Queen Mary, University of London
School of Electronic Engineering & Computer Science

SUPERVISED DICTIONARY LEARNING FOR ACTION RECOGNITION AND LOCALIZATION

Thesis submitted to University of London
in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy

B. G. Vijay Kumar

Supervisor: Dr. Ioannis Patras

London, 2012.

Abstract

Image sequences with humans and human activities are everywhere. With the amount of produced and distributed data increasing at an unprecedented rate, there has been a lot of interest in building systems that can understand and interpret the visual data, and in particular detect and recognise human actions. Dictionary based approaches learn a dictionary from descriptors extracted from the videos in the first stage and a classifier or a detector in the second stage. The major drawback of such an approach is that the dictionary is learned in an unsupervised manner without considering the task (classification or detection) that follows it. In this work we develop task dependent(supervised) dictionaries for action recognition and localization, *i.e.*, dictionaries that are best suited for the subsequent task. In the first part of the work, we propose a supervised max-margin framework for linear and non-linear Non-Negative Matrix Factorization (NMF). To achieve this, we impose max-margin constraints within the formulation of NMF and simultaneously solve for the classifier and the dictionary. The dictionary (basis matrix) thus obtained maximizes the margin of the classifier in the low dimensional space (in the linear case) or in the high dimensional feature space (in the non-linear case). In the second part the work, we develop methodologies for action localization. We first propose a dictionary weighting approach where we learn local and global weights for the dictionary by considering the localization information of the training sequences. We next extend this approach to learn a task-dependent dictionary for action localization that incorporates the localization information of the training sequences into dictionary learning. The results on publicly available datasets show that the performance of the system is improved by using the supervised information while learning dictionary.

To my wife
Archana

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Ioannis Patras for his excellent guidance, intuitive comments, encouragement and feedback. This work would not be the same without his support. I also thank Dr. Irene Kotsia for being my collaborater for the first two years of my PhD. Her suggestions and insights have helped me a lot. I would also like to thank QMUL and EPSRC for supporting me with the PhD scholarship program (EP/G033935/1).

I am grateful to all my colleagues from Multimedia and Vision lab: Stefano Asioli, Karthikeyan Vaiapury, Sertan Kaymak, Heng Yang, Wei Wei, Markus Brenner, Daria Stefic, Dr. Naeem Ramzan, Ravi Garg, Giuseppe Passino, Thomas, Eduardo Peixoto, Saverio Blasi, Prathap Nair, Konstantinos Bozas, Dr Krishna Chandramouli, Dr Sander Koelstra for creating a vibrant and friendly working environment. I would also like to thank my friends Narayan KV, Sathyanarayan, Sharan and Hicham for their encouragement.

Finally, I would like to thank my parents, brother and sister-in-law for their love, understanding and constant, unconditional support.

Contents

List of Abbreviations

BO[V]W	Bag Of [Visual] Words
DoG	Difference of Gaussians
GMM	Gaussian Mixture Model
HOF	Histogram of Optical Flow
HOG	Histogram of Oriented Gradients
ISM	Implicit Shape Model
KNMF	Kernel Non-Negative Matrix Factorization
LDA	Linear Discriminant Analysis
LLC	Locality Constraint Linear Coding
LoG	Laplacian of Gaussians
NMF	Non-Negative Matrix Factorization
PCA	Principal Component Analysis
SIFT	Scale-Invariant Feature Transform
STIP	Spatio Temporal Interest Point
SURF	Speeded Up Robust Features
SUSAN	Smallest Univalue Segment Assimilating Nucleus
SVM	Support Vector Machine
VOC	(Pascal) Visual Object Challenge

List of Figures

Chapter 1.

Introduction

There has been a rapid growth in the generation, transmission and storage of video data over the past few years. This tremendous growth of data can be attributed to the advancement of the Internet technology, low cost digital cameras, smart phones, inexpensive disks, online storage sites, social networking sites etc. The statistics from the popular online video site, YouTube reveals that 72 hours of video is uploaded every minute, and over 4 billion hours of video are watched every month [1]. In addition, the forecast suggests that video traffic will be $\approx 55\%$ of all consumer Internet traffic by 2016 [2]. These figures indicate the growing interest of the users for video data. In particular, the videos with human actions have received much attention because of its potential applications in video retrieval, video surveillance, security, human-computer interaction, gaming and so on. In this thesis, we present supervised dictionary learning approaches for human action recognition and localization. The objective of the human action recognition system is to assign a label to the video based on the action content and the task of action localization involves the prediction of the spatial center, temporal extent and label of the action in the video.

Computers have already outperformed humans in terms of the computational ability. However, they are too far from matching the recognition capabilities of humans. The human visual system can efficiently recognize hundreds of human activities in the presence of complex conditions like background clutter, changes in view angle, intra/inter class variations and occlusion. Extending such capabilities to computers is one of the main goals of the human action recognition system.

[1] http://www.youtube.com/yt/press/statistics.html
[2] $http://www.cisco.com/$

1.1. Applications

In this section, we briefly discuss a few applications of human action recognition and localization systems.

Video Surveillance: One of the tasks in video surveillance is the process of monitoring human activities and behavior. The video surveillance system consists of a number of cameras constantly capturing the scenes at different locations of a building. These videos are regularly monitored by a human to detect any abnormal behavior or suspicious activity as shown in Fig. 1.1. Due to an increase in the concern over the safety and security, there is a huge rise in the number of surveillance cameras which makes the human monitoring tedious. This motivates for a system that can automatically detect human activities and abnormal behaviors in videos.

Figure 1.1.: A Human constantly monitoring the videos

Video Retrieval: As the amount of video data is increasing at an unprecedented manner, it is crucial to annotate, categorize and index the videos so that the retrieval is fast. At the same time, it is equally important to retrieve the relevant videos. As the manual annotation of videos is labour-intensive, most of the online video sites substantially depend on the user tags for video retrieval. However, due to inconsistencies in the interpretations of the users or unreliable tags, the retrieved video may not be relevant to the query. This demands for a system that can automatically annotate the videos.

Gaming: With the advancement of computer vision and graphics, the gaming sector has experienced a significant development. In contrast to the traditional gaming technology

where the characters or objects were controlled by controller devices, the current systems use a special sensor called Kinnect to sense the human body and uses tracking and action recognition systems to simulate the human actions on the screen as shown in Fig. 1.2.

Figure 1.2.: Human actions are simulated on the screen using a Kinnect sensor

1.2. Challenges

Unlike other multimedia contents, videos are typically of large size with redundant information. In order to ease the interpretation for machines, there is a need for compact representation of the video data. In addition, such representation and the system trained on these representation should cope with many real world scenarios such as intra and inter class variations, background clutter and occlusion. In this section, we briefly describe the challenges for these systems.

(a) (b)

Figure 1.3.: a) shows the images from the CMU dataset [49] where the subjects perform the action *pickup* in two different ways. b) shows the images having similar appearance but belong to different classes *two hand wave* and *jumping jacks* respectively

Intra-class variations: This represents the variations that occur within a class. An action can be performed by different people in different ways. Fig. 1.3(a) shows the

images form the CMU dataset [49] where the action *pick up* is performed by the subjects in two different ways. The recognition system should categorize these actions into the same class.

Inter-class variations: This represent the variations that occur between two classes. Two actions belonging to different classes can have similar appearance and may differ marginally in certain other aspects. For *e.g.* the actions *jog* and *run* have similar appearance but vary in the speed at which the actions are performed. Fig. 1.3(b) shows the images from two different actions *two hand wave* and *jumping jacks* having similar appearance. A good system should categorize the actions into respective classes.

