strengths of graphical processing units (GPUs) very favorably. In Chapter 5, we describe GPU algorithms that yield a generous increase in performance with respect to the CPU algorithms.

CHAPTER 2

A LEARNING FRAMEWORK FOR LOCAL FEATURES

2.1 Introduction

Carefully engineered, gradient-based patch descriptors, such as HOG [12] and SIFT [3], have become a staple of computer vision algorithms. Object detection, image classification, registration, and many other applications benefit from local descriptors that enable robust correspondence. Due to their importance, much research has gone into exploring variations on the feature representations, normalization, and pooling of HOG and SIFT. Most efforts take the simple gradient as the basic building block.

This work demonstrates that replacing gradient filters with a set of more general, learned filters leads to major improvement in object detection and interest point matching. The filters are small (i.e., 3×3) which makes them robust against affine transformations. Furthermore they are constrained to be zero–mean and unit–norm, encoding the intuition that contrast is most informative. The filters are learned by the proposed K–medoids clustering method with a cosine similarity (or absolute cosine) on $d \times d$ patches that are sampled with a bias favoring high-contrast patches. Local descriptors are created by summing the filter responses within a small cell (e.g., 8×8) and applying $L1$–sqrt normalization. Our experiments support the importance of these details, and we validate the general utility of our descriptor on several datasets. On INRIA–Pedestrian, our descriptors outperform HOG with a 50% reduction in miss rate. By replacing HOG with our descriptor in the latest Felzenszwalb et al. detectors [13], we improve results in PASCAL VOC 2007 for 15 out of 20 categories.

A key advantage of the proposed descriptor is that it can be directly integrated into existing learning frameworks using similar block descriptors such as [14] or multilevel pyramid–like representations (e.g., [15], [16]).

In this chapter we will give a brief background of relevant image representations, followed by a description of the filter learning process and an explanation of the relation to HOG and SIFT. We will briefly touch on the implementation details for efficient computation, which will be expanded in Chapter 5. The experiments illustrate how to apply our representation to object detection and interest point matching. We validate the design decisions and demonstrate state-of-the-art performance on several datasets. Finally, we conclude with a discussion of directions for further evaluation and development.

2.2 Background

Efficient and robust representations of visual data are of major interest in vision research and generally are one of the most important factors defining the ultimate performance of algorithms. Simple wavelet–like filters, which have shown very good performance in face [17] and pedestrian detection [18], are one of the early examples of descriptors with both efficient computation and high discriminative performance. When the contours of the object can be successfully extracted, as with Shape Contexts of Belongie et al. [19], a histogram of edge points, with log polar bins around the center of the object, have shown good performance in matching and recognition. Ahonen et al. [2] proposed binary representations of intensity changes around pixels as a representation of local texture. The so–called Local Binary Patterns are especially useful in applications where the texture is the main source of information.

The most relevant type of representation to this work is the histogramming of gradient orientations, which is inspired by Scale Invariant Feature Transform (SIFT [3]), which was originally intended for resolving the problem of keypoint correspondence. Many variants, such as PCA–SIFT [6] and Speed Up Robust Features [20], have been proposed, improving on computational efficiency. Some modifications to the binning strategy of SIFT have demonstrated better affine and rotation invariance properties (e.g., Daisy descriptor [21]). Furthermore, densely sampled variants of SIFT with no alignment for orientation have been proven very useful for detection of objects with reasonably rigid part configurations [12, 22].

Recently, there has been interest in optimizing feature representations through learning. LeCun et al. [23] have pioneered use of convolutional neural networks for bottom–up learning of object detectors, where the first learned layer consists of patch level filters. Dictionary based image representations can also be improved through iterative subgradient methods [24] or through sparse coding [25]. However, such dictionary optimization methods are difficult to apply in low level representations because of non–convexities introduced through the use of various heuristics such as block-normalization of histograms, max-clipping of bins and weighted histogramming. Furthermore, learning based methods tend to yield object or task specific representations, whereas the representation proposed in this work is general.

2.3 Overview

We propose a new feature transformation for encoding local blocks of image information for discriminative purposes. Inspired by block-descriptors such as HOG and SIFT, multiple local histograms of appearance information are grouped and normalized together to form a local descriptor. The key difference from the aforementioned feature transformations is that the locally pooled appearance information is not orientation histograms, but rather a less restricted set of "snippets" of appearance, which we will call the patch appearance dictionary (PAD).

2.3.1 Patch Appearance Dictionary

We propose to replace the gradient orientations with general filters, which can be directly extracted from empirical data and thus can better capture the statistical properties of the underlying visual structures.

The first step is to define a vocabulary of image patches through clustering. Staying faithful to gradient based image representations, we define the similarity measure in the space of $d \times d$ image patches to be the dot product of vectorized representations of the image patches:

$$s(\mathbf{p}_i, \mathbf{p}_j) = \mathbf{p}_i^T \mathbf{p}_j, \tag{2.1}$$

where i and j denote the pixel indexes and $\mathbf{p}$ is the vectorized representation of a $d \times d$ patch.

Dot product as the similarity measure is frequently utilized in the document retrieval research under the term "cosine similarity" and is tied naturally to the popular variant of the k-means algorithm known as spherical k-means [26], which groups points based on their cosine similarity. This method is different from regular k-means because the underlying probability distribution of the points is no longer assumed to be mixture of k unit variance Gaussians, and furthermore, as the name implies, spherical k-means produces clusters on the unit hypersphere.

However, spherical k-means cannot be applied directly to build representations of local image patches, because unlike word frequency histograms used in document retrieval, convolving the clustered patches with an image does not necessarily produce positive or 0 responses. Hence the cosine similarity is not guaranteed to be positive in all cases. This difference in data domain does not affect the convergence of the spherical k-means clustering algorithm because the objective function is still bounded, but presents the practical question whether an image patch is more similar to a patch with 0 or a small positive correlation value than a patch with a very high negative correlation (e.g., its contrast negative). Another way of thinking about this is whether one prefers the representation to be contrast sensitive or insensitive. In the former case, cosine similarity shall be used, while for the latter taking the absolute value of the cosine similarity is appropriate. Using the absolute cosine similarity is equivalent to assuming that for gradients with the same orientation, the direction of dark-to-bright transition is not important (Figure 2.1). In the case of pedestrian detection, Dalal and Triggs [12] have shown that the direction of the contrast change is not relevant in detection of objects with widely varying appearance such as pedestrians standing against arbitrary backgrounds. Unless otherwise noted, we will assume the use of absolute cosine similarity in the rest of the thesis.

In order to cluster image patches with absolute cosine similarity, we propose a modified spherical k-means algorithm using sample medoids as opposed to sample means. The spherical k-medoids algorithm is summarized in Algorithm 1. The main motivation for using medoids instead of means is the lack of definition for a meaningful center of mass of points when using cosine similarity (Figure 2.2). However an additional advantage is obtained

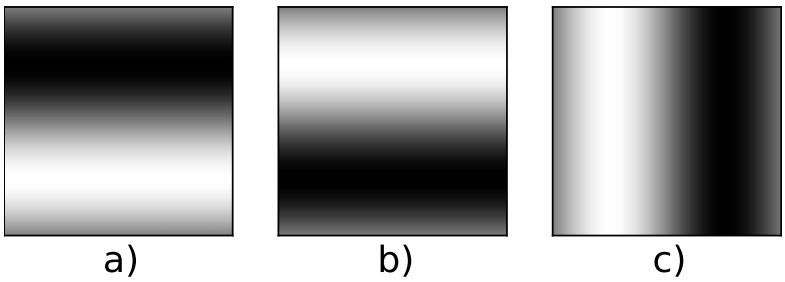


Figure 2.1: Illustrated are three 128×128 grayscale patches. Patch (b) is the negative of patch (a) and patch (c) is the 90° rotated version of (b). All patches are normalized to have magnitude equal to 1.0. The cosine similarity between patches (a) and (b) is -1.0, while absolute cosine similarity between them is 1.0. Both of their cosine similarities to patch (c) are 0.

in terms of speed. Since the solution has to be a subset of initial points, the local minima are quite stable. This leads to quick convergence in practice. In our experience, we have found that the algorithm with 1 million initial points quickly converges to a local minimum within 10–20 iterations regardless of k.

Algorithm 1 spherical k-medoids clustering

Input: Set of exemplar points with unit norm to be clustered: $\chi = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$
Initialize K cluster centers $M = \{\mathbf{m}_1, \ldots, \mathbf{m}_k\}$ by selecting K samples from the set χ [27].
Initialize clustering fitness: $C = 0$
while $\triangle C > 0$ **do**
  For each $\mathbf{x}_i$ set $y_i = \texttt{argmax}_k \mid \mathbf{x}_i^T \mathbf{m}_k \mid$
  Update each cluster k:
  $\mathbf{m}_k \leftarrow \texttt{argmax}_{\mathbf{x}_i} \sum_j I(k - y_j) \mid \mathbf{x}_i^T \mathbf{x}_j \mid$
  $C \leftarrow \sum_i \sum_k I(k - y_i) \mid \mathbf{x}_i^T \mathbf{m}_k \mid$
end while

Similar to the well known k-means algorithm, the first step in the outer loop associates each exemplar with the maximally similar cluster center. In the second step, for each cluster, the exemplar that is most similar to all other exemplars assigned to the same cluster is chosen as the new cluster center. Again, similar to k-means, k-medoids is also sensitive to initialization. In order to remedy the effect of bad initialization, we pick the initial k centers using a maximum dissimilarity criterion analogous to the k-means++ method

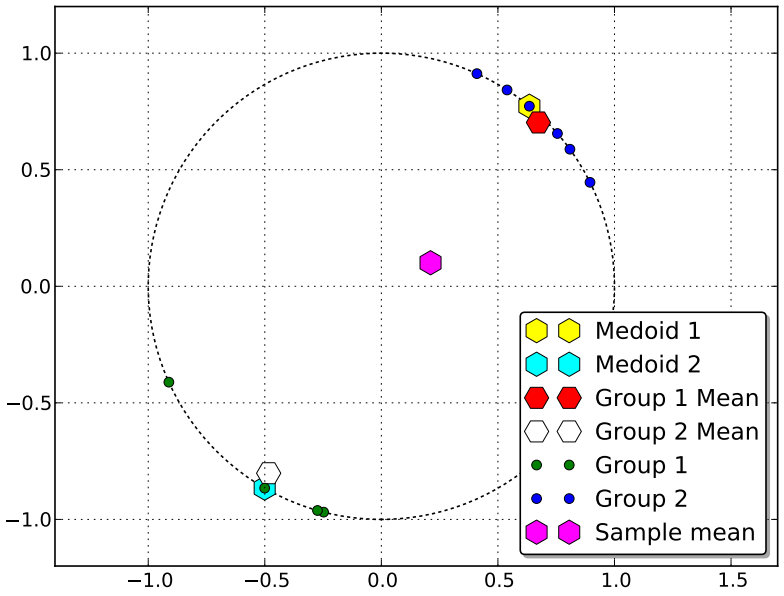


Figure 2.2: Presented are two groups of points on the unit circle (Group 1 (green) and Group 2 (blue)). According to absolute cosine similarity, they can be seen as forming one tight cluster, whereas under cosine similarity there would be two clusters. In both cases, however, means of the points will not lie on the unit circle and may end up further away from both clusters when projected back on to the unit sphere. Medoids provide a more stable alternative for determining exemplars for each clusters.

[27].

The training set for dictionary learning is collected from a set of natural, grayscale images. In our experiments we limited the size of this collection of patches to be around one million, which seems to produce clusters with good variety. Appearance prior of image patches in natural images is not uniform. Smooth structures such as uniform patches or ramp discontinuities comprise most types of patches. Unfortunately, from a discriminative standpoint such patches are rarely interesting. Patches at shape discontinuities with sharp edge content are far more useful from a discriminative standpoint. Thus for dictionary learning, we perform a biased sampling of image patches to form the set of exemplars. The probability of the patch being sampled as an exemplar is proportional to the strength of the pixel intensity contrast within the patch:

$$P_{\texttt{sampling}}^{(p_i)} \propto \|\mathbf{p}_i - \mu(p_i)\|\,, \tag{2.2}$$

where $\mu(p_i)$ denotes the average pixel intensity within the patch $\mathbf{p}_i$.

The modeling properties of the learned dictionaries can be readily observed in Figure 2.3. With a learned dictionary size of 100 codewords (see Figures 2.4 and 2.5 for an example visualization), only less than 10% of randomly sampled patches do not have a cluster center, whose dot product with the patch is at least 80% of the patch magnitude. Also, the dictionary trained with bias–sampled data is slightly better at modeling than the dictionary trained with uniformly sampled patches. The effect of the biased sampling becomes more apparent in Figure 2.6, where the same experiment in Figure 2.3 is repeated on bias–sampled data.

The modest gap of 100 item dictionaries with the HOG dictionary (Section 2.3.2) is not surprising because, as the number of dictionary items approaches infinity, all patches should have an arbitrarily close dictionary neighbor. However, even a trained dictionary of 9 items, where the size is equal to the HOG dictionary, is still able to demonstrate significantly better modeling performance.

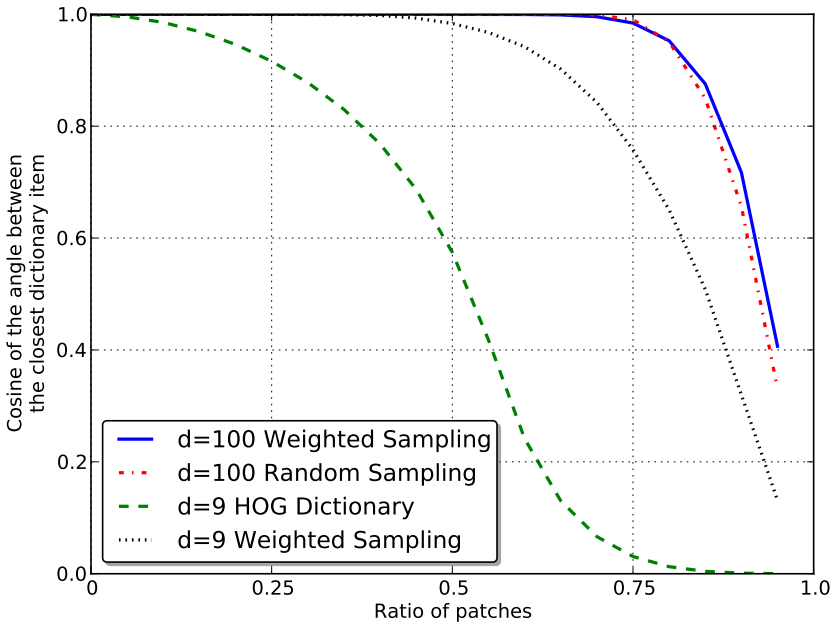


Figure 2.3: Visualization of patch modeling accuracy of representations. Each curve shows the percentage of randomly sampled patches for which there is at least one dictionary item, whose cosine similarity is at least the value on the vertical axis.

Figure 2.4: Visualization of a patch dictionary of 3×3 patches learned using spherical k-medoids. The dictionary contains 100 elements and the elements are ordered by the frequency with which appear in the training set, with the most frequent dictionary item being on the top left and the least frequent on the bottom right. The most frequent patches appear to be horizontal and vertical boundary segments followed by diagonal boundaries and line-like shapes. Corner–like and point–like segments make up the rest of the words.

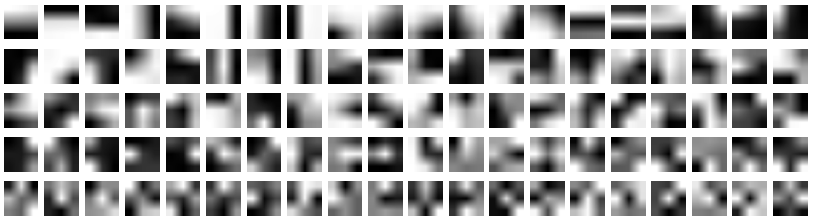


Figure 2.5: Interpolated versions of patches in Figure 2.4 (for viewing convenience).