(a) (b)

Figure 1.4.: a) shows videos having background clutter. b) shows actions where some parts of the subject are occluded

Background clutter: Generally, the video data obtained from the real world applications like surveillance contain moving or dynamic background. For *e.g.*, the videos captured by a camera in a public place contain moving vehicles, moving people, changing background as shown in Fig. 1.4(a). This poses a huge challenge for the systems to extract foreground from the background. A robust system should be able to handle this.

Occlusion: This is the terminology used when some parts of the action of interest are hidden or covered by some other objects. Hence some part of the data is non informative and act as noise. Fig. 1.4(b) shows a few instances of occlusion.

1.3. Contributions:

The main goal of the thesis is to learn the dictionaries in a supervised manner for human action recognition and localization. In the first part of the work, we propose a supervised learning framework for a feature selection technique called Non-Negative Matrix Factorization (NMF) and use the extracted features to classify the actions. Summarizing, the main contributions of this part are:

- We propose a max-margin framework for Semi Non-negative Matrix Factorization (MNMF) which incorporates the maximum margin constraints within the Semi-NMF formulation in order to obtain the basis vectors that can efficiently discriminate the features belonging to different classes.
- An optimization scheme that simultaneously solves for the separating hyperplane of the max-margin classifier and the basis matrix. The constrained optimization problem of the proposed framework is non-convex with respect to the unknown parameters. We employ an iterative procedure where at each iteration we solve a set of convex (quadratic or SVM-type) subproblems. Each of those convex subproblems results from the original one when we fix some of the unknown parameters.
- We extend the above framework to the case of nonlinear NMF, *i.e.*, KNMF. For this, we incorporate the max-margin constraints into the formulation of KNMF such that the resulting bases maximize the margin of the svm classifier (nonlinear) in the reconstructed feature space.

In the second part of the thesis, we propose a supervised dictionary learning framework action localization. We use a part based model called the Implicit Shape Model (ISM) which internally uses k-means for learning dictionary of the parts. The work was motivated by the fact that the dictionaries learned in an unsupervised manner (k-means) may not be optimal for the task of action localization. In order to adapt the dictionary for the task of localization, we learn weights for the dictionary and then we extend this approach to learn a task-dependent (supervised) dictionary. For this part, the main contributions are:

- We design an approach to learn local weights for the matched codewords of each feature based on the degree of match between the codeword and the feature. This is achieved in a principled way by incorporating the Hough voting scheme into Locality Constrained Linear Coding (LLC) [118] framework.
- We develop a framework that enables us to measure the discriminative response at the output Hough space of the training sequences. We use this framework to quantify the contribution from each dictionary atom to the location of the action center or the hypothesis.
- We propose a discriminative weighting scheme to learn the global weights for the dictionary that maximize the Hough voting response at the spatio-temporal location of the activity compared to background of the training set.

- We extend the above technique and design a framework that enables supervised learning of the dictionary for the ISM, *i.e.*, the proposed algorithm enables us to incorporate the localization information into the dictionary learning. This is in contrast to the ISM [60] that only uses the appearance information of the training descriptors to learn the dictionary. To the best of our knowledge, this is the first attempt to use the output Hough space information for dictionary learning.
- We employ the above framework to learn a task dependent dictionary which is optimized for the detection task, *i.e.*, we use the localization information of the training sequences to learn a discriminative dictionary that maximize the response at the spatio-temporal location and extend of the activity compared to background.
- We also extend the above approach to include the background information into the dictionary learning which results in a dictionary that can discriminate between the descriptors extracted at the foreground from that of the background.

1.4. Thesis Organization

This report summarizes the work carried out during my PhD. The rest of the thesis is organized as follows.

Chapter 2 presents a brief survey of the related work in the field of action recognition and localization. The first part of the chapter is dedicated to action recognition where we categorize the methods into two classes namely global and local methods based on the type of action representation and describe few popular algorithms. The second part of the chapter briefly describes a few methods for action localization. The methods are broadly classified into two categories namely pattern search and Hough voting methods and a few algorithms in the respective categories are described. Finally the action datasets used in the thesis are discussed.

Chapter 3 starts with a motivation for the supervised feature selection transform. A brief overview of the works related to NMF are discussed. The remainder of the chapter presents the methodologies for the supervised feature selection: incorporation of discriminate information from the classifier to Semi-NMF, Solving for bases and classifier, extension to the nonlinear version of the NMF. Finally the evaluation of the proposed method on several computer vision datasets are presented.

Chapter 4 proposes a dictionary weighting approach for action localization. The first part of the chapter gives an overview of the part based model called the ISM. It investigates the assignment of a local features to the nearest codewords and presents a coding method based on LLC. A framework that quantifies the output Hough space information at the foreground and background and how we use the framework to learn a discriminative global weighting scheme for the dictionary is also described. The performance of the proposed method are compared with the baseline method and state-of-the-art.

Chapter 5 describes the shortcomings of the dictionary weighting approach and proposes a framework for the incorporation of the output localization information into dictionary learning. The details of using the framework to include a discriminative information from the training sequences is presented. An approach to include the background model into the learning of the dictionary is also presented.

Chapter 6 summarizes and concludes the work. The chapter also presents the future directions of the work.

List of Publications

[1] B. G. Vijay Kumar, Irene Kotsia, and Ioannis Patras. Max-margin non-negative matrix factorization. In *Image and Vision Computing*, pages 279–291. Elsevier, 2012.

[2] B. G. Vijay Kumar and Ioannis Patras. Supervised dictionary learning for action localization. In *IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, 2013. [oral].

[3] B. G. Vijay Kumar and Ioannis Patras. Learning codebook weights for action detection. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012*, pages 27–32. IEEE, 2012.

[4] B. G. Vijay Kumar, Irene Kotsia, and Ioannis Patras. Max-Margin Semi-NMF. In *Proceedings of the BMVC 2011*, pages 129.1–129.11. BMVA Press, 2011.

[5] B. G. Vijay Kumar and Ioannis Patras. A discriminative voting scheme for object detection using hough forests. In *Proceedings of the BMVC 2010*, pages 3.1–3.10. BMVA Press, 2010.

Chapter 2.

Related Work

Action detection and recognition has been one of the most active topics of research in the computer vision community. A number of methods have been proposed in literature to solve this problem. In this chapter, we review a few methods for action detection and recognition and describe the datasets used in our experiments. We discuss action recognition methods in first part of the chapter and then proceed to action detection.

2.1. Action Recognition

State-of-the-art methods have been presented in several survey articles [5, 93, 112, 121]. The structure presented in this chapter is similar to [93]. Action recognition methods can be broadly classified into two categories namely:

- **Global representation:** Extracts the region of interest (human body) in the video using background subtraction or tracking. The extracted region is represented using features and a model is learned on these features. The global methods are powerful but heavily depend on accurate background subtraction or tracking for efficient representation of the features and fails under uncontrolled conditions. Further, these methods are also sensitive to occlusion, view point changes.
- **Local representation:** In contrast to global representation, the local representation model the observation as a collection of local parts or descriptors. By nature, the local representations are less sensitive to view point changes, occlusion and in most of the cases do not require background subtraction or tracking.

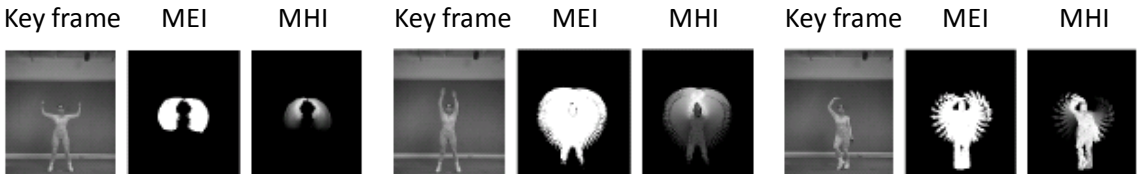


Figure 2.1.: The key frames and corresponding MEI and MHI for three different actions

2.1.1. Global Representation

In this section, we discuss the algorithms that follow the global representation for action recognition. The global representation describe the region of interest as a whole without any notion of local parts. Thus it requires background subtraction or tracking to extract the region of interest. These methods can be broadly classified into two categories namely Silhouette based and Optical flow based methods.

Silhouette based methods

Bobick *et al.* [13] proposed a representation of action called *temporal template* that has two components namely *motion-energy image* (MEI) and *motion-history image* (MHI). MEI indicates where motion has occurred in an image sequence and MHI is an image where the values of the image are a function of recency of motion as shown in Fig. 2.1. The temporal templates are compared using Hu moments.

The method in Sullivan and Carlsson [108] stored a set of key frames to represent an action. These key frames are matched to the frames of an image sequence to obtain a point to point correspondence between them. This correspondence is used to transfer the body part locations from the key frame to the actual frame which can then be used for tracking as well as reanimation of the sequence.

Yilmaz *et al.* [128] generated a 3D spatio-temporal volume (STV) by stacking only the object regions from the frames of the image sequences. A set of such STVs are called *action sketch* for a category of actions. A set of descriptors corresponding to the changes in direction, shape and speed of the parts (of contours) are computed using differential geometry. These descriptors are categorized based on the sign of Gaussian and mean of curvatures.

Gorelick *et al.* [41] followed a similar approach to extract space time shapes but the descriptors such as local space-time saliency, action dynamics, shape structure, and orientation are extracted from the 3D shapes using Poisson equation. The space-time

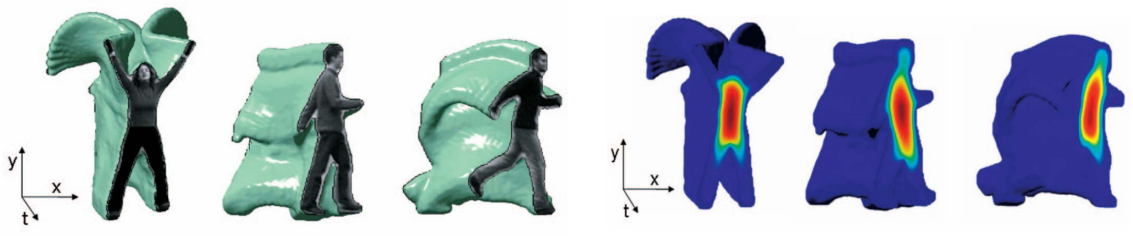


Figure 2.2.: The space-time shapes and the solution of Poisson equation for space-time shapes

shapes and the solution of Poisson equation of space-time shapes for three action are depicted in Fig. 2.2. During testing, the descriptors extracted from the shapes of the test sequence are matched with the training shape-time descriptors in a sliding window manner using k-nearest neighbor.

The action recognition method in [107] used R transform to extract shape descriptors from silhouettes of the action. The main goal of the method is to represent the appearance of an action from a single camera as a function of the view point of the camera, however, it suffers from self occlusion due to the use of silhouettes based representation.

Optical Flow based Methods

These methods employ the descriptors obtained by the optical flow. The algorithm in [31] describe a descriptor called *spatio temporal motion* descriptor which is an aggregate set of features sampled in space and time, that describe the motion over a small time period. The optical flow vector field from a figure centric sequence are extracted and are split into horizontal and vertical components. These components are further quantized into four non-negative channels and blurred with a Gaussian to form the final descriptor. The classification is done using a nearest neighbor classifier by matching the descriptors with a database of pre-classified actions.

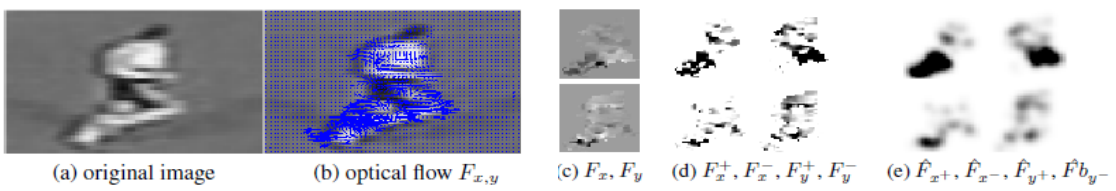


Figure 2.3.: Computation of the low level descriptors in [33]

Fathi *et al*. used a similar approach to extract the *spatio temporal motion* descriptors but added another motion channel to the 4 channels mentioned in [31]. The construction of low level features are shown in Fig. 2.1.1. The authors report that the low level features at individual locations are not capable of discriminating between two classes and hence they divide the figure centric volume into subvolumes and apply adaboost classifier to the subvolumes to obtain a mid level descriptors. A final descriptor is obtained by merging the mid level features which are again classified using the adaboost classifier.

Another method involving flow is proposed by [94] where the spatio-temporal regularity flows (SPREF) are used as the features. The authors propose a template based method for action recognition based on *Maximum Average Correlation Height* (MACH) filter which can efficiently capture the intra-class variability. The SPREF are incorporated into the synthesis of MACH filter using Clifford Fourier Transform, a generalization of traditional Fourier transform to vector fields. The high computational cost commonly incurred in template based methods is avoided by analyzing the filter response in frequency domain.

Yan Ke *et al*. [48] introduced a volumetric feature framework for action detection by extending Viola and Jones work [115] in spatial domain (2D) to spatio-temporal domain. The framework is applied on the optical flow extracted from a video sequence to compute the box features. Also the authors developed a data structure called *integral video* to efficiently compute the box features which could achieve a real time performance.

Schindler and Van gool [98] used a short sub-sequences of the video called *Action Snippets* instead of the entire video. The method use a hybrid approach where both shape (Gabor filter responses) and motion (optical flow) information are extracted from the snippets. The extracted filter responses are max-pooled and concatenated to obtain a final descriptor and compared against the learned action templates.

2.1.2. Local Representation

For Local representation, the local space-time descriptors are extracted at different scales from local regions of the video. Since the local descriptors are extracted independently, the representation is robust to view point changes and occlusion and also does not require any background subtraction or tracking. In this section, we briefly review a few algorithms that employ the local representation for action recognition.

In [53], the notion of interest points in spatial domain is extended to spatio-temporal domain, in particular the authors build on the idea of Harris interest point operator [42]. The interest points are detected using the eigen values of the second moment matrix. They analyzed the importance of scale (both spatial and temporal) for the detection of interest points and proposed a method for simultaneous estimation of scale parameters. They illustrated the robustness of spatio-temporal interest point by detection and pose estimation of walking people in presence of occlusion and highly cluttered dynamic background. Fig. 2.4 shows the detected interest points for walking action. A local svm classifier was used in [101] to classify actions belonging to six categories of KTH dataset where the local features were obtained using the spatio-temporal descriptors described in [53].