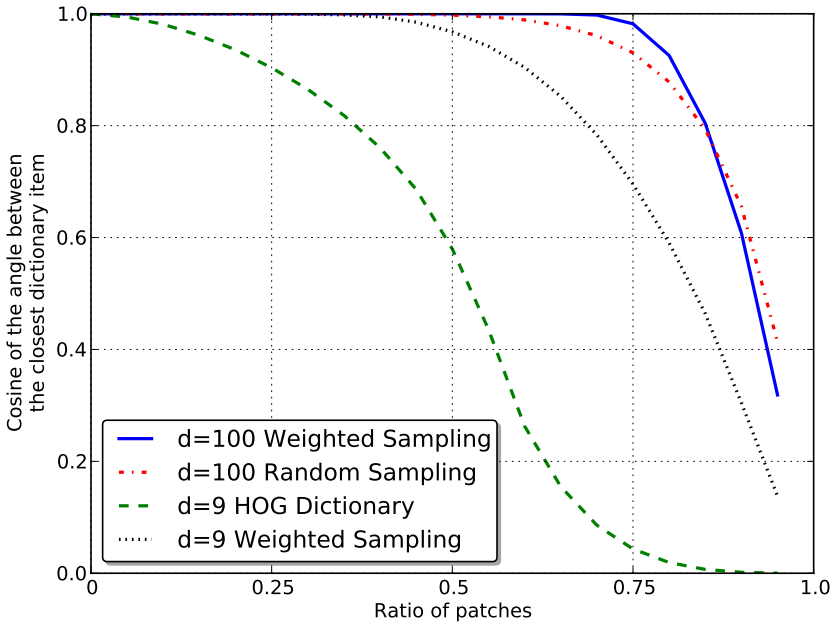


Figure 2.6: Same visualization as in Figure 2.3, except the modeling accuracy is measured on the set of patches which are sampled with bias (i.e., high contrast patches). The dictionary trained on biased data has the better modeling power.

2.3.2 Connection to Gradient Orientations

Gradient histogram based feature transformations can be thought of as dictionary based representations of local image patches. To illustrate this, note that image gradient orientations are computed through measuring the responses of two filters: h_x and h_y corresponding to horizontal and vertical gradient filters. Equation 2.3 shows an example pair of commonly used gradient filters.

$$h_y = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, h_x = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.3}$$

Let $h_x^{(\mathbf{p}_j)}$ be the response of the filter h_x centered at the pixel $\mathbf{p}_j$. The gradient orientation for the interval $[-90°, 90°]$ at pixel $\mathbf{p}_j$ is given by:

$$\tan^{-1}\left(h_y^{(\mathbf{p}_j)} / h_x^{(\mathbf{p}_j)}\right). \tag{2.4}$$

Quantizing this orientation information is equivalent to finding the maximally responding filter from the following set:

$$h_n = \begin{bmatrix} 0 & -\sin(\theta_n) & 0 \\ -\cos(\theta_n) & 0 & \cos(\theta_n) \\ 0 & \sin(\theta_n) & 0 \end{bmatrix} \tag{2.5}$$

$$\theta_n = \frac{180° \times n}{N+1}, n \in \{1, \ldots, N\}, \tag{2.6}$$

where N is the number of quantization levels.

The histogram descriptor of one cell can be expressed as:

$$f_n = \sum_{\mathbf{p}_j \in c_l} g(\mathbf{p}_j, \mathbf{h}_n) \tag{2.7}$$

$$g(\mathbf{p}_j, \mathbf{h}_n) = \begin{cases} (\mathbf{p}_j^T \mathbf{h}_n)^2, & \text{if } n = \text{argmax}_m((\mathbf{p}_j^T \mathbf{h}_m)^2) \\ 0 & else. \end{cases} \tag{2.8}$$

Note that this is essentially a histogram based representation, by doing a locally constant estimation of 3×3 patch appearances. According to the

default feature parameters, the appearance constant in the case of SIFT is 8 and for HOG it is 9, arguably small numbers for piecewise constant modeling of a 9-dimensional signal. Further, since they all have the structure in Equation 2.5, they are placed on a one–dimensional manifold (parameterized by θ) on $\Re^9$.

When studying SIFT and HOG closely, a general pattern of feature extraction emerges. The first step usually involves extraction of some low level image information, which is believed to be most effective in solving the particular vision task. In the next step this local information is spatially grouped through a local histogramming process, which assigns these patches of local information to a pre–defined vocabulary. As a last step, one frequently normalizes local groups of these histograms. This process is useful for histograms built using the magnitudes of local information patches as weights because the normalization process ameliorates feature magnitude variations due to changes in local contrast.

A common feature of the gradient orientation filters is that they are 0-mean and they have unit norm. The unit norm property ensures that no filter is biased in building the histogram, since the square correlation is used as the similarity measure and the 0-mean property has biological justification since it is well known that the human visual system is more sensitive to local changes in the contrast than the absolute brightness of the signal [7].

2.4 Block Descriptor

The construction of the proposed descriptor is exactly the same as the construction of widely used histogram based descriptors. The atomic units of information for the descriptor are the "bag of word"–like histograms of dictionary similarities for each pixel in a square group of pixels called cells. Each block descriptor is the concatenation of $c \times c$ neighboring but disjoint cell histograms. The location of these blocks can be defined on a dense rectangular grid for rigid object detection applications, or they can be constructed around interest points produced by an affine interest point detector.

To construct the descriptor for an image block, the image is first convolved with all elements of the dictionary. Because of the 0-mean property, the elements of the dictionary can be used directly as filters and it is not necessary

to subtract the mean from each patch. The image block region contains $c \times c$ cells. For each pixel in these cells, the dictionary element with highest similarity value is found and a weighted vote equal to this similarity value is accumulated on the histogram bin on the corresponding cell. After concatenation of cell histograms in each block, the block descriptors are normalized with respect to an appropriate norm (the selection of this norm is further discussed in Section 2.5). The cell histograms are computed by pooling the information from $p \times p$ pixels. In the case of SIFT, $c = 4$ and $p = 4$, whereas the original HOG paper sets $c = 2$ and $p = 8$ for optimum performance on the INRIA–Pedestrian dataset.

2.4.1 Computation and Memory

Dictionary sizes on the order of 100 can produce seemingly high–dimensional representations. However the non-zero entries in the cell histograms are upper–bounded by the number of sampled pixels in each of the cells. Therefore, using a sparse vector representation yields a maximum of $2 \times N$ units of memory footprint per cell histogram, independent of the dictionary size.

GPU

Computation of the filter responses for each of the dictionary items can be performed in parallel very efficiently using a GPU. Straightforward convolution is the preferred method for convolutions with small convolution kernels up to 7 pixels wide [28]. Our CUDA version of the filterbank code performs 100 convolutions and reductions in approximately 25 ms on a GeForce GTX 560 for a 1000×1000 image.

2.5 Evaluation

We test the performance of our descriptor on the INRIA–Pedestrian dataset as well as on the 2007 dataset of the Pascal VOC.

INRIA–Pedestrian

The INRIA–Pedestrian dataset contains 1208 training images of pedestrians with their reflections along the vertical median axis. In addition, 566 images of pedestrians and their axis reflections are provided for testing. The dataset also contains negative training and testing images that do not contain any people.

For training, we extract the block descriptors based on concatenation of 2×2 cell histograms on a dense grid of 8×8 pixels inside a 128×64 window centered around the pedestrian images. This constitutes our positive training set consisting of 105 block descriptors of dimensions $2 \times 2 \times d$, where d is the size of the dictionary. Unless otherwise specified we set $d = 100$. Negative features are collected from random 128×64 subimages from the negative training set, which does not contain any images with people in them. We learn the initial dictionaries from a set of one million 3×3 image patches randomly collected from the training set. First a support vector machine classifier is learned with linear kernel. Then this classifier is used to densely scan the negative training set to look for "difficult" samples, which are falsely classified as pedestrians by this first stage classifier. A second stage classifier is then trained using the initial training set with the addition of all difficult samples detected by the first classifier. All of the SVM training is performed through LIBLINEAR [29], which we have slightly modified to increase memory efficiency tailored to take advantage of the sparse nature of our proposed representation.

We verify the pedestrian detector by running it on 566 pedestrians (and their mirrors) on the testing set, as well as all image windows of size 128×64 on the negative testing set at the original scales of the images and down-scaled versions by a scaling step of 1.2 until a no object window can fit. The window stride length is 8 both in horizontal and vertical directions, which yields 2 million image windows with no pedestrians for testing. The results are reported in the form of detection error tradeoff curves, where the logarithmically scaled x-axis shows the false positives per image window on the negative set versus the y-axis which plots the miss rate (1−true positive). The first experiment (Figure 2.7) shows the effectiveness of the proposed descriptor versus the standard HOG, which is implemented verbatim to the description in [12].

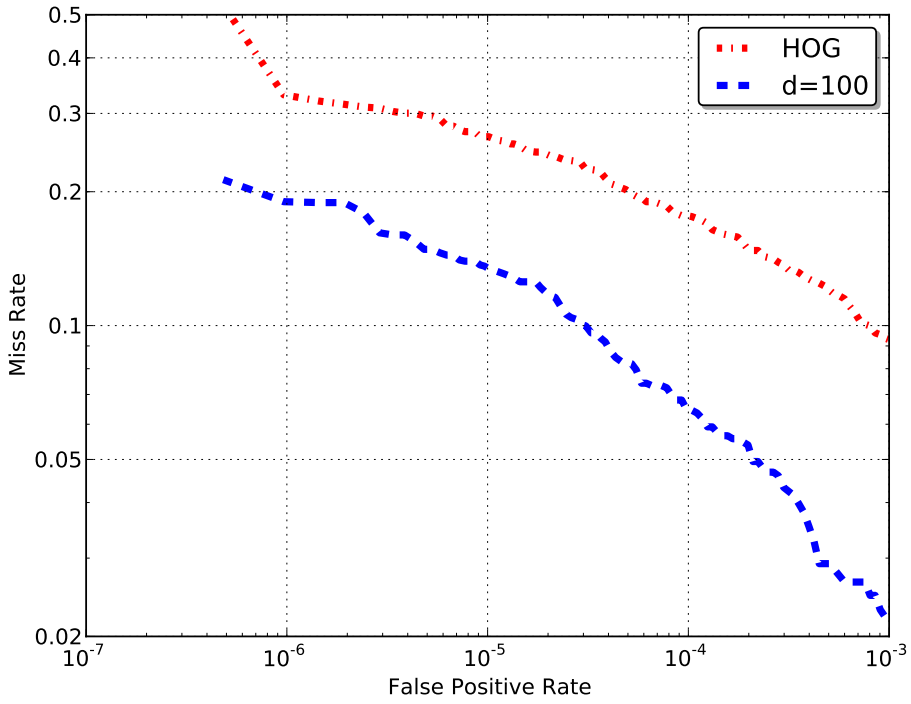


Figure 2.7: The proposed descriptor yields a false positive rate of 10^{-4} false positives per window at 6.5% miss rate.

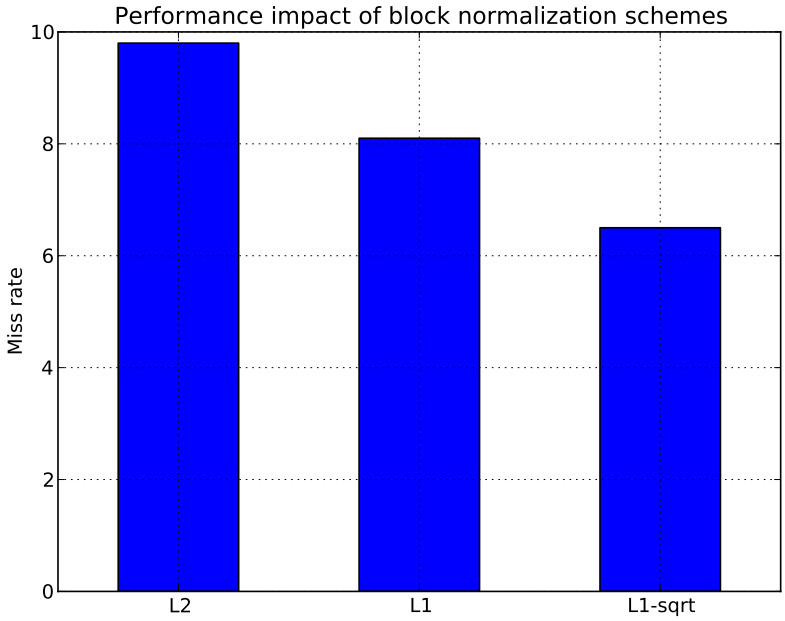


Figure 2.8: Normalization of cell histograms within each block is essential for optimum performance. Since our histogram is large, a gentle norm like $L1 - sqrt$ works better.

The normalization of the concatenated histograms in a block has a profound effect on the performance. While authors in [30] found $L2$ normalization to work the best on the normalization of the gradient orientation histograms, $L1$–$sqrt$ norm works significantly better in our case (Figure 2.8). This can be explained by the high–dimensional structure and sparse nature of our cell histograms. Normalizing with respect to the $L2$ norm reduces the small values in the histogram too aggressively.

For the effects of the dictionary size on the overall detection performance, conventional wisdom from bag-of-words research carries over, and is also reflected in our results, where larger dictionaries outperform smaller ones for what is otherwise the same parametrization of the feature. The automatically learned dictionary of about 9 words narrowly beats the HOG baseline, which uses a manually constructed dictionary of 9 items long; however as the dictionary size increases, the proposed descriptor becomes more discriminative. Figure 2.9 shows the miss rate at 10^{-4} false positives per window rate for varying dictionary sizes. The returns start diminishing after 200 words and, furthermore, increased computation becomes another consideration for

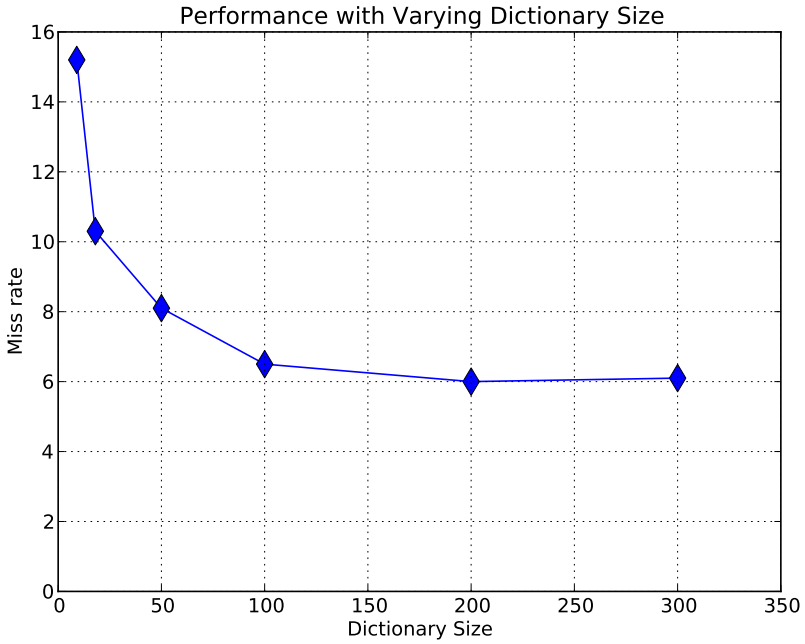


Figure 2.9: The miss rate drops as larger dictionaries are utilized. However the computational cost also increases linearly for histogram computation as d convolutions are required for each feature transformation.

trade–off.

The size of the sampled patches also has a noticeable effect on the performance. We tested 3×3, 5×5 and 7×7 patch sizes for training dictionaries. For a constant dictionary size of 100 and all other feature parameters kept constant, in terms of the miss rate at 10^{-4} false positives per window operating point, the 5×5 patch dictionary performed 2.53% worse than the dictionary of 3×3 patches, whereas the 7×7 dictionary was the worst with 7.67% increase in miss rate over the 3×3 dictionary baseline. These findings suggest that modeling more complex patches can become very difficult very quickly as the patch size increases. Furthermore as the base patch size increases, less and less variety can be captured in terms of local appearance and texture properties.

Finally a visualization of the trained linear SVM classifier can be seen on Figure 2.10. The most positively performing appearance resembles the averaged human pose of the training images with the positive label. Slight variations in pose, such as bent legs, are also successfully captured by the learning stage.

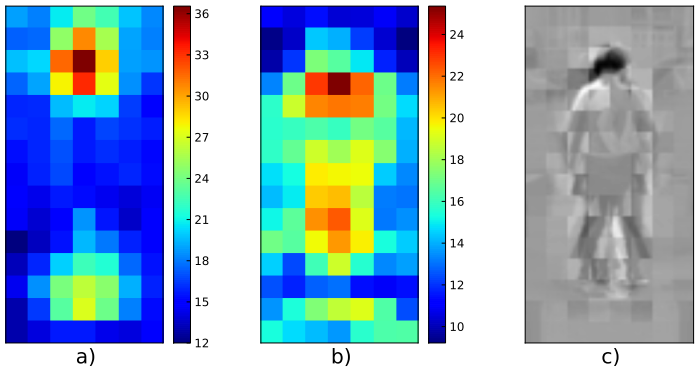


Figure 2.10: A visualization of the classifier trained on the INRIA-Pedestrian dataset. (a) Spatial distribution of the positive classifier weights. (b) Spatial distribution of the negative classifier weights. The most distinguishing features of pedestrians are around the head and shoulders area as well as around the feet. The negative weights are distributed more uniformly, with a slight absence of weights around the detection window boundaries, which are expected to be mostly background regions regardless of the class label of the detection window. (c) The collage of image blocks in the training samples with the highest response to the classifier weights for the corresponding block.

Pascal VOC

Pascal VOC provides a challenging dataset and a good testbed for measuring object detection performance. The authors of [14], which is one of the state–of–the–art object detectors, opened the source-code of their parts based object detector for free use. The original detector uses HOG, sign variant (contrast sensitive) HOG and a texture measure as the base features of their parts detectors. We modified the source code [13] of the parts based detector to operate with the proposed feature transformation instead of the original features. The local support of the atomic histograms was set to be 8×8 pixels, and the dictionary size was 50. For each cell we produced two histograms, one with cosine similarity as the similarity measure and the other with absolute cosine similarity. These contrast sensitive and insensitive representations were finally concatenated to produce the final cell descriptor. All other training parameters were kept fixed. As can be observed in Figure 2.11, our descriptor's performance exceeds or matches the performance of the baseline detector at all but 5 object categories.

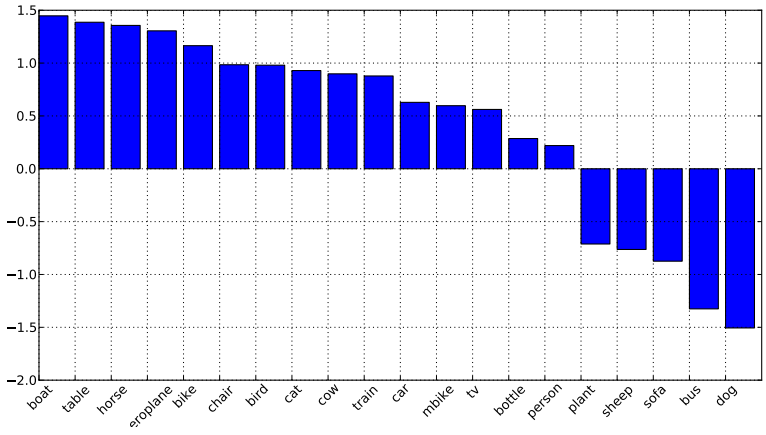


Figure 2.11: Relative performance difference in terms of average precision on the Pascal VOC2007 classes over the baseline detector[13]. Training the parts based detector with the proposed feature set increases the average precision performance over the baseline parts based detector with HOG family of features on most classes in the 2007 Pascal VOC dataset.

2.6 Conclusions

We have described a robust alternative to gradient orientation based image features. Our new proposed feature transformation adopts many of the carefully engineered properties of previous feature transformations (e.g., blockwise contrast normalization, locally constructed histograms), while improving on the power of the unit histograms in representing the underlying image appearance. The proposed method is directly applicable to all existing methods using the aforementioned descriptors through a simple substitution. As it is experimentally demonstrated, the proposed feature transform offers robust performance on object detection tasks. We would like to further investigate whether similar performance gains can be obtained in other areas such as keypoint correspondence problems and representations of spatio-temporal data.

2.7 Future Directions

By noting that the dictionary size is generally much larger than the vectorized patch dimensionality, we propose to take advantage of an efficient binary search structure such as kd-Trees [31]. The kd-Tree is one of the most popular

representational structures for efficient nearest neighbor search. The basic idea is to partition the search space into separate bins, which can be accessed through traversing a binary search tree, whose internal nodes contain simple thresholding tests for a particular component of the query vector. From a feature space standpoint, the root node of the tree partitions the space into two, and the two children corresponding to each of these subspaces from the root partitioning respectively further partition these into two partitions. When a leaf node is encountered during the tree traversal, all of the points in the leaf are compared to the query point and the closest point is determined. However, the closest point found in this binary search is not necessarily the nearest neighbor of the query point. The kd-Tree algorithm with Euclidean distance measure tests for this by checking whether the bin boundary is closer to the query point than the current nearest neighbor, and repeating this test recursively at every internal node visited. If true, the test is repeated for all unvisited children of the node, searching for a closer nearest neighbor than the current. A similar test can be applied with cosine similarity search, where we can define an upper bound on the maximum similarity a sample can have in the unvisited bins. Theoretically, the proposed method can speed up the lookup of the best matching filter from O(N) time to O(log(N)) time with respect to the size of the dictionary. However, the practical performance depends on the distribution of the filters in the feature space, which remains to be experimentally verified.

CHAPTER 3

LEVERAGING LOCAL CORRELATIONS

3.1 Representing Multi–Modal Data

It is essential for the success of a general object detection approach to utilize more than a single visual cue. The usage of multiple modalities has been addressed previously in the literature through methods ranging from naive concatenation to more advanced methods, which take the individual nature of different topologies of feature spaces with different modalities into account. Naive concatenation is the simplest method of feature combination, but can yield good results with good selection of feature modalities. Wang et al. [32] showed a successful combined feature approach for a human detection application, in which they form an augmented descriptor by concatenating HOG and LBP descriptors.

Naive concatenation is oblivious to the fact that different types of features may require different strategies for comparison. For example, with histogram type data, domain specific distance (or similarity) measures such as Bahttacharyya distance [33], Earth Mover's Distance [34], Intersection Kernels [35] and Pyramid Match Kernels [36] are known to produce more robust similarity information than simple L_p norms. With image type data, where location of features is also discriminant, a Spatial Pyramid Kernel [16], which takes rough spatial correspondence information into account, may produce more favorable results. In a support vector machine setting, this can be addressed by combining multiple feature–specific kernels [37, 9, 38].

An unsupervised feature fusion method has been introduced by Fu et al. [39], in which a low–dimensional feature representation is learned from multiple feature sets.

In this chapter we address the problem of feature fusion from a novel standpoint. We reason that at the local level, co-occurrence information between

feature modalities can be leveraged to produce discriminative information. We achieve this by finding canonical correlations of sub-features that have roughly the same spatial support in the image domain. In this work, our contributions are twofold. First, we introduce a novel feature fusion method using a class aware canonical correlation analysis of to extract co-occurrence information of local features from two separate modalities. Furthermore, we show that the generalization properties of the local correlation analysis can be improved with a max-margin formulation.

3.2 Feature Combination through Canonical Correlations

An important issue that arises in the context of multiple features is how to take advantage of meaningful correlations and co-occurrence patterns between two types of seemingly unrelated data. Our approach involves canonical correlation analysis [40], a standard method for finding correlations between two datasets. Starting with feature representations of the same object instance in two different modalities, we would like to reduce these representations onto a real line, where the class label ordering is respected as much as possible (Figure 3.1).

Suppose $\mathbf{X_1}$ and $\mathbf{X_2}$ are $(n \times d_1)$ and $(n \times d_2)$ matrices respectively. Each row represents a single feature descriptor extracted from a given location in the training image (e.g., a specific location in the dense feature grid). In our case these would be Patch Appearance Dictionary features introduced in Chapter 2 and LBP descriptors for blocks, which we describe in Section 3.2.3. Note that the block descriptor is not the full feature but a subset of it with a certain spatial support region. The term n is the number of training images, while d_1 and d_2 are dimensions of the two types of features respectively. For the sake of notational simplicity, assume that both $\mathbf{X_1}$ and $\mathbf{X_2}$ are already centered around the respective row means. The goal is to find two vectors, $\mathbf{z_1}$ and $\mathbf{z_2}$, such that the transformations $\mathbf{X_1 z_1}$ and $\mathbf{X_2 z_2}$ are maximally correlated. Formally we maximize the following expression:

$$\rho = \frac{(\mathbf{X_1 z_1})^T \mathbf{X_2 z_2})}{\|\mathbf{X_1 z_1}\|_2 \, \|\mathbf{X_2 z_2}\|_2}. \tag{3.1}$$

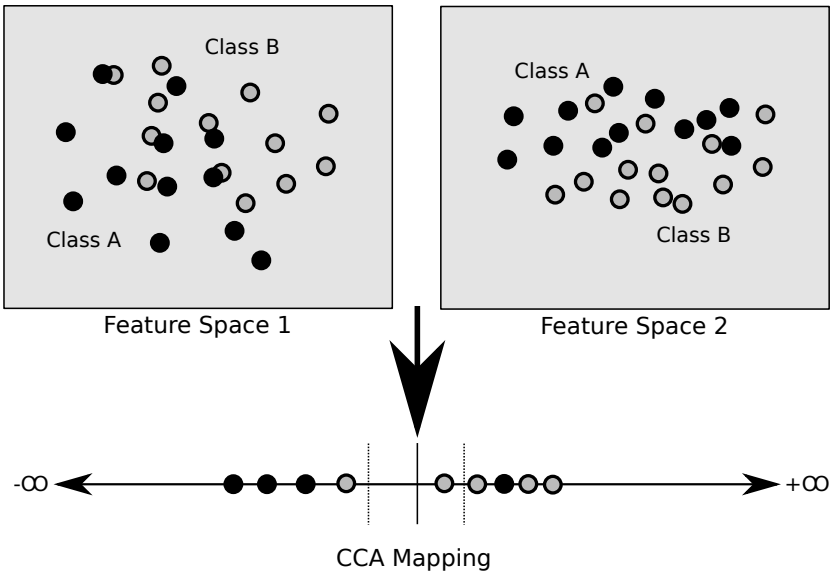


Figure 3.1: The goal of class aware canonical correlation analysis is to find a mapping from two separate feature spaces onto the real line that respects the class ordering.

Scaling any of the two terms in the denominator does not change the correlation value. Therefore, both can be constrained to be equal to unity.

$$\|\mathbf{X_1 z_1}\|_2 = \|\mathbf{X_2 z_2}\|_2 = 1. \tag{3.2}$$

Finally, the Lagrangian for maximizing the correlation is formulated as follows:

$$\begin{aligned} L(\alpha_x, \alpha_y, \mathbf{z_1}, \mathbf{z_2}) =& \mathbf{z_1^T \Sigma_{(1,2)} z_2} - \\ & \frac{\alpha_1}{2}(\mathbf{x_1}^T \mathbf{\Sigma_{(1,1)} x_1}) - \frac{\alpha_2}{2}(\mathbf{x_2}^T \mathbf{\Sigma_{(2,2)} x_2}), \end{aligned} \tag{3.3}$$

where we have defined the $\mathbf{\Sigma}$ matrices as ($\mathbf{\Sigma}_{(i,j)} = \mathbf{X}_i^T \mathbf{X}_j^T$). Evaluating the Kuhn-Tucker conditions on $L(\alpha_1, \alpha_2, \mathbf{z}_1, \mathbf{z}_2)$ yields the generalized eigenvalue problems (Equations 3.4 and 3.5) after a few basic algebraic manipulations outlined in [40].

$$(\mathbf{\Sigma}_{(1,2)}^T \mathbf{\Sigma}_{(1,1)}^{-1} \mathbf{\Sigma}_{(1,2)} - \alpha^2 \mathbf{\Sigma}_{(2,2)}) \mathbf{z}_2 = 0 \tag{3.4}$$

$$(\mathbf{\Sigma}_{(1,2)}^T \mathbf{\Sigma}_{(2,2)}^{-1} \mathbf{\Sigma}_{(1,2)} - \alpha^2 \mathbf{\Sigma}_{(1,1)}) \mathbf{z}_1 = 0 \tag{3.5}$$

which can be solved in closed form for $\mathbf{z}_1$ and $\mathbf{z}_2$ respectively.

3.2.1 Augmented Feature Representation through CCA

For discriminative analysis it is not enough that the two different features are transformed to be maximally correlated. It is also necessary that the samples from two different classes get mapped to the different extremes of the correlation range. This means features from samples in one class get mapped to achieve maximum positive correlation values, whereas features from the other class get mapped to achieve a maximally negative correlation value. Thus we introduce a weak discriminative criterion to the canonical correlation analysis through a simple sign change when building the feature matrices $\mathbf{X}_1$ and $\mathbf{X}_2$, namely:

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{X}_1^{(class_1)} \\ \mathbf{X}_1^{(class_2)} \end{bmatrix} \mathbf{X}_2 = \begin{bmatrix} \mathbf{X}_2^{(class_1)} \\ -\mathbf{X}_2^{(class_2)} \end{bmatrix}. \tag{3.6}$$

Note that when the number of samples from two classes are unbalanced, this formulation is heavily biased towards the dominant class. This is the case in many object detection problems involving the sliding window approach (labeling is expensive, negatives are cheap). To alleviate this fact, we only use a subset of the samples from the negative class. To get a good representation of difficult cases, we collect the negative samples that are identified as false positives after the first round of classifier training with PAD+LBP (PAD and LBP concatenated) features. The details of the hard negative sample harvesting are discussed in Section 3.3.

At the training stage we compute the pairwise canonical correlation analysis bases separately for all pairs of PAD and LBP block descriptors that are extracted from the same blocks in the image sub-rectangle. Pairwise correlations of the transformed block descriptors and their pairwise products with each other are used to form a new feature vector, which is then augmented to the original PAD+LBP feature vector.

3.2.2 Max-Margin Canonical Correlations

Although an effective method, the CCA representation proposed in Section 3.2.1 has no protection against over-fitting to the training data and therefore the generalization performance may suffer. While the training data obtains a clear separation of classes in almost every block, this separation is much less pronounced in the testing set. We propose a new formulation for extracting meaningful feature correlations by combining two powerful ideas: canonical correlation analysis and the maximum margin principle. The maximum margin principle stems from the intuition that when there are infinite separators for two given separable clouds of points (Figure 3.2), the separator, which maximizes the margin between the two point clouds (Figure 3.3), shall have the optimum generalization properties. This intuition is tightly connected to structural risk minimization [41]. Although a rigorous justification has not been worked out [42], empirical evidence from many studies suggests that maximum margin separators have very favorable generalization performance. To incorporate the max–margin criterion into the canonical correlation analysis setting, we propose the following cost function:

$$\min_{\mathbf{z}_1,\mathbf{z}_2} \|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 + \lambda \sum_i [1 - y_i(\mathbf{x}_i^{(1)T}\mathbf{z}_1)(\mathbf{x}_i^{(2)T}\mathbf{z}_2)], \tag{3.7}$$

where $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_i^{(2)}$ are the two feature vectors from the first and second feature modalities, i denotes the sample index, y_i denotes the sample label, and $\mathbf{z}_1$ and $\mathbf{z}_2$ are the CCA projection directions. The cost function is non–convex in $\mathbf{z}_1$ and $\mathbf{z}_2$; however, it can iteratively converge to a local minimum if $\mathbf{z}_1$ and $\mathbf{z}_2$ are fixed in an alternating fashion. When either projection direction is fixed, it is reduced to a standard quadratic programming form, which can be efficiently solved by any available SVM solver. The convergence to a local minimum is guaranteed because Equation 3.7 is lower–bounded by 0 and alternating iterations monotonically reduce the cost.