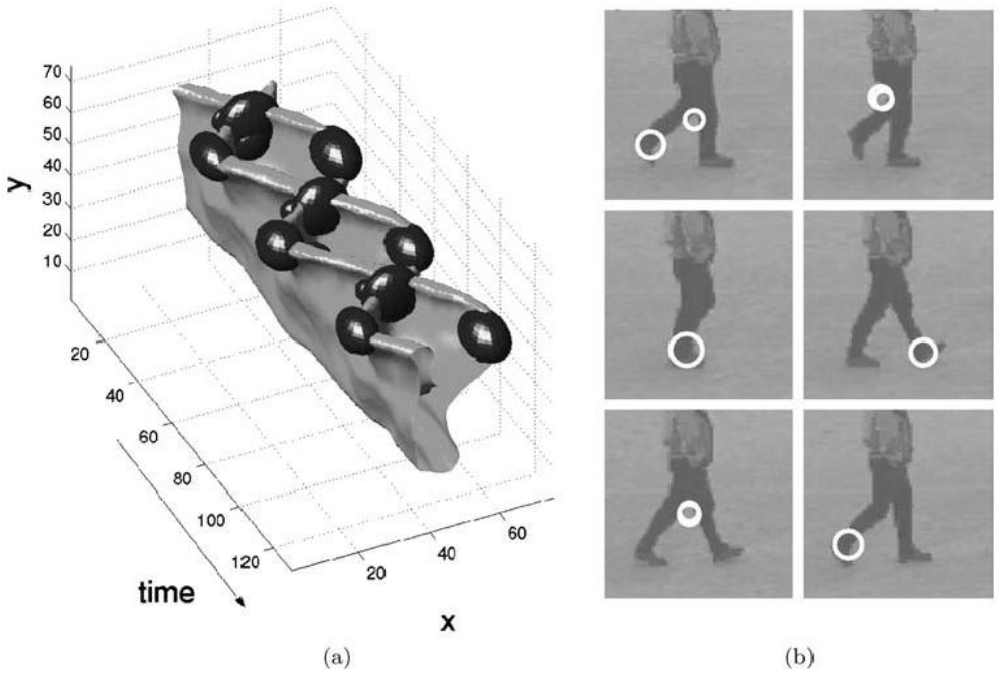


Figure 2.4.: Spatio-temporal interest point detection in [53]: a) 3D plot of leg pattern (shown upside side down), b) detected interest points

Piotr *et al.* [30] claim that the spatio-temporal corners are rare in videos like rodent behavior recognition or facial expression even when there is a significant motion and hence the spatio-temporal descriptor in [53] may not give good performance. They proposed to detect interest points by applying Gaussian and Gabor filter on spatial and temporal domains respectively and then choosing the local maxima of the response. The

descriptors were obtained by extracting PCA-SIFT features from the cuboids around the interest point locations.

A temporal extension of the salient feature detector in [47] is proposed by Oikonomopoulos *et al.* in [88]. They consider cylindrical neighborhood around each location of the video and compute entropy at different scales. The candidate scale is selected as the scale at which entropy has a local maxima. The salient regions are obtained by clustering the spatio-temporal points with similar location and scale.

Greet *et al.* [122] proposed a method to detect dense spatio-temporal interest points. They used the determinant of the Hessian matrix to compute the saliency at each point. The scale selection is done using non maxima suppression to obtain extrema on all five dimensions (space, time and scale) and they employed *integral video* data structure to efficiently compute the Hessian. SURF descriptors at the detected interest points are used as features for the classification of actions.

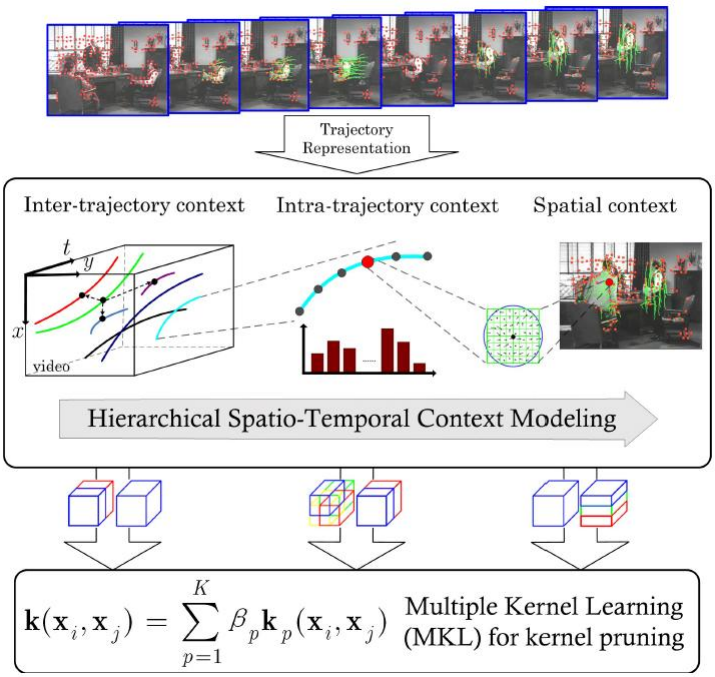


Figure 2.5.: Schematic diagram of hierarchical spatio-temporal context modelling

Trajectory based methods:

Trajectories are obtained by tracking spatial interest points along time. In [82], a framework similar to BOW is employed using trajectories. Trajectories are obtained using a KLT tracker and a dictionary of trajectories is learned by clustering the trajectories. For

a given test video, the trajectories are extracted and assigned to the nearest trajectory center from the trajectory dictionary. The labels are accumulated in a k dimensional vector which are normalized and used as a feature vector for the video. These feature vectors are classified using SVM.

Messing *et al*. [83] tracked local features and then quantized the velocity over time using log-polar coordinates with 8 bins for direction and 5 bins for magnitude. These are called *velocity history* that form the basic feature. The activities are modeled using a generative mixture model.

A framework to capture the spatio-temporal context in a hierarchical way is described in [109]. The trajectories are obtained from a video using KLT and three types of features are extracted. They are: 1) Point-level context (SIFT average descriptor), 2) intra trajectory context (trajectory transition) and inter trajectory context (trajectory proximity descriptor). The first two contexts are good at describing the solo actions (action with one object) whereas the third context is good for actions with more than one object, eg: handshaking, kissing. The descriptors are encoded using a Markov process and finally classified using multi channel non linear SVMs. This is depicted in Fig. 2.5

In [117], dense trajectories are employed to represent an action. Dense points from each frame are tracked based on the displacement information and optical flow field. The trajectories are encoded using a novel descriptor based on the motion boundary histograms which is robust to camera motion. The videos are classified with a bag-of-features approach.

2.2. Action Localization:

In this section, we discuss a few methods for action localization. We broadly classify the methods into two categories namely pattern search methods and voting based methods.

2.2.1. Pattern search methods

These methods consider the action as a whole without any notion of parts. The video is represented as a collection of spatio-temporal features and the subvolume containing maximum response (local maxima) is searched.

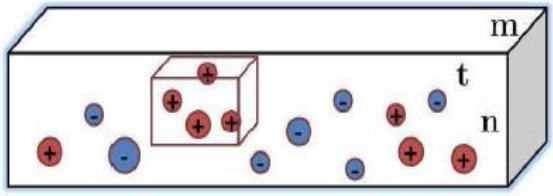


Figure 2.6.: The circles shows the STIPs with the red and blue corresponding to an action class and background respectively. The highlighted subvolume shows a region corresponding to an action where there is high response

In [132], the query video consisting of a group of STIPs are matched with the positive and negative training STIPs using a classification scheme called naive Bayes mutual information maximization (NBMIM) thus making it a discriminative matching. To search for the subvolume containing maximum STIP score (shown in Fig. 2.6), they decouple the spatial and temporal space and apply branch-and-bound and dynamic programming to the spatial and temporal domain respectively.