To illustrate the effectiveness of max-margin formulation we compute the CCA features on positive samples and randomly collected negative samples on the testing set for both max-margin CCA and the CCA method from the previous section. For each of the local blocks where the CCA features are computed, we model the features computed on the positive and negative classes with Gaussian distributions and measure the symmetric KL–

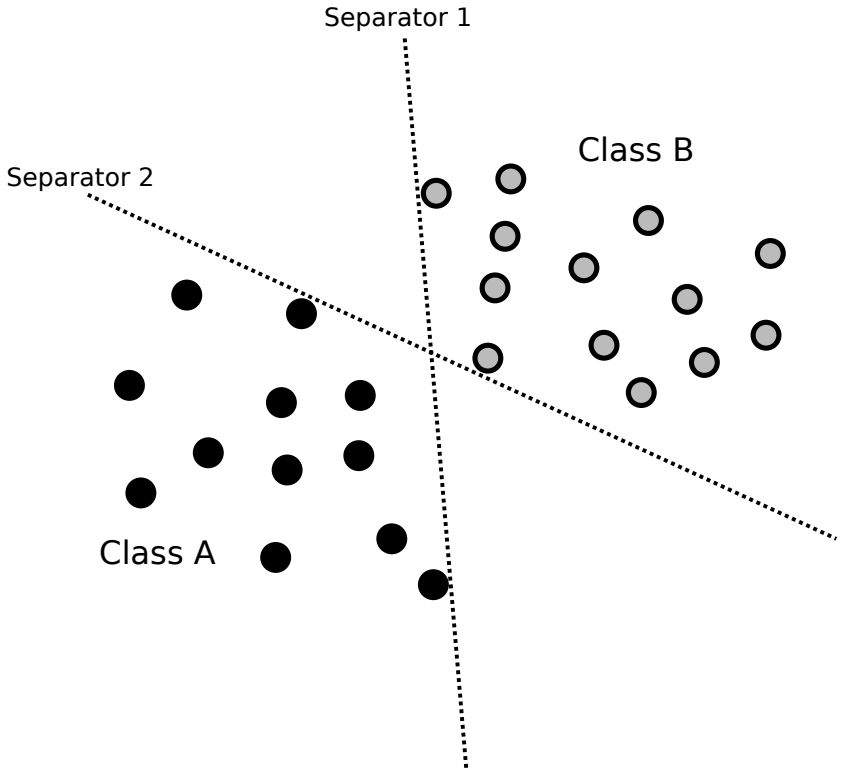


Figure 3.2: For perfectly separable classes A and B, there are infinitely many separators between Separator 1 and Separator 2.

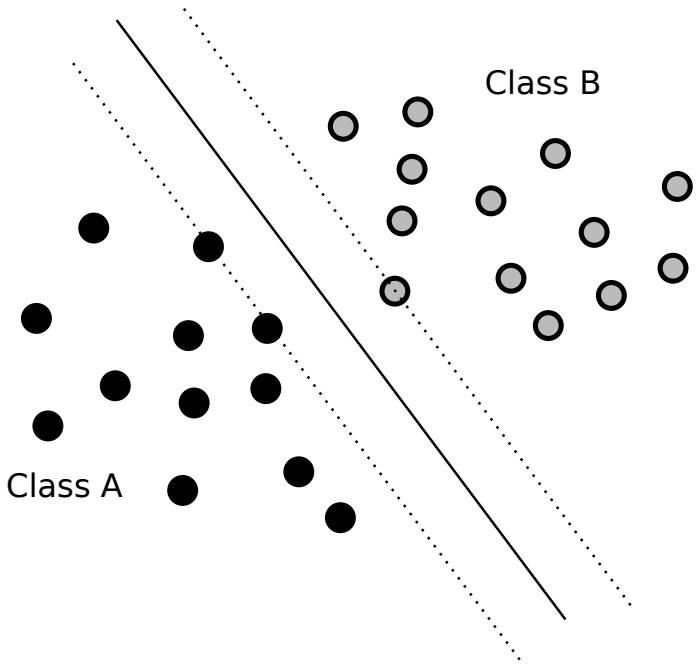


Figure 3.3: A maximum margin hyperplane is the unique separator that maximizes the margin between the two classes.

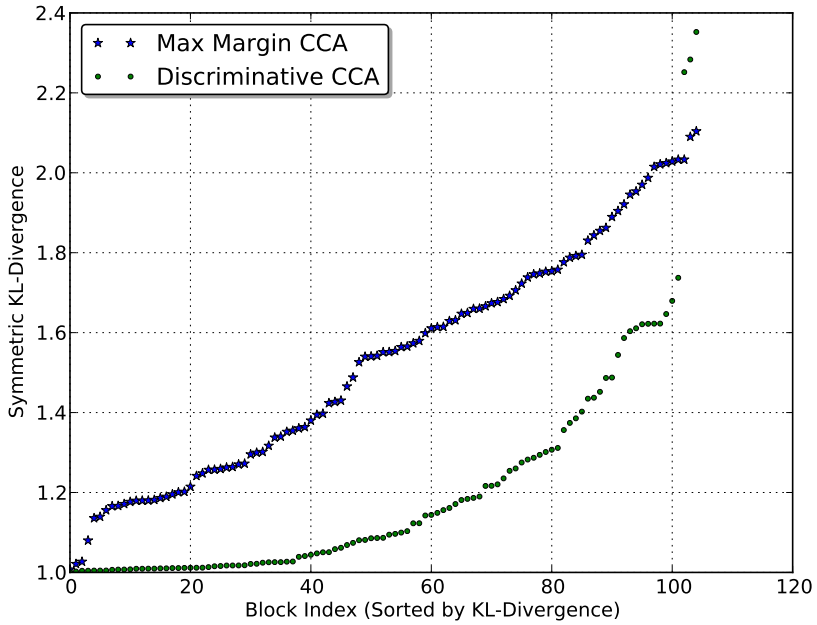


Figure 3.4: KL divergence values between the estimated Gaussian distributions of the CCA features on positive and negative samples in the testing set. The divergence values are sorted for each curve separately. Therefore bin indices do not correspond.

divergence [43] between them as an estimate of the discriminative power of the CCA feature from the particular block. As shown in Figure 3.4, the max-margin method produces more distinctive distributions for both classes, suggesting better discriminative power.

3.2.3 Features

We conduct experiments by fusing two feature types. The first is the Patch Appearance Dictionary introduced in Chapter 2. The second is the Local Binary Patterns (LBP [2]). In the following section we will briefly describe the LBP feature extraction process.

Local Binary Patterns

Local binary patterns (illustrated in Figure 3.5) are powerful descriptors of local texture information. The basic descriptor characterizes the texture

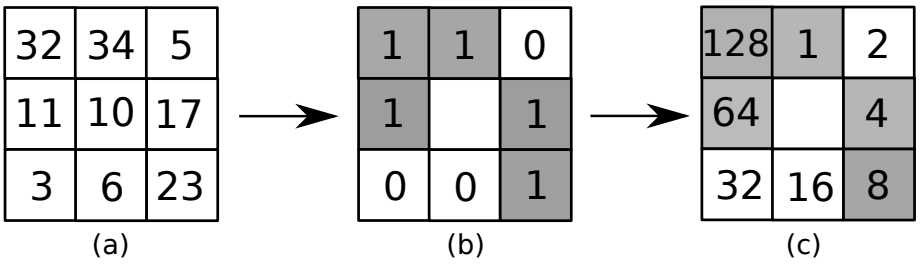


Figure 3.5: Illustration of $\texttt{LBP}_{n8}^{r1}$ feature extraction process for a single pixel. For simplicity of illustration the sampling pattern is not circular, but rather consists of the 8 neighbors of the center pixel. For the given 3×3 image patch (a), the sampling points with higher grayscale value than the center pixel are assigned the bit 1 (b). A binary number is formed by concatenation of the bits.

appearance around a given pixel by first defining a circular sampling pattern. The grayscale image value is measured on equally spaced locations on this circle. A grayscale value is linearly interpolated if the particular sampling location falls on a non-integer pixel coordinate. Subsequently the sampled grayscale values are compared to the grayscale image value at the center pixel. The ones that are greater than or equal to the pixel's grayscale value are assigned the bit 1 and the rest 0. These bits are concatenated as if they were the digits of a binary number, which constitutes the binary pattern of this pixel. The LBP descriptor is defined by several parameters:

$$R = \texttt{sampling radius} \tag{3.8}$$

$$P = \texttt{number of sampling locations} \tag{3.9}$$

$$u = \texttt{maximum number of 0-1 transitions} \tag{3.10}$$

The first two parameters R and P are already defined. The last parameter u determines the maximum number of 0–1 transitions allowed in the binary pattern. All binary patterns with 0–1 transitions exceeding the threshold u are labeled as non–uniform patterns. For example, if $u = 2$ the binary pattern 01001011 would be counted as a non–uniform pattern because it has three 0–1 transitions, whereas 11100101 would be a regular binary pattern.

Typically the last step of LBP feature extraction consists of building local histograms of regular binary patterns.

3.3 Experiments

To quantize the contribution of the LBP features as well as the feature fusion approach through CCA, we test the system on the same INRIA–Pedestrian detection setting from Chapter 2. To obtain all classifiers in the results, we follow a two–stage training strategy. At the first stage we use features from all positive training images as well as five times as many negative features to train a linear SVM classifier. In the second stage we augment the negative set with all false positives, which the classifier trained in the first stage picks up in the entire set of negative training images.

We set the size of the pedestrian images to 64 columns horizontally and 128 rows vertically. We set the cell size N to be (8×8) for both PAD descriptors and LBP descriptors. Thus a detection window of size (128×64) will contain $(16 \times 8 = 128)$ cells in total. The block size M is set as (2×2) and we extract normalized block descriptors from every grid location in the cell grid. In total we obtain $(15 \times 7 = 105)$ block descriptors for both feature modalities. The LBP feature parameters are set to $R = 1$, $P = 8$ and $u = 2$, which were experimentally shown in [32] to be a good parameterization of the LBP feature for the INRIA–Pedestrian dataset. For $u = 2$, the number of regular LBP patterns is 55, which yields a 56–dimensional histogram including the bin for non–uniform binary patterns. Subsequently the block descriptors of LBP are 56×4 dimensional. The PAD descriptors from Chapter 2 are used directly. We use the 100 element PAD, yielding 400–dimensional block descriptors.

The performance of the feature combinations is outlined in the Figure 3.6. The simple combination of LBP and PAD features already yields a significant improvement over the baseline PAD detection performance: the miss rate is reduced to about 11% at 10^{-5} false positives per window. Our proposed CCA augmentation further reduces the miss rate to just below 10% at the same operating point.

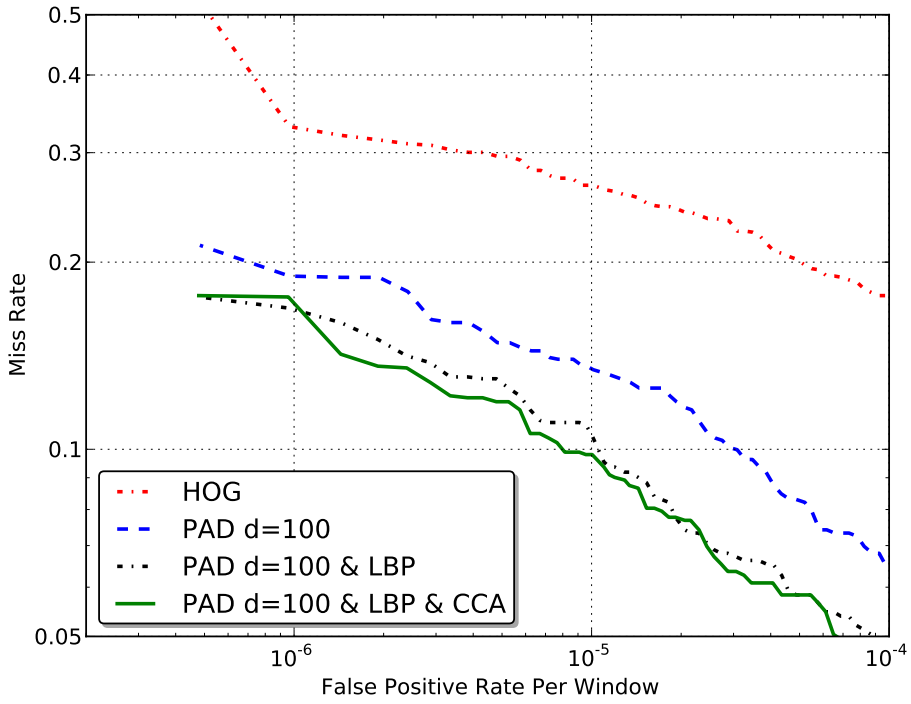


Figure 3.6: Receiver operating characteristic curve for pedestrian detection performance on the test set.

3.4 Conclusions

We have introduced a method for augmenting combined feature representations with values extracting the discriminative correlations between the feature values with close spatial support. Experimental results on the INRIA–Pedestrian dataset suggest that the augmented canonical correlation values can improve the generalization performance of fused representations. Future research directions can investigate whether multiple canonical correlation directions can be extracted from the same location. This is already achievable in the discriminative CCA method, which yields $\min(d_1, d_2)$ canonical correlation projection pairs, when d_1 and d_2 are the dimensions of the fused feature types; however, the generalization performance is limited as remarked. It should also be possible to extract multiple canonical correlation projections through the max–margin method by remarking that the proposed cost function is not convex. Thus, initializing it with different projection directions obtained with discriminative CCA could reveal different local minima.

CHAPTER 4

RECOGNITION THROUGH METRIC LEARNING

Viewpoint invariant recognition of pedestrians is a problem that appears in numerous contexts in computer vision scenarios such as multi-camera tracking, person identification with an exemplar image or re-identification of an individual upon re-entering the scene after some time. This is a key problem and has been drawing attention in recent years with the advance of visual tracking and widespread deployment of surveillance cameras, which have necessitated continuous tracking and recognition across different cameras even with significant time and location differences. Our approach handles the long time delay case: recognition of the same individual without the temporal and spatial information associated with the images of the pedestrians. By learning an appropriate distance metric we achieve high recognition with high accuracy. Although we demonstrate it in the context of this problem, the learned metric is general and can be applied to aid data association in other tracking scenarios.

Several attempts have been made to tackle the recognition problem in the context of matching pedestrians by their appearance only. Park et al. [44] perform recognition by matching color histograms extracted from three horizontal partitions of the person image. Hu et al. [45] have modeled the color appearance over the silhouette's principal axis. However, finding the principal axis requires robust background subtraction and is error prone in crowded situations. Matching spatio-temporal appearance of segments have been considered by Gheissari et al. [46]. Yu et al. [47] introduced a greedy optimization method for learning a distance function. Gray and Tao [48] defined the pedestrian recognition problem separate from multi-camera tracking context and provided a benchmark dataset (VIPeR, see Figure 4.1) for standardized evaluation. Their method transforms the matching problem into a classification problem, in which a pair of images is assigned a positive label if they match (i.e., belong to the same individual) or a negative label otherwise.

Figure 4.1: Representative image pairs from the VIPeR dataset (images on each column are the same person). The dataset contains many of the challenges observed in realistic conditions, such as viewpoint and articulation changes as well as significant lighting variations.