Inspired by the sliding window search for objects in 2D images, the method in [104] used a sliding window based search for detecting actions in video sequences. They extracted trajectories from the videos and encoded it using the method in [109]. The five channel descriptor is used to represent an action within a subvolume which are then classified using SVM. They also proposed a method using greedy k nearest neighbor algorithm which can automatically annotate the positive training data.

The method in [22] proposed a cross dataset action detection algorithm where the model is trained on one dataset and tested on another dataset. The video is represented as a collection of STIPs and they employ Gaussian Mixture Models (GMM) to learn the prior distribution of the STIPs. The detection is carried out in a computational efficient manner using branch-and-bound algorithm that finds a subvolume with maximum STIP response.

An algorithm similar to [132] is proposed in [130] where an unsupervised random forest is used to fastly compute the mutual information between the query and the training STIPs. In order to facilitate faster detection of the action, a coarse to fine subvolume search is used as opposed to branch-and-bound search in [132]. An interactive search mechanism is also incorporated by which user can incrementally add labeled examples to the query set.

Konstantinos *et al.* [27] applied Gaussian third derivative filters to decompose the video into a distributed representation according to 3D spatio-temporal orientation. The corresponding distribution of oriented energies of the template and the search video are efficiently matched in a sliding window approach and a significant local maxima in the similarity volumes are identified.

Yan Ke *et al.* [49] proposed a part-based matching method for action detection where the volumetric representation of the template is matched against the over-segmented spatio-temporal video volumes. The authors augment the shape based descriptors with the flow descriptors which can be computed in the presence of cluttered background with figure/ground separation. In contrast to the above methods where the whole template of the action is matched, the authors propose a method to match the manually constructed parts of the action that makes the matching robust to occlusion.

2.2.2. Hough voting methods

These methods employ a part based approach where the parts of the action vote for the center, scale and extend of the action. The votes are collected in an output Hough space where the location of the maxima corresponds to the location of action.

Oikonomopoulos *et al.* [87] extended the spatial Hough voting approach in images [61] to spatio-temporal voting in videos. During training, a set of codebooks for each category of the action is constructed using Gentleboost. The codebooks store the occurrence distribution of the codewords that encode the location and scale at which each codeword is activated during training. During testing, the detected local features are matched against the codebooks and activated codewords cast votes to the spatio-temporal center and extend of the action.

A similar Hough transform based scheme is proposed in [127] where the codebooks are constructed using an adaptation of Hough voting to Random forest called Hough forest. The trees of the Hough forest map the densely sampled features in the videos to their corresponding votes in the Hough space. The authors apply Hough transform based people detection [38] to detect the hypothesis in each frame. The extracted hypothesis are linked using particle filter to form action tracks. Subsequently the action tracks vote for the action label and center.

2.3. Datasets

We present in this section, some of the datasets used to evaluate our method.

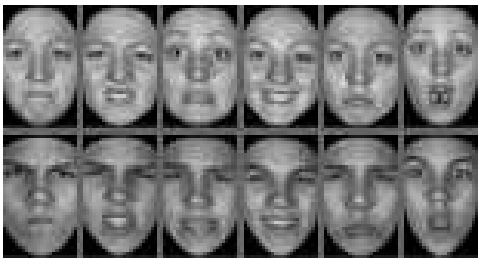


Figure 2.7.: A sample of the images used from the BU-3DFE dataset.

INRIA Pedestrian Dataset: The INRIA pedestrian dataset [26] consists mostly front and background views of human. The dataset includes several variations caused by partial occlusions and scale, pose, clothing and illumination changes. For our experiments, we created a set of positive examples (containing pedestrians) and another one of negative examples (by sampling the background). The extracted bounding boxes were cropped to a size of 51×100 pixels.

BU-3DFE Facial Expression Dataset: The BU-3DFE facial expression dataset [129] consists of 100 subjects (about 60% female and 40% male). The dataset contains six different facial expressions (anger, disgust, fear, happiness, sadness, surprise) performed by each subject at four intensities plus an image of the neutral state, captured at 5 yaw angles. In our experiments, we only used the 2D images of the frontal view. The original images were cropped and down sampled to a size of 24×40 pixels. Fig. 2.7 shows a set of sample images used for the experiments.

Mediamill Dataset: The Mediamill [106] is a dataset used for object classification. The dataset consists of 43907 sub-shots of 101 classes. Each image was represented using a 120-dimensional feature vector. We randomly chose two object categories from the available 101 ones and performed a binary classification task.

KTH Dataset: The KTH dataset [101] consists 600 videos of six human actions (*walking*, *jogging*, *running*, *boxing*, *hand waving*, *hand clapping*) performed by 25 subjects under 4 scenarios: outdoor $s1$, outdoor with scale variations $s2$, outdoor with different clothes $s3$ and indoors $s4$. The sequences are downsampled to a spatial resolution of 160×120. The background is static and homogeneous in most of the sequences. Fig. 2.8 shows different actions of the KTH dataset.

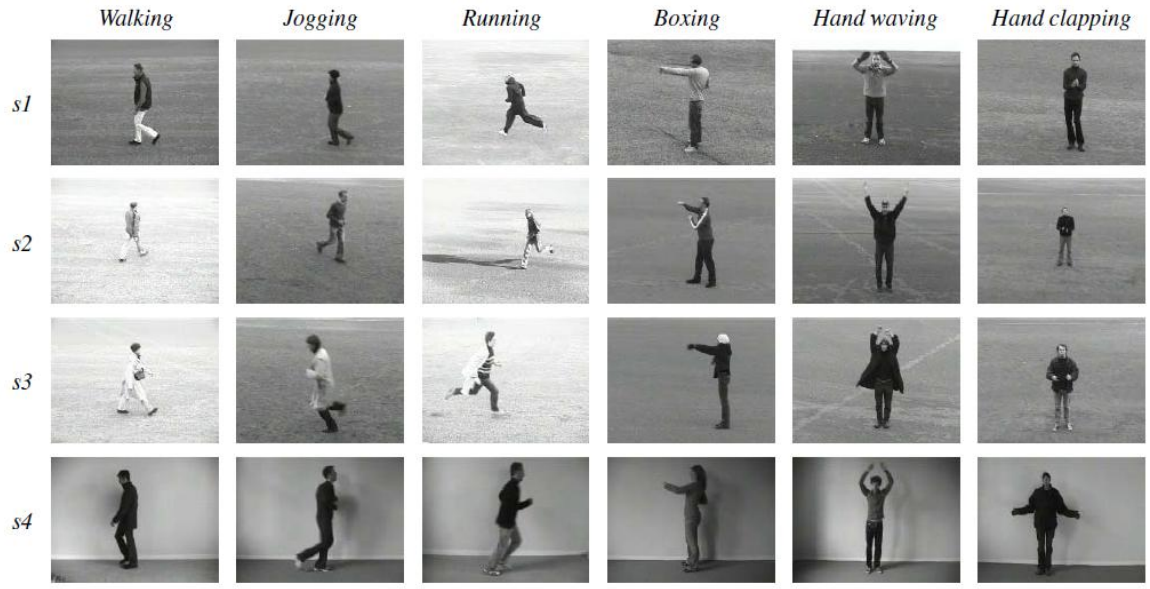


Figure 2.8.: Six different actions of the KTH dataset

CMU Dataset: The CMU dataset [49] consists of five actions (*pickup*, *onehand wave*, *twohand wave*, *jumping jacks*, *push button*). The videos were acquired using a hand-held camera in cluttered environments with moving people or cars in the background. The dataset is designed to evaluate the performance of the algorithm in crowded environments. The dataset consists of 110 events with resolution 160×120. The dataset is challenging as there are variations in how the subjects performed actions and in the background. The dataset also has significant variations in both spatial and temporal scales.