This classifier is learned in a greedy fashion using AdaBoost. The weak classifiers are decision stumps on individual dimensions of histograms of various features within a local rectangle in the person image. The rectangles span the entire horizontal dimension, while they are densely sampled vertically over all positions and sizes. Note that in the context of nearest neighbor classification, the $\{+1, -1\}$ labeling scheme of the matches vs. non-matches creates a naturally unbalanced learning problem with N vs. N^2 samples in two classes respectively (N = number of training points). Also worth noting is that the two methods [47, 48], which learn the pairwise comparison function, achieve this through greedy optimization, which is not globally optimal and furthermore makes indirect use of covariances in the feature space. Our method not only is globally optimal but also has an explicit covariance modeling of features.

The contributions of this work are the following: (1) We apply a large margin nearest neighbor approach to the pedestrian recognition problem to achieve significantly improved results, (2) we define a novel cost function for learning a distance metric specifically for nearest neighbor problems with rejection. In addition we show that despite using only color as the appearance feature, our method is robust under significant illumination changes.

4.1 Metric Learning

In this section, we briefly introduce the metric learning framework of Weinberger and Saul [49] for large margin nearest neighbor (LMNN) classifier. The goal is to learn a Mahalanobis metric where the squared distances are denoted by

$$\mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^{\mathsf{T}}\mathbf{M}(\vec{x}_i - \vec{x}_j), \tag{4.1}$$

$\mathcal{D}_{\mathbf{M}}^{1/2}$ is a valid distance if and only if $\mathbf{M}$ is a symmetric positive-semidefinite matrix. In this case $\mathbf{M}$ can be factored into real-valued matrices as $\mathbf{M} = \mathbf{L}^{\mathsf{T}}\mathbf{L}$. Then, an equivalent form for (4.1) is

$$\mathcal{D}_{\mathbf{L}}(\vec{x}_i, \vec{x}_j) = ||\mathbf{L}(\vec{x}_i - \vec{x}_j)||^2. \tag{4.2}$$

LMNN learns a real-valued matrix $\mathbf{L}$ that minimizes the distance between each training point and its K nearest similarly labeled neighbors (Equation 4.3), while maximizing the distance between all differently labeled points, which are closer than the aforementioned neighbors' distances plus a constant margin (Equation 4.4).

$$\varepsilon_{pull}(\mathbf{M}) = \sum_{i,j \rightsquigarrow i}^{N} \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j), \tag{4.3}$$

$$\varepsilon_{push}(\mathbf{M}) = \sum_{i,j \rightsquigarrow i} \sum_{k=1}^{N} (1 - y_{ik}) \left[1 + \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) - \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_k)\right]_+ \tag{4.4}$$

Here, y_{ik} is an indicator variable which is 1 if and only if $\vec{x}_i$ and $\vec{x}_j$ belong to the same class, and $y_{ik} = 0$ otherwise. The $j \rightsquigarrow i$ notation means that $\vec{x}_j$ is one of the K similarly labeled nearest neighbors of $\vec{x}_i$ (i.e., $\vec{x}_j$ is a target neighbor of $\vec{x}_x$). Note that for ε_{pull} to be a continuous and convex function, it is necessary that the K target neighbors of each training sample be fixed at the initialization. In practice they are determined by choosing the K nearest neighbors by Euclidean distance.

The $\vec{x}_k$ in Equation 4.4 for which $y_{ik} = 0$ are called the impostors for $\vec{x}_i$. The expression $[z]_+ = max(z, 0)$ denotes the standard hinge loss. Although this hinge loss is not differentiable at $z = 0$, we did not observe any convergence issues. Nevertheless it is always possible to replace the standard hinge

loss with a smooth approximation [50].

The affine combination of ε_{pull} and ε_{push} through the tuning parameter μ[1] (Equation 4.5) defines the overall cost, which essentially maximizes the margin for K nearest neighbor classifier by pulling together identically–labeled points and repelling differently labeled ones (impostors).

$$\begin{aligned}\varepsilon_{\mathbf{LMNN}}(\mathbf{M}) =&(1-\mu)\sum_{i,j\rightsquigarrow i}\mathcal{D}_{\mathbf{M}}(\vec{x}_i,\vec{x}_j)\\ &+\mu\sum_{i,j\rightsquigarrow i}\sum_{k=1}^{N}(1-y_{ik})\left[1+\mathcal{D}_{\mathbf{M}}(\vec{x}_i,\vec{x}_j)-\mathcal{D}_{\mathbf{M}}(\vec{x}_i,\vec{x}_k)\right]_+ .\end{aligned} \tag{4.5}$$

4.1.1 Nearest Neighbor with Rejection

In this section we introduce our LMNN-R framework for doing K nearest neighbor classification with the option of rejection. As a practical example for this problem, consider the person re-identification task, where given an image of a pedestrian, one would like to determine whether the same person is in the current scene or not. The target set of the people in the scene may not contain the query person. One way to adapt the nearest neighbor classifier to the problem of re-identification is to adopt a universal threshold (τ) for maximum allowed distance for matching image pairs. If the distance of the nearest neighbor of the query in the target set is greater than τ, one would deem that the query has no match in the target set (rejection). Conversely, if there is a nearest neighbor closer than τ, then it is called a match. What we have just described is the 1 nearest neighbor with rejection problem. This problem can be extended to the K nearest neighbor case, in which a label is assigned through majority voting of P nearest neighbors within τ, where $P \leq K$. If $P = 0$ the classifier will refuse to assign a label.

The introduction of the option to refuse label assignment necessitates a distance metric that allows the use of a global threshold in all localities of the feature space. One method would be to assume unimodal class distributions as proposed by Xing et al. [51]. Their objective function maximizes the distance between all sample pairings with different labels, while a constraint is imposed on the pairs of similarly labeled points to keep them closer than

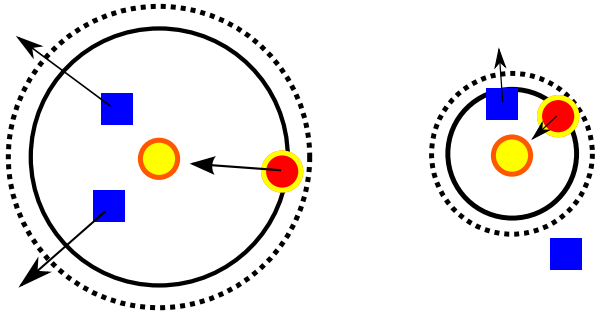
[1] All reported experiments use $\mu = 0.5$ for both LMNN and LMNN-R.

a universal distance. This model was proposed for learning a distance metric for k-means clustering. It does not directly apply to our problem formulation. One drawback is the situation when similarly labeled samples do not adhere to a unimodal distribution (e.g., two islands of samples with same labels). Another problem is the lack of margin in their formulation, which is essential for good generalization performance in classification. A cost function, which emphasizes local structure, is more suitable in our case.

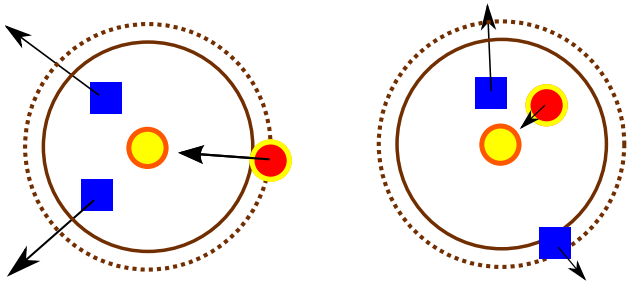
We adopt the LMNN cost function (Equation 4.5), which minimizes the distance between each training point and its K nearest similarly labeled neighbors (Equation 4.3), while maximizing the distance between all differently labeled points, which are closer than the aforementioned neighbors' distances plus a constant margin (Equation 4.4). The margin imposes a buffer zone to ensure good generalization. It is this local property that makes the LMNN metric learning very suitable to nearest neighbor classification. Note that the distance to determine the impostors is varying for each training point $\vec{x}_i$ (Equation 4.4). We replace this with a universal distance: a distance proportional to the average distance of all K nearest neighbor pairs in the training set (Equation 4.6). LMNN-R cost function forces the closest impostors of a training point to be at least a certain distance away, determined by this average which is only weakly affected by where its own K nearest neighbors are (Figure 4.2). The net effect of this modification is that now we can use a universal threshold on pairwise distances for determining rejection, while still approximately preserving the local structure of the large margin metric learning. The only requirement for the loss function to be convex is that the K nearest neighbor structure of the training points needs to be pre–defined. However, extensions such as multi-pass optimization [49] proposed to alleviate this problem for LMNN apply to LMNN-R also.

$$R = \frac{\lambda}{NK} \sum_{m,l \rightsquigarrow m} \mathcal{D}_{\mathbf{M}}(\vec{x}_m, \vec{x}_l) \tag{4.6}$$

$$\varepsilon_{\mathbf{LMNN\text{-}R}}(\mathbf{M}) = (1-\mu)\varepsilon_{pull}(\mathbf{M}) + \mu\varepsilon^*_{push}(\mathbf{M}) \tag{4.7}$$



(a) LMNN



(b) LMNN-R

Figure 4.2: Illustration contrasting our proposed approach with [49]. Note that the point configurations for (a) and (b) are the same. For a given training point (yellow), the target neighbor (red) is pulled closer, while the impostors (blue) are pushed away. (a) To determine the impostors, the LMNN cost function uses a variable distance from the training point depending on the proximity of the target neighbors; (b) LMNN-R, on the other hand, forces the impostors out of a universal distance from the training point, while simultaneously attracting target neighbors.

$$\varepsilon^*_{push}(\mathbf{M}) = \sum_{i=1}^{N}\sum_{k=1}^{N}(1-y_{ik})\left[1+\frac{\lambda}{NK}\left(\sum_{m,l\rightsquigarrow m}\mathcal{D}_{\mathbf{M}}(\vec{x}_m,\vec{x}_l)\right)-\mathcal{D}_{\mathbf{M}}(\vec{x}_i,\vec{x}_k)\right]_+, \tag{4.8}$$

The LMNN-R cost (Equation 4.7) can be minimized as a semidefinite program, which is formulated by writing ε^*_{push} as a constraint through the introduction of slack variables, or it can be minimized by following the gradient directly and projecting $\mathbf{M}$ back to the semidefinite cone at each iteration (iterative sub-gradient projection as in [49]).

4.2 Experiments

We demonstrate the performance of our method on the VIPeR dataset [52], which is a specifically constructed dataset for the viewpoint invariant pedestrian recognition problem. This dataset contains images of 632 unique pedestrians and a total of 1264 images composed of two views per pedestrian seen from different viewpoints. The images are captured outdoors under uncontrolled lighting. Therefore, there is a great deal of illumination variance in the dataset, including between the images belonging to the same pedestrian (e.g., the first and last columns in Figure 4.1). Compared to the previously available datasets (see [48]), the VIPeR dataset has many more unique subjects and contains a higher degree of viewpoint and illumination variation, which makes it realistic and more challenging (Figure 4.1).

4.2.1 Methodology

As done in [48], we randomly split the set of pedestrians into two halves: training and testing. The LMNN and LMNN-R frameworks learn their respective distance metric using the training set. For testing, each image pair of each pedestrian in the test set is randomly split to query and target sets. The results are generated using the pairwise distance matrix between these query and target subsets of the images in the test set. For thoroughness, we report our results as an average over 10 train-test splits. When reporting an average is not appropriate, we report our best result out of the 10 splits.

We follow the same evaluation methodology of [48] in order to compare our results to theirs and other benchmark methods. We report results in the form of cumulative matching characteristics curve (CMC), re-identification rate curve and expected search time by a human operator. In addition, we also provide an average receiver operator characteristic curve to demonstrate the improvement of the LMNN-R method over LMNN for automated recognition.

4.2.2 Image Representation

The images in the dataset are 128 pixels tall and 48 pixels wide. We use color histograms extracted from 8x24 rectangular regions to represent the images. The rectangular regions are densely collected from a regular grid with 4 pixel

spacing in the vertical and 12 pixel spacing in the horizontal direction. This step size is equal to half the width and length of the rectangles, providing an overlapping representation.

For the color histograms, we use RGB and HSV color spaces and extract 8-bin histograms of each channel separately. We tried several combinations for all of the mentioned parameters and found that these numbers worked reasonably well through our preliminary experiments. We concatenate the histograms extracted from an image and obtain a feature vector of size 2232 for RGB and HSV representations each. The combined representation is simply the concatenation of these two. Dimension reduction through PCA is applied to these high-dimensional vectors to obtain subspaces of specific dimensionality. This step is necessary to reduce redundancy in the color based representation and to filter out some of the noise. The reported results are obtained with 20–, 40– and 60–dimensional representations. We have observed that we get diminished returns above 60 dimensions.

To account for the illumination changes we experiment with a simple color correction technique where each RGB channel of the image is histogram-equalized independently to match a uniform distribution as close as possible in ℓ_1 norm. Since in the cropped images, a significant number of the pixels belong to the pedestrian, this is a reasonable way of performing color correction. We also experimented with brightness and contrast correction methods, as well as histogram equalizing the V channel of the HSV images. However, they were not able to perform as well as the described RGB histogram equalization method.

4.2.3 Results

Recognition

We present the recognition performances as CMC curves in Figure 4.3. This curve, at rank score k, gives us the percentage of the test queries whose target (i.e. correct match) is within the top k closest match. As it is not appropriate to take the average of CMC curves over different random splits of the dataset, we report the CMC of a single split where the normalized area under the curve is maximum. This corresponds to using "RGB+HSV"

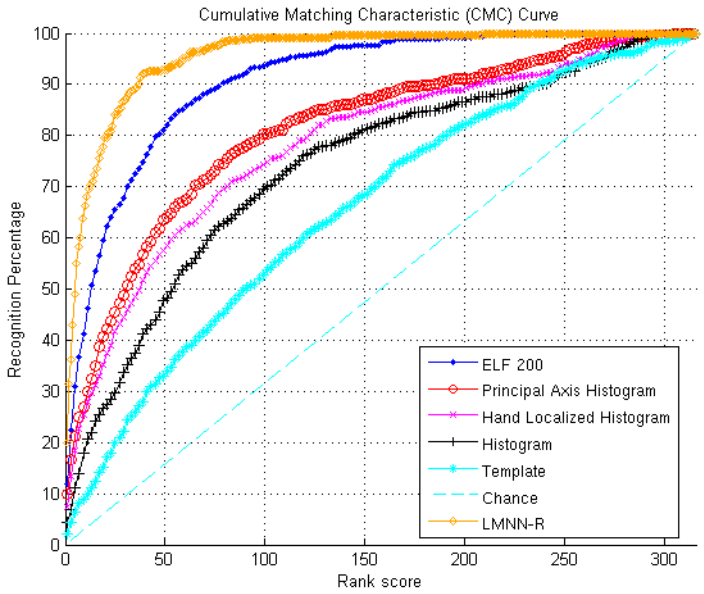


Figure 4.3: Cumulative matching characteristics (CMC) curve for our method and others. This result is obtained using a combined HSV and RGB representation in a 60 dimensional subspace learned with PCA.

features reduced to 60 dimensions via PCA and using our proposed approach LMNN-R. We outperform all previously reported results.[2] An explanation of the methods used to obtain these previous results is as follows. "Chance" refers to random matching, "Template" refers to pixelwise sum-of-squared distances matching. "Histogram" and "Hand Localized Histogram" refer to the method by Park et al. [44], and "Principal Axis Histogram" refers to the method of Hu et al. [45]. "ELF 200" (or just "ELF" in the remaining of the text) refers to the work of Gray et al. [48].

CMC curves can be summarized using the "expected search time" measure defined in [48]. Assuming that a human operator reviews a query image's closest matches sequentially according to their distance from the query and assuming an average review time of 1 s per image, the total expected search time for finding the correct match would be the average rank of the target. Our method's expected target rank is 23.7, which is an improvement of over 15% with respect to the state-of-the-art 28.9 (see Table 4.1).

To evaluate the performance of LMNN and LMNN-R over all different combinations of parameter and feature choices, we use the normalized area under the CMC curves. Table 4.2 shows the mean and standard deviation of

[2]Results of other methods are from [48] courtesy of D. Gray.

Table 4.1: Expected search times for LMNN-R and other methods.

Method	Expected Search Time (in seconds)
Chance	158.0
Template	109.0
Histogram	82.9
Hand Localized Histogram	69.2
Principal Axis Histogram	59.8
ELF	28.9
LMNN-R	**23.7**

these values over 10 random splits of the dataset. Best results are obtained using RGB and HSV together on original (non-color corrected) images. RGB alone performs worse than HSV alone, which is expected because HSV is more robust to variations in intensity of the lighting.

Since the dataset has a significant degree of illumination variation, one expects that color correction should help increase the matching accuracy. While this is true for the plain ℓ_2 norm (i.e. no learning), it is not the case for learned metrics of LMNN and LMNN-R. A possible explanation for this can be made by realizing that the histogram equalization process is a non–linear transformation of the data. While improving the performance of the marginal cases for simple matching by Euclidean distances, this procedure may affect the average transformation that image pairs undergo in realistic scenarios, such that this transformation cannot be reliably modeled by LMNN and LMNN-R methods anymore. Therefore we suggest letting the learning algorithm handle the color correction issues.

For the number of reduced dimensions, 60 is slightly better than 40. And LMMN-R gives slightly better results than LMNN in general.

In the previous re-identification experiments, we assume that the target set will have a match for the query image. This is not the case in many practical scenarios as often it is not known whether the query person is in view. Therefore we also show the receiver operator characteristic curve (ROC) for cases where one would like to detect the query pedestrian in a target set of pedestrians. The detection performance is measured by comparing the true positive rate vs. the false positive rate, which shows, for a given recall rate (true positive), what fraction of non–matching images in the target set will be returned as false positives. Due to the universal threshold, the LMNN-R

Table 4.2: Table of results averaged over 10 random splits of the dataset. 20, 40 and 60 denote the number of dimensions (of the reduced subspace found by PCA) used; L_2 refers to the regular ℓ_2 norm which, in our case, corresponds to "no learning." "Corr'd" means "color corrected" and "orig" indicates that no modification was done to the original image. We obtain our best average results using RGB and HSV together on original images with the proposed learning approach LMNN-R. The overall best result, i.e. the one given in Figure 4.3, has a normalized area of 95.88 under its CMC curve, which is comparable to the average results.

		RGB+HSV		HSV		RGB	
		corr'd	orig	corr'd	orig	corr'd	orig
20	L_2	76.61 ± 0.88	72.54 ± 0.77	80.09 ± 0.59	77.97 ± 0.81	67.85 ± 1.13	60.63 ± 0.79
	LMNN	91.81 ± 0.39	93.46 ± 0.36	92.11 ± 0.47	92.90 ± 0.34	82.06 ± 0.69	86.39 ± 0.72
	LMNN-R	92.14 ± 0.37	93.59 ± 0.37	92.35 ± 0.47	92.87 ± 0.51	82.47 ± 0.83	86.63 ± 0.68
40	L_2	77.48 ± 0.87	73.73 ± 0.81	80.79 ± 0.66	78.89 ± 0.80	68.73 ± 1.11	60.90 ± 0.92
	LMNN	92.68 ± 0.44	94.54 ± 0.42	92.82 ± 0.33	94.40 ± 0.32	83.81 ± 1.27	87.14 ± 0.86
	LMNN-R	93.13 ± 0.48	94.76 ± 0.47	93.04 ± 0.45	94.64 ± 0.43	84.71 ± 1.24	87.49 ± 0.92
60	L_2	77.85 ± 0.86	74.14 ± 0.79	80.97 ± 0.67	79.17 ± 0.80	68.83 ± 0.91	61.20 ± 0.91
	LMNN	92.27 ± 0.50	94.67 ± 0.55	92.52 ± 0.28	94.54 ± 0.29	84.23 ± 0.63	87.56 ± 1.01
	LMNN-R	92.56 ± 0.53	$\mathbf{94.95 \pm 0.46}$	92.62 ± 0.43	94.69 ± 0.37	84.94 ± 0.57	87.79 ± 1.04

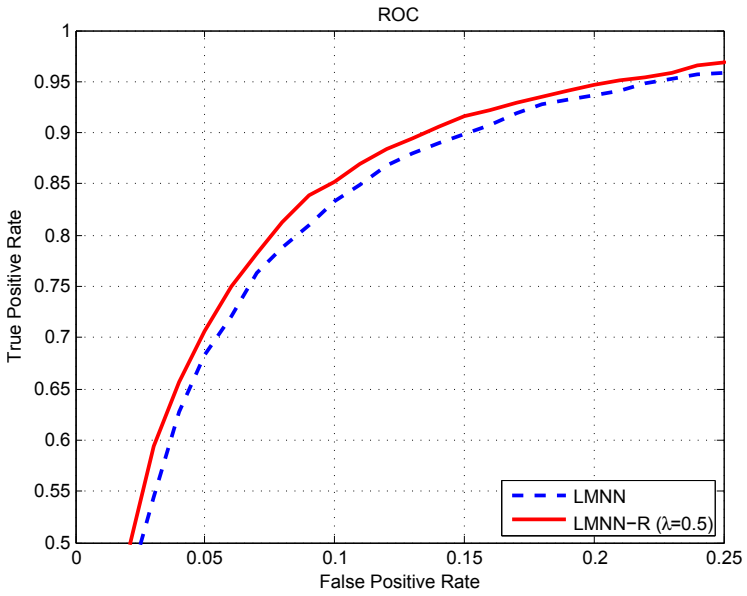


Figure 4.4: The receiver operator characteristic curve showing the true positive vs. the false positive rate of our system.

method with properly selected λ parameter was able to outperform LMNN by about 4% at a false positive rate of 10% (Figure 4.4).

Re-identification

This is another measure for evaluating the performance of pedestrian matching methods. It is the probability of finding a correct match as a function of the number of possible targets. A formal definition can be found in [52]. Figure 4.5 shows the re-identification rates of our method and the previous methods.

Execution times

We implemented LMNN and LMNN-R in MATLAB[3] and although we have not employed the active set method which was designed to make LMNN more efficient (described in [49]), our code runs reasonably fast in practice. For the VIPeR dataset, a typical training session takes 160 s and finding the target of a query pedestrian takes only 1.2 ms on a 2GHz Intel Core2–Duo.

[3]The MATLAB code for LMNN and LMMN-R optimization as well as replicating the experiments is available by request.

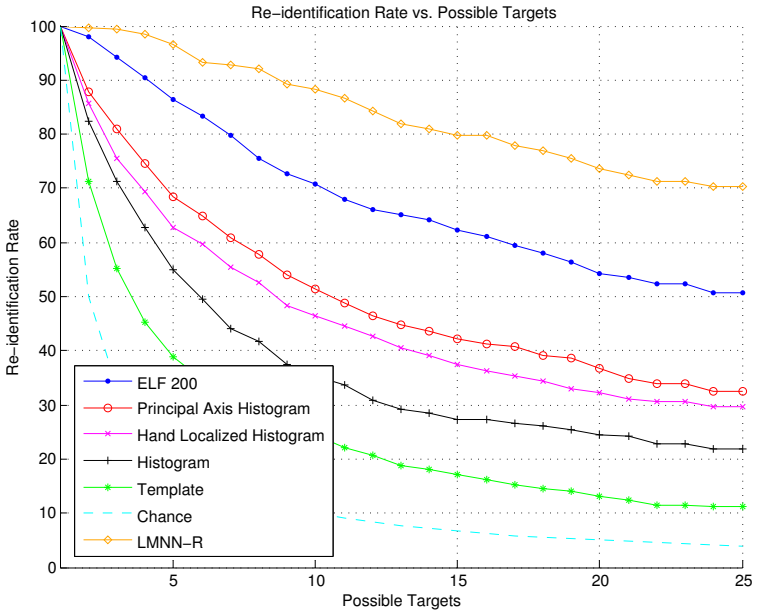


Figure 4.5: Re-identification rate vs. the number of targets for our method and others.

CHAPTER 5

COMPUTATION AND PARALLEL PROCESSING

5.1 Introduction

Graphics processing units (GPU) were first developed as specialty hardware for computationally intensive 3D rendering tasks. Recently, owing to the introduction of device programming frameworks such as Compute Unified Device Architecture (CUDA [53]) and Open Computing Language (OpenCL [54]), programmers have been granted highly granular control over a majority of the compute capabilities of the GPUs, paving the way for much general purpose computing to be executed on GPUs with potentially big performance gains. GPUs by design typically consist of many computation elements that operate in parallel, allowing many tasks to be executed simultaneously. In comparison, single–core central processing units (CPUs) do most processing serially, with a few exceptions such as branch prediction [55] or memory and instruction prefetching [56], which aim to decrease the CPU wait states through speculating about the states of execution in the immediate future. The tradeoffs for the GPU many–core execution are several. First, the memory and cache available to each execution elements are very limited in comparison to CPUs; furthermore, the current architectures operate with on–board memory (hereafter called "device memory"), which is separate from the main memory of the computer (hereafter called "host memory"). This duality of memory structures necessitates the transfer of the relevant data between the host and the device, which can potentially negate the performance gains obtained by parallelism. Lastly, although the processing cores on the GPUs can operate in parallel, this does not necessarily mean they can operate fully independently. Grouping of instructions and carefully designed memory access to maximize the throughput of the memory bandwidth are required in order to make optimum use of the computational

capabilities of the GPUs.

In summary, we can identify several properties of algorithms, to assess their suitability for execution on massively parallel architectures: (1) the computation must consist of bits of execution which can be mapped to independent regions of the data, (2) the cumulative time spent on computation must exceed the time spent on data transfer to make it worthwhile, (3) the GPU execution kernel must be small and should follow a mostly deterministic code path with little to no branch divergence.

Fortunately, many examples in computer vision fit the described conditions well. Within the spectrum of all computer algorithms, most computer vision algorithms tend to reside on the computationally heavy side, generally expanding the given image data into a larger, more descriptive structure (e.g., feature description, scale space pyramid, Markov random field) before reducing it to the desired output. Typically, the expansion stage involves a lot of image processing and the reduction stage involves pattern recognition tasks, both which are data parallel on several levels. On a deeper level (e.g., a single image), one can frequently identify parts of algorithms, in which different chunks of the data can be distributed onto the parallel computational resources and be processed either completely or almost independently. For example, in the basic convolution operation, the filter response at a particular image location only depends on the contents of the image block that is as large as the filter's support at that location. Thus, assuming the filter kernel is already cached, the data reads of two convolution computations at locations more than the filter size apart in either direction are completely independent.

In this section we concentrate on processor–level parallelism. However, it should be noted that an orthogonal direction of parallelism is also possible in computer vision. In most applications such as indexing, retrieval, detection and image segmentation, there is no dependency between different images in a given dataset. Subsequently there is no need for communication between parallel tasks when they are distributed on individual images. Under these "embarrassingly parallel" [57] conditions it is not difficult to realize gains proportional to the number of computation units operating in parallel. Software frameworks such as MapReduce [58] make this kind of distributed computing paradigm possible.

5.2 Introduction to ViVid

The Video Processing Library (ViVid) has been conceptualized as a framework to utilize massively parallel processing capabilities of GPUs to aid the Image Formation and Processing Lab's participation in TRECVid Surveillance Event Detection Evaluations in 2008 and 2009 [59, 60]. Since then, we have successfully used ViVid on the analysis of image sets as well. ViVid has been organized with the intention that vision researchers will be the primary end users, specifically, researchers who would benefit from using performance optimized computational primitives in their algorithm design and testing process.

ViVid is organized in two levels (Figure 5.1): the high–level Python interface, which is used for scripting and prototyping, and the low–level C++/CUDA interface that is used to facilitate high–performance computing.

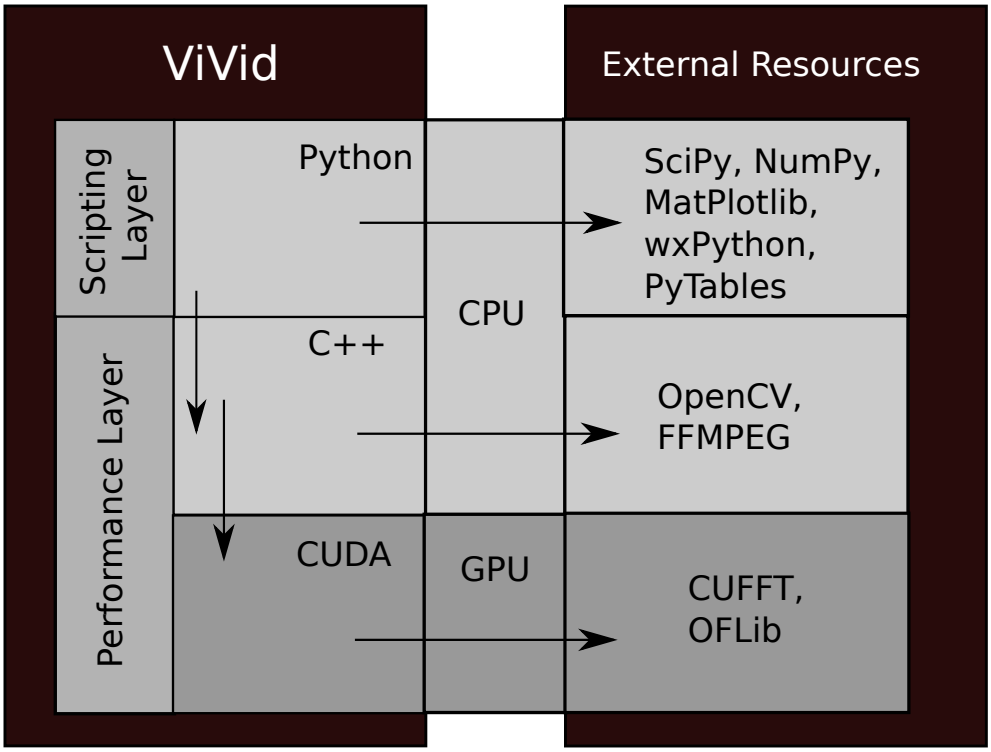


Figure 5.1: ViVid organization. External libraries are utilized at all levels for code reuse and to extend the capabilities.

The Python scripting layer enables access to an extensive range of tools in Python's extensive scientific ecosystem. SciPy with NumPy can be used for fast manipulative and arithmetic operations on array type data, and there is

an ever growing list of compatible libraries which can be useful for pattern recognition under the Scipy project. Matplotlib offers a strong alternative to Matlab's plotting capabilities. PyTables is an open source project for storing data in hierarchical data format (HDF5 [61]), which is specifically designed for efficient access to large datasets, which are frequently encountered in computer vision. Lastly wxPython is a tool for building cross-platform graphical user interfaces. In an ideal setting, the end users are only exposed to ViVid's Python scripting layer, where they can rapidly define the overall program flow and incorporate modules and functions from C++ and CUDA performance layers at will.

In the following sections we outline ViVid's capabilities relevant to local feature computation and processing. We concentrate on performance optimizations on NVIDIA GPUs, which can be programmed through CUDA. In the next section we will give a brief introduction to the CUDA computation model, and the individual discussions of algorithms to follow will make frequent references to terminology specific to the CUDA paradigm.

5.3 GPU Execution Model

A CUDA capable NVIDIA GPU has a certain number of multiprocessors each of which has a given number of stream processors. For instance, the test bed of the experiments in this thesis is a GeForce GTX560, which has 336 stream processors uniformly distributed over 7 multiprocessors.