Chapter 3.

Max-Margin Non-negative Matrix Factorization

3.1. Introduction

Representing a data vector (signal) as linear combination of a set of basis vectors (signals) is one of the most popular techniques used in signal processing, machine learning and statistics. The set of basis vectors are referred to as a *dictionary* [74] and each of the basis vectors is called an *atom* of the dictionary. The dictionary can be a set of pre-defined bases such as wavelets [80] or a data-dependant dictionary [6] in which the dictionary atoms are learned from a training set. Among these the latter has received more attention in the computer vision community recently due to its applications in denoising [6,77] and image classification [75,76,135]. Non-Negative Matrix Factorization (NMF) [57, 64] is a special case of the dictionary learning [7, 28, 74] where the size of the dictionary (the number of basis vectors) is less than the dimension of the input data and also has positivity constraints on the dictionary and the codes. Learning the bases (dictionary) adaptive to the data can significantly improve the reconstruction accuracy of the data as compared to the case where pre-defined bases [6]. However, these representative characteristics of the data may not be optimal for a classification task, where the objective is to learn a discriminative dictionary, that can be used to distinguish the data belonging different categories. In this work, we develop a method to learn the bases of NMF in a supervised manner by considering the sample labels along with the data. We incorporate the discriminative information obtained by an svm classifier into the dictionary such that the representation obtained by such dictionary maximizes the margin of the svm classifier.

3.2. Related Work

The Non-Negative Matrix Factorization (NMF) algorithm is one of the most popular Machine Learning techniques for finding parts-based representations of the non-negative data. It has been widely used in several computer vision applications such as image retrieval, face and gesture recognition, object detection and action recognition. NMF decomposes the data matrix into non-subtractive combinations of non-negative bases [64]. Its ability to produce parts-based representations has been theoretically justified and experimentally demonstrated in [57]. By contrast, other dimensionality reduction methods, such as the Principal Component Analysis (PCA) [103] result in bases and projection coefficients that can take either positive or negative values.

NMF was initially proposed in [57, 89]. In both approaches, the bases and coefficient matrices were obtained by minimizing the reconstruction error, that is the discrepancy between the approximation obtained by the matrix factorization algorithm and the original data. The reconstruction error was quantified either using the Kullback-Leibler divergence [57] or the least squares error [90]. In [58] the authors proposed an efficient implementation that uses a set of multiplicative update rules that were derived from the optimization of an upper bound of the cost function. In [66] the authors showed that the minimization of the upper bound indeed reduces the cost function but does not guarantee the convergence of the algorithm to the stationary point of the original optimization problem. In [65] the authors proposed two projected gradient-based methods for NMF that exhibited strong optimization properties. Motivated by the fact that the multiplicative update rules for computing the factor matrices converge slowly and aiming at reducing expensive NMF update steps, a few matrix initialization techniques that ensured rapid error reduction rate and faster convergence were proposed in [18].

Although NMF usually results in a part-based representation, its various parts are not always well-localized. In order lo better localized (sparse) representation, local constraints were imposed along with the non-negativity constraints [23,64]. Several other algorithms aiming to achieve sparsity with tunable parameters were also developed [44, 70,91]. In [44,70], sparseness constraints were imposed on the elements of the coefficient matrix and a parameter was used to control the trade-off between the sparseness and the accuracy of the reconstruction. Such methods have an implicit control over the degree of sparseness. The approaches presented in [91] impose explicit sparseness constraints on both the base and coefficient matrices, allowing in that way an explicit control on the degree of sparseness.

The fact that NMF leads to a low-rank approximation of the data makes it suitable for subspace learning, that is for embedding high dimensional data into a low dimensional subspace. In this context, it has been extensively used for facial analysis including detection [23], recognition [64], verification [133] and expression recognition [50, 119]. Several other applications of NMF in Computer Vision include pose estimation [3], action recognition [96, 111], object recognition [69], subspace learning [20] and clustering [29].

In [3], NMF bases and coefficients were learned using a set of features extracted from clutter-free images containing objects. In [96], the NMF coefficients were extracted using appearance features and motion vectors. These coefficients were subsequently used to train a cascaded Linear Discriminant Analysis (LDA)-based classifier. The method presented in [111] followed an approach similar to [3] for the detection of humans in image sequences, where NMF was employed to learn a set of pose primitives. In [69], two approaches were followed in order to improve the recognition rate using features extracted by NMF. The first used a Riemannian metric for the learned feature vectors instead of the classic Euclidean distance, while the second orthonormalized the NMF bases and then used the features projected onto these bases. The authors in [20] introduced the Graph Regularized NMF (GNMF) that modelled the data subspace as a sub-manifold embedded in an ambient space. By learning NMF on such a manifold, GNMF showed better discriminative ability when compared to NMF that only considers the Euclidean space. Semi-NMF was introduced in [29] for clustering by relaxing the non-negativity constraints on the bases matrix. This lead to a bases matrix that contained the cluster centers and non-negative coefficients that can be interpreted as cluster indicators. A non linear extension to NMF, the so-called Kernel NMF (KNMF), was presented in [134].

Only few NMF-based works obtain the matrices in a supervised manner, that is, by utilizing the label information of the samples. In [133] the authors introduced discriminative constraints in order to extract bases that correspond to discriminative facial regions for the problem of face recognition. The proposed Discriminant NMF (DNMF) [133] results in bases corresponding to salient facial features such as eyes and mouth, that are useful for discrimination. The authors in [50] proposed the Projected Gradients DNMF (PGDNMF) algorithm for facial expression recognition which extends DNMF in two major ways. First, projected gradients were used instead of multiplicative update rules in order to guarantee the convergence of the algorithm to a limit point that is also a stationary point of the original optimization problem. Second, discriminant analysis was employed on the classification features and not on the reconstructed data. In both of the above mentioned approaches the discriminant constraints were introduced in the

cost function, yielding in this way discriminative bases. However, the introduced constraints were tailored for a rather simplistic LDA-based classifier. Here, we propose a method in which the acquired projections are chosen so that they maximize the discriminative ability of a Support Vector Machine (SVM) classifier, a fact that results in higher classification performance as will be demonstrated in the experimental results section.

In this chapter, we first introduce soft max-margin constraints to the objective function of NMF in order to obtain a bases matrix that will enable us to extract features that maximize the classification margin. More precisely, in the proposed scheme we optimize a weighted combination of the reconstruction error term, that is used in the typical NMF formulations, and the cost function, that is used in typical SVM formulations, under SVM-type linear inequality constraints. The optimization is performed with respect to the unknown bases, the projection coefficients and the parameters of the separating hyperplane and is solved in an iterative manner, where at each iteration we solve only for a subset of the unknown parameters while keeping the others fixed. The resulting sub-optimization problems are either instances of Quadratic programming with linear inequality constraints or classical SVM-type problems. We proceed with extending the above framework to include the nonlinear version of NMF (KNMF) [134]. In that way we are able to obtain a bases matrix that maximizes the classification margin of the classifier in the reconstructed high dimensional feature space. The proposed method is applied to publicly available databases (the INRIA pedestrian, the BU-3DFE, the KTH action and the Mediamill datasets) where we demonstrate that it consistently outperforms SVM classification schemes that use features extracted using Semi-NMF [29], KNMF [134] and DNMF [133].