Each multiprocessor has one instruction decoder, which means we cannot treat the stream processors of a multiprocessor as independent computation cores. Rather, the same set of instructions, called the kernel, needs to be run simultaneously on a group of stream processors in a multi-threaded fashion. This group of threads constitutes a thread block. Each thread block executes the same kernel, preferably on a different section of the data. The number of threads per block is given by the number `BLOCK_SIZE`, whereas the number of thread blocks is given by the number `GRID_SIZE`. The threads in a block are executed *logically* in parallel but not necessarily *physically* in parallel. Only threads within a warp are guaranteed to execute physically in parallel. The warp size is currently 32 for all CUDA capable devices. The grouping of threads in blocks is the main idea behind the scalability of CUDA because

these blocks can be distributed for simultaneous execution depending on the number of available multiprocessors in a given system.

On a CUDA capable GPU, there are several levels of memory, each with a range of availability with respect to individual threads. This availability is generally inversely correlated with the latency of the specific memory. For instance the global device memory is available for read/write access to all threads, but the access latency can consume up to 400-600 clock cycles. The on–chip shared memory is only available to threads within a thread block, but in general has very low latency. The registers are exclusive to individual threads, but the access is virtually free. There are two additional globally available memories with their own caches, which can enable fast access. These are the constant read-only memory (currently limited to 64 KB) and the texture memory.

5.4 Operations

In this chapter we will describe the parts of ViVid that are relevant to the proposed feature extraction mechanism (Figure 5.2). Specifically, we address three modules:

1. Filterbank: Convolving a bank of filter kernels against the image and measuring the filter similarity or distance at every image location and finding the `argmax` (or `argmin` for distance).

2. Block Histogramming: Building histograms from local data.

3. Pairwise Distance: Measuring the pairwise distances or similarities between two sets of vectors.

All of the timings provided are computed on the same computer with a quad–core 2.67 GHz Intel i5-750 Lynnfield CPU. The system memory is 16 GB. The operating system is GNU-Linux x86_64 with kernel version 3.1.4-1-ARCH. An NVIDIA GeForce GTX 560 with 1GB DDR3 device memory is connected through the PCI-e×16 port. A propitiatory Linux display driver, version number 290.10 from NVIDIA Corporation, is used.

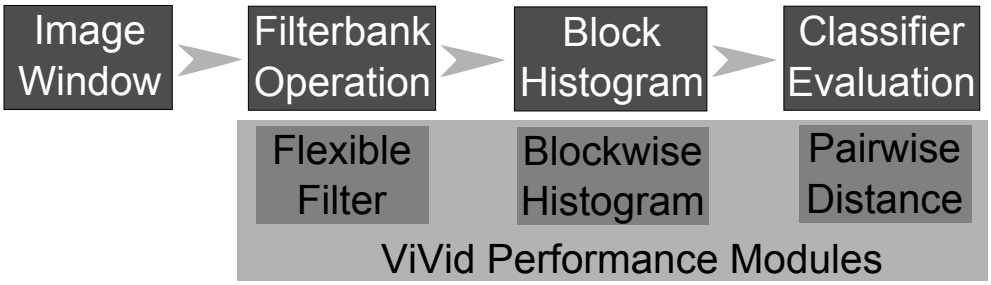


Figure 5.2: Overview of a typical feature extraction process and optimized parts contributed by ViVid. Each operation block provided by ViVid is associated with the corresponding processing step in the block above.

5.4.1 Filtering with a Filterbank

In this problem, we are concerned with finding the most similar filter in a d–long bank of $c \times c$ filters at every square patch of equal size in the image. This is similar to the convolution algorithm, whose GPU implementation has been addressed in several studies [62, 28]. The convolution operation is bandwidth bound for small filters, while it is compute bound for large filters [63]. Typically for the feature transform proposed in Chapter 2, the filter sizes are small, which suggests that our problem could be bandwidth bound as well. Yet, our problem differs from convolution because it consists of d convolution patterns followed by a reduction for selecting the best matching filter index and the corresponding similarity value at every pixel. Since d is generally large, this pushes us well into the compute bound realm.

The Algorithm

In our GPU implementation, we store the filterbank in the constant device memory. As stated in Section 5.3, this is currently limited to 64 KB for CUDA capable NVIDIA GPUs. Maximum size of the filterbank in terms of the number of filters that can fit into the constant memory for a given filter kernel size is outlined in Table 5.1. The constant device memory has a dedicated cache which facilitates fast access to filter coefficients by the threads.

The `BLOCK_SIZE` is set to 16×16, and for each thread block a $(N + \text{filter\_size} - 1) \times (N + \text{filter\_size} - 1)$ shared memory block is allocated for loading an image patch of corresponding size. N is a multiple of `BLOCK_SIZE`.

Table 5.1: Maximum number of filters that can be stored in the constant device memory, assuming single precision floating point coefficients.

Filter Size	3×3	5×5	7×7
Maximum Filterbank Size	1,820	655	334

Table 5.2: Raw timings (μs) for filtering operations with a given filter support size.

	CPU Intel i5-750		GPU GeForce GTX560
Filterbank Size	Single-threaded	Multi-threaded	
$100 \times 3 \times 3$	287,295.1	67,802.7	1,602.48
$100 \times 5 \times 5$	596,630.4	151,968.0	10,530.6
$100 \times 7 \times 7$	1,150,230.6	290,569.5	20,695.8

The extra padding is necessary, such that we can compute the filter responses for an $N \times N$ region. Since we are compute-bound, it is important to keep the maximum number of threads busy. In each block, the shared memory is first populated by the threads simultaneously reading from the device memory. Next, each thread computes the filter responses to $N \times N$ pixels and determines the index of the most similar filter along with the corresponding index. These two values are recorded back to the device memory.

Timings

We tested filterbank filtering against a straightforward implementation on the CPU. Multithreading on the CPU is achieved through OpenMP by parallelizing the outermost loop. In both GPU and CPU implementations, inner loops are unrolled whenever possible. The benchmarks shown in Table 5.2 demonstrate that the pattern of heavy local computation with very little memory communication is rather favorable for the GPU and a clear advantage is reflected in the timings.

5.4.2 Block Histogram

Block histogramming refers to building histograms from local image data with small support. The support regions are usually placed on a regular grid on an image. Staying faithful to the parameterization of the feature

transform introduced in Chapter 2, we set the local support size to be 8×8 image patches, which are densely placed over the image.

The histogramming process can be broken down into two steps. In the first step, the corresponding histogram bin is identified for each data item. In the second step, this bin value is incremented according to the weighting scheme. The first step is basically a nearest neighbor search. This is a well studied problem and efficient structures have been designed to speed up the search for various scenarios depending on the nature of the data [64]. In this section, we address the second step of histogramming and assume the histogram bin associations for all data items have already been resolved.

Histogramming on the GPU

Parallelization of histogramming is an interesting problem because of irregular access patterns for accumulation. Several works have investigated viable GPU algorithms for building histograms from full images [65, 66]. On a GPU, multiple blocks of threads can be launched simultaneously. This behavior naturally suggests a program flow, where several blocks of threads are launched and each block, with its multiple threads, builds one histogram for one image block. There are two obvious algorithmic patterns for this scenario: scatter and gather.

In the gather pattern, all threads read all data items but only increment the histogram bins when the data items read fall into the disjoint region of the histogram for which they are responsible (Figure 5.3). The gather pattern avoids all possible write collisions, but the tradeoff is that all threads will read all data items, which is not a real improvement over straightforward serial implementation.

In scatter pattern the input data will be distributed evenly to worker threads, who then increment the corresponding bin out of a total of G histogram bins, as dictated by the data items (Figure 5.4). This in theory could give a speed–up linear in the number of available worker threads (P); however, this is hindered by collisions that occur when two of the threads want to increment the same histogram bin at a given iteration. In such cases the accumulation must be serialized in order to avoid race behavior. The collision problem at a high level is similar to the birthday paradigm, where the question is, out of P people in a room, are there two or more people

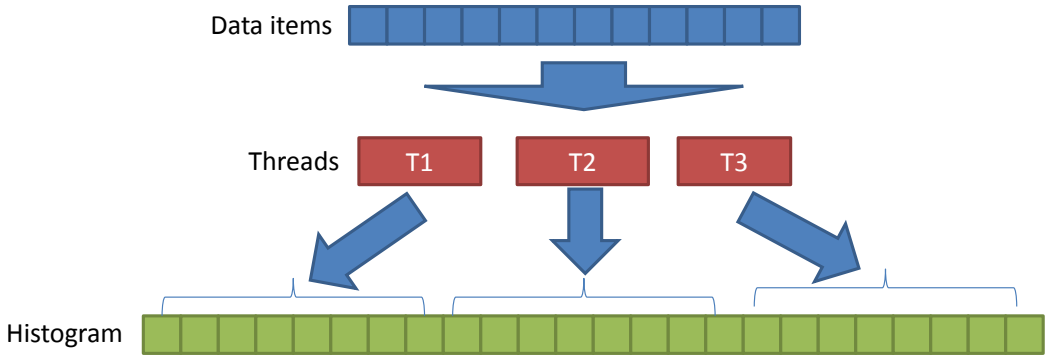


Figure 5.3: Gather strategy for multi-threaded histogramming. Individual processing threads read all data items but only accumulate to a sub-section of the histogram.

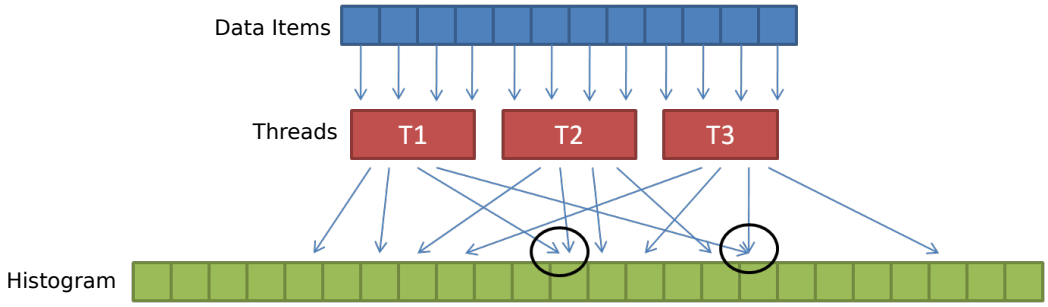


Figure 5.4: Scatter strategy for multi-threaded histogramming. The data is divided among individual processing threads. Threads accumulate values to all histogram bins. Collisions (marked by black circles) can create race conditions and therefore must be accounted for.

who have the same birthday. The expected number of birthday (or thread) collisions (C) in this case is [67]:

$$C = P(P-1)/(2G) \tag{5.1}$$

However, image structures such as gradients or visual words are non–uniformly distributed and frequently follow a power law distribution. Figure 5.5 illustrates averaged block histograms of 8×8 image patches for three data sources. The first source is the image grayscale values. The second source is the histogram of patch appearances with a Patch Appearance Dictionary size of 100. Scattering data items with such distributions will often cause collisions and therefore performance penalties which in some cases may end up in large serialization penalties. However, relative to grayscale intensity histogramming, there are fewer collisions when using Patch Appearance Dic-

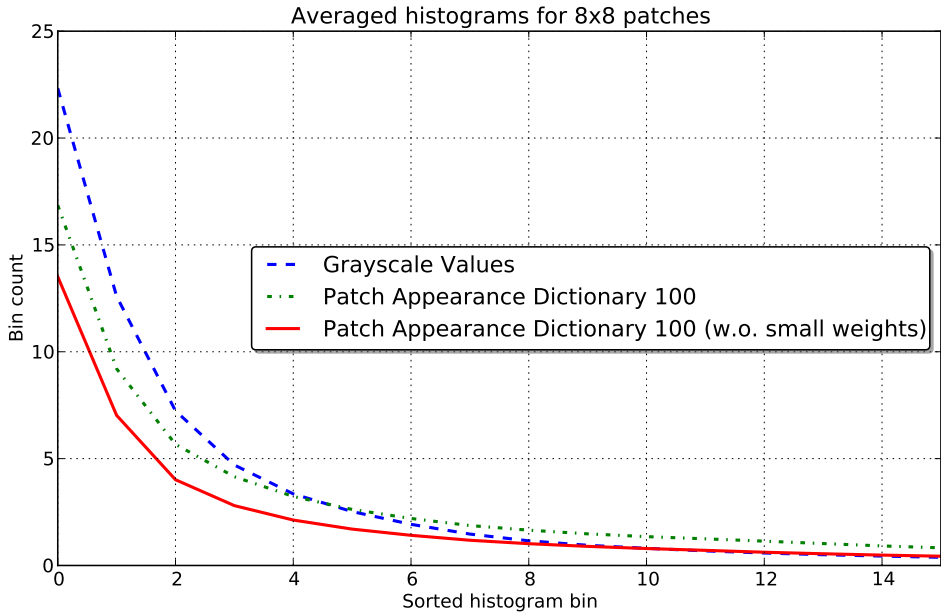


Figure 5.5: Averaged block histograms from 8×8 blocks of a typical image. All histograms are 100 bins long. Before averaging the histograms, values have been sorted from high to low.

tionaries. Furthermore, the number of collisions can be greatly reduced with a slight loss of fidelity in local histograms. Note that in PAD representation the local histograms are weighted, with the weights equal to the absolute cosine similarity to the most similar dictionary item. Because the filters are 0 mean, in smooth image patches, which constitute most of the patches in natural images, this similarity is going to be very close to 0. Thus we can take advantage of this fact by not performing an add operation when the similarity value is less than a predetermined threshold (Figure 5.6).

The Algorithm

Encouraged by the relatively low number of expected collisions within blocks, the proposed block histogramming algorithm follows the scatter pattern, taking advantage of the hardware atomics for handling bin collisions. It is natural to think that the `BLOCK_SIZE` should be equal to the size of the image patch. That way each block could build precisely one histogram. The number of active blocks per multiprocessor is limited to 8, which yields $8 \times 64 = 512$ threads per multiprocessor. The maximum number of active threads per multiprocessor is 1536 however, resulting in a block occupancy

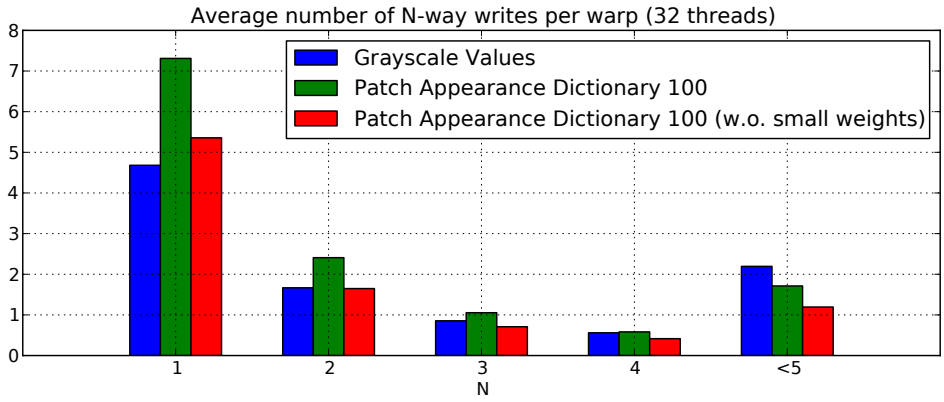


Figure 5.6: Expected number of N–way writes in a group of 32 threads, which execute in parallel. $N = 1$ means there are no collisions. On average, histogramming PAD features cause fewer collisions than grayscale histograms. The number of collisions can be further reduced by not accumulating values from pixels with low contribution.

of $\frac{512}{1536} = 33\%$. According to CUDA C Best Practices Guide [68] block occupancy figures less than 50% may lead to underutilization of resources. To increase the block occupancy we set the `BLOCK_SIZE` to 16×16. This means each thread block will be building 4 histograms. A shared memory block of $4 \times$ `MAX_HISTOGRAM_SIZE` is allocated for fast local access. We set `MAX_HISTOGRAM_SIZE` to 500, which almost completely fills the shared memory limit per block. The `MAX_HISTOGRAM_SIZE` does not impose a limit on the size of the histograms we can handle, however, because histograms larger than 500 bins can still be processed through multiple invocations of the kernel, each time building a different 500 bin subset of the histogram.

Each thread reads the assignment value (i.e., the target bin index) of a single pixel in the image block and the corresponding weight. If the weight value is larger than a threshold τ, the corresponding bin in the block histogram is incremented through the invocation of `atomicAdd()`, which guarantees accurate summation in the case of collisions. Finally, the threads transfer the block histograms from the shared memory to the device memory (Algorithm 2).

Timings

We provide the timing comparison against a straightforward multi-threaded CPU implementation using OpenMP. The CPU implementation consists of

Algorithm 2 Block histogramming kernel

Allocate 4 temporary histograms ($H_{\texttt{temp}}$) of size `MAX_HISTOGRAM_SIZE` and initialize the values to 0.
Read histogram assignments $b_{i,j}$ and weights $w_{i,j}$ from the corresponding 16×16 image block.
if $w_{i,j} > \tau$ **then**
  `atomicAdd`$(H_{\texttt{temp}}[.][b_{i,j}], w_{i,j})$
end if
Transfer $H_{\texttt{temp}}$ to device memory.

Table 5.3: Raw timings (μs) for building histograms.

	CPU Intel i5-750		GPU GeForce GTX560	
Histogram Size	Single-threaded	Multi-threaded	Exact	Approx
100	1,746.7	637.8	150.0	126.5
200	1,771.0	668.9	162.4	138.4
300	1,918.6	713.2	172.2	148.6

each thread building a separate histogram from a different block on the image data. Table 5.3 shows the run-times for GPU and CPU implementations. This is the raw time it takes to build the histograms (e.g., total kernel time for the GPU). Any other overhead such as the time it takes to transfer the image data from the CPU to the GPU device is not accounted for. The multi-threaded implementation on the CPU yields a performance improvement almost proportional to the number of CPU cores. The GPU is the best performer with about 150 μs total kernel time for 100 bins. The approximate method yields an additional speed–up of 25 μs.

5.4.3 Pairwise Distance

Pairwise operations on elements of two sets are extremely common patterns used in many algorithms. One of the prime examples is matrix multiplication, where one computes the dot product of all the rows of one matrix against the columns of another. The pairwise dot product can be replaced by any pairwise vector operator τ ($\tau := \Re^m \times \Re^m \rightarrow \Re$) to give rise to arbitrary distance measures, similarity transforms. For example, the radial basis

function, a popular support vector kernel, can be expressed as

$$K_{\text{rbf}}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma\tau(\mathbf{x}, \mathbf{y})) \tag{5.2}$$

$$\tau(\mathbf{x}, \mathbf{y}) = \| \mathbf{x} - \mathbf{y} \|_2 \tag{5.3}$$

The typical work flow of the general pairwise vector operation is outlined in Algorithm 3.

Algorithm 3 Straightforward pairwise operation

```
Input: sets of vectors A (m1 × n) and B (m2 × n)
Output: C (m1 × m2)
for i = 1 to m1 do
  for j = 1 to m2 do
    c_{i,j} = τ(a_{i,:}, b_{i,:})
  end for
end for
```

For large collections of large vector sets, pairwise operations start quickly and become prohibitively expensive because of the sheer number of atomic operations involved in the computation. Loop tiling (Figure 5.7) is a parallel algorithm, very well suited to address this problem on massively parallel processing units. The basic idea is to break up the data into smaller subsets, which can be processed independently and accumulate the results in the corresponding subset of the output. This results in a net gain because threads are utilized to move the data into fast shared memory, thereby reducing the global memory access by a factor of `BLOCK_SIZE`. Furthermore, threads process the subsets by computing the pairwise distances of subvector pairs concurrently. The loop tiling algorithm is outlined in Algorithm 4.

The loop tiling operation is simple, yet one of the best manifestations of leveraging true computing power of GPUs due to lack of branches in the atomic computation kernels and very low number of idle threads at any given instance. The performance gain is reflected in our benchmark in Table 5.4. For this benchmark, we measure the Euclidean distance between two sets of 1000-dimensional vectors. The number of vectors in each set is varied, which should result in a quadratic increase in processing time in a serial implementation. The multithreaded CPU implementation is realized through OpenMP. The multithreaded CPU version results in a speed–up almost equal to the number of CPU cores utilized. However, the GPU implementation

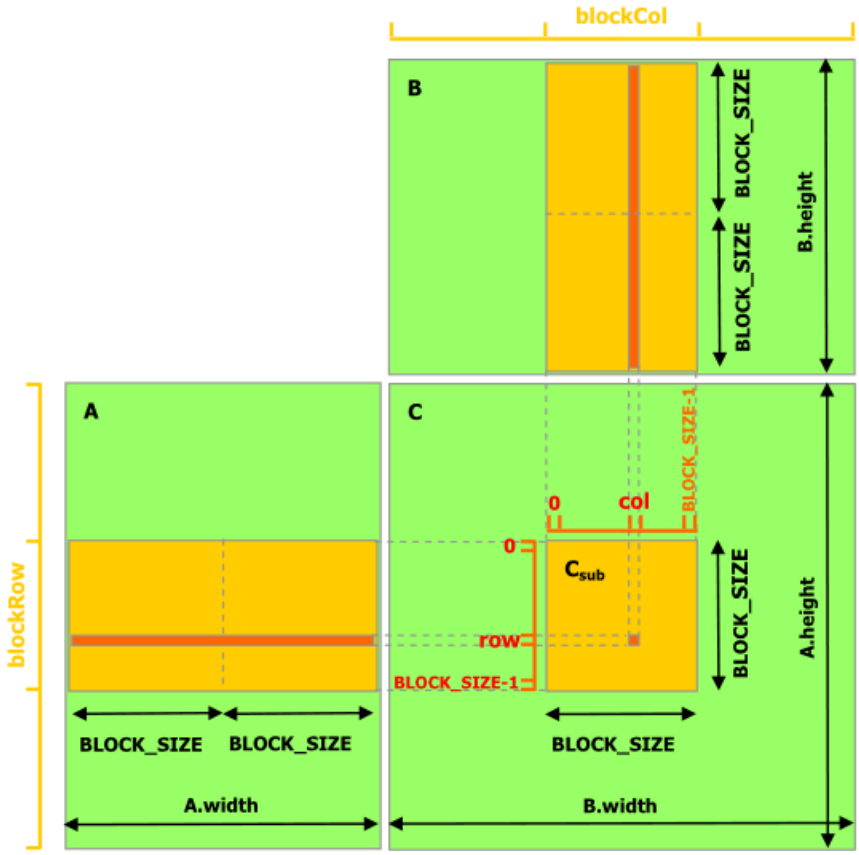


Figure 5.7: Visualization of loop tiling operation. The worker threads concurrently read square–shaped (of dimension `BLOCK_SIZE`) sub-matrices of **A** and **B**. After this, each thread is responsible for applying the specified pairwise operation to a row of the sub-matrix from **A** and a column of the sub-matrix from **B**. Figure courtesy of NVIDIA CUDA Programming Guide [**?**].

Algorithm 4 Pairwise vector operations on two sets of vectors via loop tiling

Input: sets of vectors $\mathbf{A}$ ($m_1 \times n$) and $\mathbf{B}$ ($m_2 \times n$)
Output: $\mathbf{C}$ ($m_1 \times m_2$)
Initialize: `BLOCK_SIZE` $\times$ `BLOCK_SIZE` threads indexed by $\{t_i, t_j\}$
Partition $\mathbf{A}$ and $\mathbf{B}$ into disjoint `BLOCK_SIZE` $\times$ `BLOCK_SIZE` submatrices indexed by $\{i_{\{a/b\}}, j\}$
for $j = 1$ to $j^{(\max)}$ **do**
  for all submatrices $\mathbf{A}_{i_a,j}$ and $\mathbf{B}_{i_b,j}$ **do**
    for all threads (t_i, t_j) **do**
      Read one element of $\mathbf{A}_{i_a,j}$ and $\mathbf{B}_{i_b,j}$ into the device shared memory
      $C_{i_a,i_b}[t_i, t_j] \leftarrow C_{i_a,i_b}[t_i, t_j] + \tau(\mathbf{A}_{i_a,j}[:, t_i], \mathbf{B}_{i_b,j}[:, t_j])$
    end for
  end for
end for

Table 5.4: Raw timings (μs) for computing the pairwise distances of two $N \times N$ matrices.

	CPU Intel i5-750		GPU GeForce GTX560
Histogram Size	Single-threaded	Multi-threaded	CUDA
100×100	2,414.0	772.7	101.4
500×500	300,257.4	72,191.5	8,682.7
1000×1000	2,410,599.8	599,843.1	66,637.6

through loop tiling is the best performer in all cases by a large margin.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis we have investigated object detection and recognition problems through the context of local image features. This bottom–up approach lets us formulate a robust and descriptive feature transformation that preserves the positive qualities of popular and successful gradient operators. Our descriptors show promising performance in object detection tasks. This favorable performance prompts the question whether the descriptors can be useful in other areas such as registration, wide baseline matching and texture recognition. One potential drawback is the computational burden that we alleviated through utilizing GPUs. For processing on the CPUs, an option is to utilize binary search trees to speed up processing at the pixel level.

We extended our study of local descriptors in two directions. In Chapter 2 we proposed a method for extracting discriminative correlations of local features of different types. The correlation values obtained from this method can be augmented into the existing features to boost detection performance. In Chapter 3, we demonstrated a metric learning framework for solving recognition problems using local feature representations.

To facilitate good computational performance, we have produced GPU algorithms that show an order of magnitude improvement on the average over the CPU equivalents. One drawback of the current code base is that our algorithms are written in CUDA language, which is more or less exclusive to NVIDIA devices. Availability of GPUs from other vendors and the emergence of multi-core CPUs necessitate the porting of our code base to these platforms. A reasonable direction of the port is OpenCL, which is a unifying framework for coding parallel devices and is supported as a standard by all major high–performance processor vendors.

REFERENCES

[1] B. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.

[2] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[3] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2005, pp. 886–893.

[5] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[6] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 506–513.

[7] F. Campbell and J. Kulikowski, "Orientational selectivity of the human visual system," *Journal of Physiology*, vol. 187, no. 2, pp. 437–445, 1966.

[8] T. Poggio and S. Edelman, "A network that learns to recognize 3d objects," *Nature*, vol. 343, no. 6255, pp. 263–266, 1990.

[9] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in *IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large–scale hierarchical image database," in *2009 IEEE Computer*

Society Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2005, pp. 886–993.

[13] P. Felzenszwalb, R. Girshick, and D. McAllester, "Discriminatively trained deformable part models, release 4." 2010. [Online]. Available: http://www.cs.brown.edu/~pff/latent-release4/

[14] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, September 2010.

[15] A. Agarwal and B. Triggs, "Hyperfeatures - multilevel local coding for visual recognition," in *2006 European Conference on Computer Vision*, 2006, pp. 30–43.

[16] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2169–2178.

[17] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. 511–518.

[18] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates," in *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 193–199.

[19] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[20] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *2006 European Conference on Computer Vision*, 2006, pp. 404–417.

[21] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 815–830, 2010.

[22] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1491–1498.

[23] Y. LeCun, F. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 97–104.

[24] X. Lian, Z. Li, B. Lu, and L. Zhang, "Max–margin dictionary learning for multiclass image categorization," in *2010 European Conference on Computer Vision*, 2010, pp. 157–170.

[25] J. Yang, K. Yu, and T. Huang, "Supervised translation-invariant sparse coding," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3517–3524.

[26] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, pp. 143–175, 2001.

[27] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027–1035.

[28] D. Lin, X. Huang, Q. Nguyen, J. Blackburn, C. Rodrigues, T. Huang, M. Do, S. Patel, and W. Hwu, "The parallelization of video processing," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 103–112, 2009.

[29] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE 2005 Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[31] J. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[32] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *2009 IEEE International Conference on Computer Vision*, June 2009, pp. 32–39.

[33] F. Aherne, N. Thacker, and P. Rockett, "The bhattacharyya metric as an absolute similarity measure for frequency coded data," *Kybernetika*, vol. 32, no. 4, pp. 1–7, 1997.

[34] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[35] S. Maji, A. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[36] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *2005 IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1458–1465.

[37] A. Kumar and C. Sminchisescu, "Support kernel machines for object recognition," in *2007 IEEE International Conference on Computer Vision*, Oct. 2007, pp. 1–8.

[38] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3539–3546.

[39] Y. Fu, L. Cao, G. Guo, and T. S. Huang, "Multiple feature fusion by subspace learning," in *CIVR '08: Proceedings of the 2008 International Conference on Content–Based Image and Video Retrieval*, 2008, pp. 127–134.

[40] D. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural Computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[41] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag New York, Inc., 1995.

[42] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[43] S. Kullback, *Information Theory and Statistics*. Mineola, NY: Dover Publications, 1997.

[44] U. Park, A. Jain, I. Kitahara, K. Kogure, and N. Hagita, "Vise: Visual search engine using multiple networked cameras," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, 2006, pp. 1204–1207.

[45] W. Hu, M. Hu, X. Zhou, T. Tan, J. Lou, and S. Maybank, "Principal axis-based correspondence between multiple cameras for people tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 663–671, 2006.

[46] N. Gheissari, T. Sebastian, and R. Hartley, "Person reidentification using spatiotemporal appearance," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1528–1535.

[47] J. Yu, J. Amores, N. Sebe, P. Radeva, and Q. Tian, "Distance learning for similarity estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 451–462, 2008.

[48] D. Gray and H. Tao, "Viewpoint invariant pedestrian recognition with an ensemble of localized features," in *2008 European Conference on Computer Vision*, 2008, pp. 262–275.

[49] K. Weinberger and L. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[50] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 713–719.

[51] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Advances in Neural Information Processing Systems*, 2003, pp. 521–528.

[52] D. Gray, S. Brennan, and H. Tao, "Evaluating appearance models for recognition, reacquisition, and tracking," in *IEEE International Workshop on Performance Evaluation for Tracking and Surveillance (PETS)*, vol. 3, 2007, pp. 5–12.

[53] NVIDIA Corporation, "CUDA C programming guide v4.0," 2011. [Online]. Available: http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf

[54] Khronos OpenCL Working Group, "The OpenCL specification," 2008.

[55] T. Yeh and Y. Patt, "Two-level adaptive training branch prediction," in *Proceedings of the 24th Annual International Symposium on Microarchitecture*, 1991, pp. 51–61.

[56] S. Vanderwiel and D. Lilja, "Data prefetch mechanisms," *ACM Computing Surveys (CSUR)*, vol. 32, no. 2, pp. 174–199, 2000.

[57] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Boston, MA: Addison–Wesley, 1995.

[58] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[59] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and trecvid," in *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, 2006, pp. 321–330.

[60] M. Dikmen et al., "Surveillance event detection," in *TRECVID Video Evaluation Workshop*, 2008, pp. 95–104.

[61] M. Folk, A. Cheng, and K. Yates, "Hdf5 a file format and io library for high performance computing applications," in *Proceedings of Supercomputing*, vol. 99, 1999.

[62] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick, "Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures," in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, 2008, pp. 4:1–4:12.

[63] V. Lee et al., "Debunking the 100x gpu vs. cpu myth: An evaluation of throughput computing on cpu and gpu," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, 2010, pp. 451–460.

[64] R. Weber, H. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the International Conference on Very Large Data Bases*, 1998, pp. 194–205.

[65] V. Podlozhnyuk, "Histogram calculation in CUDA," 2007. [Online]. Available: http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/histogram256/doc/histogram.pdf

[66] C. Nugteren, G. van den Braak, H. Corporaal, and B. Mesman, "High performance predictable histogramming on GPUs: Exploring and evaluating algorithm trade-offs," in *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units*, 2011, pp. 1:1–1:8.

[67] T. Cormen, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 2001.

[68] NVIDIA Corporation, "CUDA C best practices guide," 2010. [Online]. Available: "http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Best_Practices_Guide.pdf"