Summarizing, the main contributions of this chapter are

- A max-margin framework for Semi Non-negative Matrix Factorization (MNMF) is proposed, which incorporates the maximum margin constraints within the Semi-NMF formulation, in order to jointly find both the factorization matrices and the separating SVM hyperplane.
- An optimization scheme that solves simultaneously for the separating hyperplane of the max-margin classifier and the factorization matrices. The constrained optimization problem of the proposed framework is a non-convex one with respect to the unknown parameters. We propose an iterative procedure where at each iteration we solve a set of convex (quadratic or SVM-type) subproblems. Each of those

convex subproblems results from the original one when we fix some of the unknown parameters.

- We extend the above framework to the case of nonlinear NMF, *i.e.*, KNMF. For this, we incorporate the max-margin constraints into the formulation of NMF such that the resulting bases maximize the margin of the svm classifier (nonlinear) in the reconstructed feature space.

The rest of the chapter is organized as follows. In Section 3.3.1, we briefly describe the NMF and Semi-NMF algorithms. In Section 3.3.3, we formulate the proposed max-margin framework for semi-NMF and present an algorithm that solves the corresponding optimization problem. In Section 3.3.5, we discuss convergence issues of the proposed MNMF algorithm. In Section 3.4, we introduce the Kernel NMF algorithm. In Section 3.5 we present experimental results on several Computer Vision and Multimedia problems using publicly available datasets. Finally, we conclude in Section 3.6.

3.3. Max-Margin Semi-NMF (MNMF)

In this section, we give a brief overview of the NMF and its variant semi-NMF algorithm for matrix decomposition and proceed with formulating the proposed maximum margin NMF framework.

3.3.1. Non-negative Matrix Factorization

Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ represent a non-negative matrix having n examples in its columns. The NMF algorithm [57] decomposes $\mathbf{X}$ into two non-negative matrices, the bases matrix $\mathbf{G} \in \mathbb{R}^{m \times k}$ and the coefficients matrix $\mathbf{H} \in \mathbb{R}^{k \times n}$ such that $\mathbf{X} \approx \mathbf{GH}$, *i.e.*, $\mathbf{X}$ is approximated by the product $\mathbf{GH}$. In order to obtain a low dimensional representation of the data, k is typically chosen to be small, $(< \min(m, n))$. The columns of $\mathbf{G}$ can be regarded as the bases vectors and thus each example can be represented as the linear combination of those bases vectors as $\mathbf{x}_i \approx \mathbf{G}\mathbf{h}_i$. Here $\mathbf{x}_i$ and $\mathbf{h}_i$ are the i^{th} columns of $\mathbf{X}$ and $\mathbf{H}$, respectively. From now onwards we will use the notation $\mathbf{H} \succeq 0$ to express that the elements of the matrix $\mathbf{H}$ are non-negative. Fig. 3.1 illustrates the reconstruction of face images using NMF, Vector Quantization (VQ) and PCA. NMF reconstructs the image using a sum of non-subtractive bases, *i.e.*, both the bases and coefficients are positive.

VQ reconstructs the image by choosing the nearest basis and PCA reconstructs the image by taking a linear combination (both +ve and -ve) combination of the bases. From Fig. 3.1 it is evident that NMF results in part-based representation whereas VQ and PCA results in holistic representation.

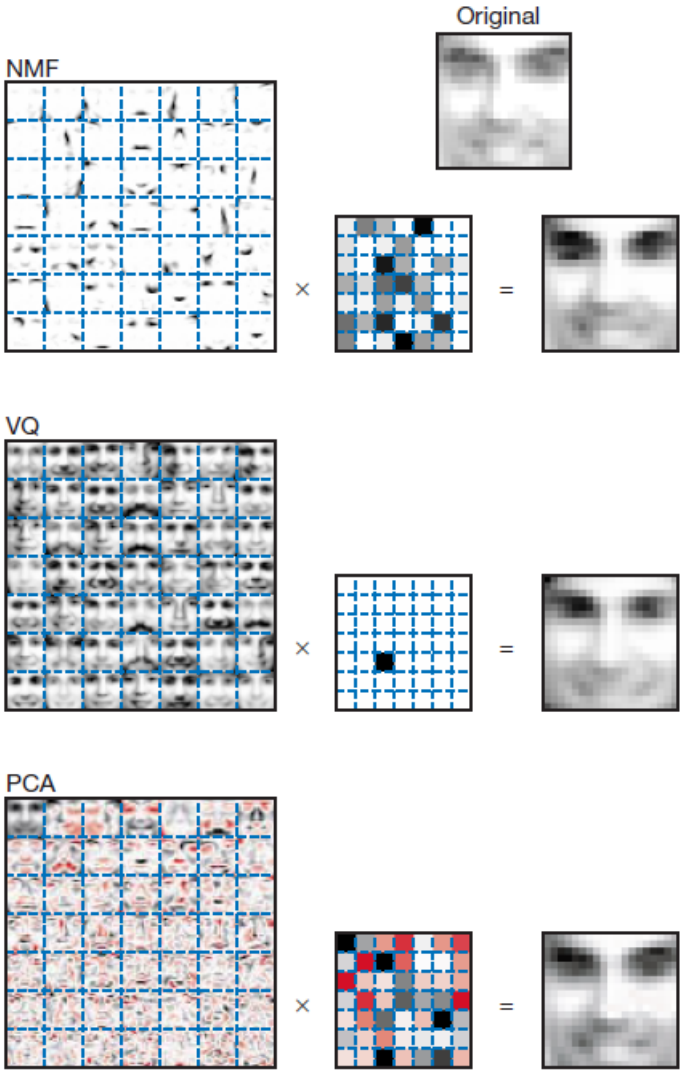


Figure 3.1.: Comparison of NMF bases with Vector Quantization(VQ) and PCA [57]: The three learning methods were applied to a database of faces containing 2429 images with size 19×19 pixels. The number of bases is restricted to 49 which is shown by 7×7 grid on the left side. A small grid on the right side shows the corresponding coefficients for each of the bases in the reconstruction. Positive values are indicated with black color and negative values with red color. NMF results in a part-based representation whereas the VQ and PCA results in a holistic representation.

The base matrix $\mathbf{G}$ and the coefficient matrix $\mathbf{H}$ are obtained by minimizing the reconstruction error $\|\mathbf{X} - \mathbf{GH}\|_F^2$ or the divergence $\mathrm{D}(\mathbf{X}||\mathbf{GH})$ w.r.t. $\mathbf{G}$ and $\mathbf{H}$:

$$\underset{\mathbf{G}\succeq 0,\mathbf{H}\succeq 0}{\arg\min}\|\mathbf{X} - \mathbf{GH}\|_F^2 = \underset{\mathbf{G}\succeq 0,\mathbf{H}\succeq 0}{\arg\min}\sum_{ij}\left(\mathbf{X}_{ij} - \mathbf{G}_{ij}\mathbf{H}_{ij}\right)^2 \tag{3.1}$$

which yields the multiplicative update rules for $\mathbf{G}$ and $\mathbf{H}$

$$\mathbf{G}_{ij} \leftarrow \mathbf{G}_{ij}\frac{(\mathbf{XH}^T)_{ij}}{(\mathbf{GHH}^T)_{ij}}, \qquad \mathbf{H}_{ij} \leftarrow \mathbf{H}_{ij}\frac{(\mathbf{G}^T\mathbf{X})_{ij}}{(\mathbf{G}^T\mathbf{GH})_{ij}} \tag{3.2}$$

or

$$\underset{\mathbf{G}\succeq 0,\mathbf{H}\succeq 0}{\arg\min}\,\mathrm{D}(\mathbf{X}||\mathbf{GH}) = \underset{\mathbf{G}\succeq 0,\mathbf{H}\succeq 0}{\arg\min}\sum_{ij}\left(\mathbf{X}_{ij}\,\log\,\frac{\mathbf{X}_{ij}}{\mathbf{G}_{ij}\mathbf{H}_{ij}} - \mathbf{X}_{ij} + \mathbf{G}_{ij}\mathbf{H}_{ij}\right) \tag{3.3}$$

resulting in the update rules for the matrices $\mathbf{G}$ and $\mathbf{H}$ as

$$\mathbf{G}_{ij} \leftarrow \mathbf{G}_{ij}\frac{\sum_k \mathbf{H}_{jk}\mathbf{X}_{ik}/(\mathbf{GH})_{ik}}{\sum_k \mathbf{H}_{jk}} \qquad \mathbf{H}_{ij} \leftarrow \mathbf{H}_{ij}\frac{\sum_k \mathbf{G}_{ki}\mathbf{X}_{kj}/(\mathbf{GH})_{kj}}{\sum_k \mathbf{G}_{ki}} \tag{3.4}$$

3.3.2. Semi Non-negative Matrix Factorization

Ding *et al*. [29] proposed a variation of the NMF called Semi-NMF which relaxes the non-negativity constraints on $\mathbf{G}$ and the data matrix $\mathbf{X}$. The rationale behind its creation was based on the case of clustering with $\mathbf{G}$ representing the cluster centers and $\mathbf{H}$ denoting the cluster indicator. The unknown matrices $\mathbf{G}$ and $\mathbf{H}$ are estimated by minimizing the reconstruction error,

$$\underset{\mathbf{H}\succeq 0}{\arg\min}\|\mathbf{X} - \mathbf{GH}\|_F^2 = \underset{\mathbf{H}\succeq 0}{\arg\min}\sum_{ij}\left(\mathbf{X}_{ij} - \mathbf{G}_{ij}\mathbf{H}_{ij}\right)^2 \tag{3.5}$$

w.r.t. $\mathbf{G}$ and $\mathbf{H}$. The above minimization problems are iteratively solved with respect to the matrices $\mathbf{G}$ and $\mathbf{H}$ using a set of update rules [29]:
Step 1: Update $\mathbf{G}$ by keeping $\mathbf{H}$ fixed

$$\mathbf{G} = \mathbf{XH}^T(\mathbf{HH}^T)^{-1} \tag{3.6}$$

Step 2: Update $\mathbf{H}$ by keeping $\mathbf{G}$ fixed,

$$\mathbf{H} = \mathbf{H} \odot \sqrt{\frac{[\mathbf{G}^T\mathbf{X}]^+ + [\mathbf{G}^T\mathbf{G}\mathbf{H}]^-}{[\mathbf{G}^T\mathbf{G}]^+\mathbf{H} + [\mathbf{G}^T\mathbf{X}]^-}} \tag{3.7}$$

where $\odot$ represents the element-wise multiplication, $\mathbf{M}^+$ and $\mathbf{M}^-$ correspond to a positive and a negative part of the matrix $\mathbf{M}$, respectively, given by

$$\mathbf{M}^+{}_{ik} = \frac{1}{2}(|\mathbf{M}_{ik}| + \mathbf{M}_{ik}), \qquad \mathbf{M}^-{}_{ik} = \frac{1}{2}(|\mathbf{M}_{ik}| - \mathbf{M}_{ik}).$$

3.3.3. Max-Margin Semi-NMF (MNMF)

As stated above, the Semi-NMF algorithm [29] minimizes the cost function defined in Eq. 3.5 imposing at the same time non-negativity constraints only on the coefficeint matrix $\mathbf{H}$. This relaxation of non-negativity constraints on the bases extends the range of applications of NMF as Semi-NMF can also be applied on the negative data. Several NMF variants incorporating discriminant constraints were proposed in [29, 50, 133]. The variations were obtained by introducing application specific discriminant constraints to the cost function. Inspired by this, we propose to incorporate discriminant constraints into the formulation of Semi-NMF. More specifically, we aim at creating a framework that allows us to find a set of basis vectors that maximizes the margin of an SVM classifier.

Let $\{\mathbf{x}_i, y_i\}_{i=1}^n$ denote a set of data vectors and their corresponding labels, where $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{-1, 1\}$. Our aim is to determine a base matrix $\mathbf{G}$ that can be used to extract features that are optimal under a max-margin classification criterion. This is accomplished by imposing constraints on the feature vectors derived from $\mathbf{G}$. In this work, similar to [14, 50, 133], the features that are extracted from a data example $\mathbf{x}$ are given by $\mathbf{G}^T\mathbf{x}$. That is, they are the projections of the data example $\mathbf{x}$ on the basis vectors stored in $\mathbf{G}$. Then, the optimization problem is given by

$$\begin{aligned} \underset{\mathbf{G},\mathbf{H},\mathbf{w},b,\xi_i}{\arg\min}\ & \lambda\|\mathbf{X} - \mathbf{G}\mathbf{H}\|_F^2 + \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.}\ & y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0, \quad 1 \leq i \leq n, \ \mathbf{H} \succeq 0, \end{aligned} \tag{3.8}$$

where $\boldsymbol{\xi} = \{\xi_1, \ldots, \xi_i, \ldots \xi_n\}$ is the slack variable vector, λ is a scalar that controls the relative importance for the NMF cost and C a scalar that controls the relative importance of the penalty imposed for the training examples that are either too close to the separating hyperplane or misclassified. The first term of the above optimization

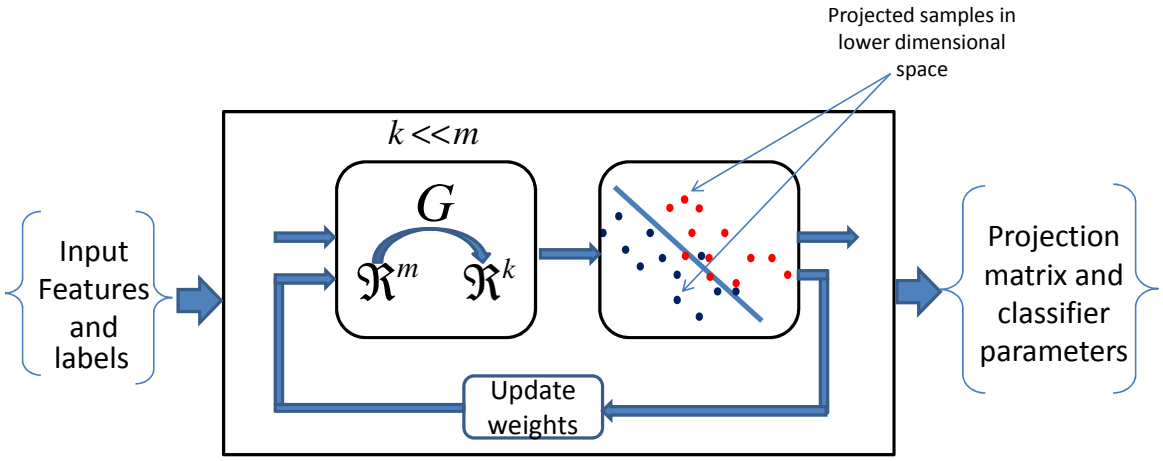


Figure 3.2.: Framework for MNMF: The input descriptors are projected onto the base matrix and a svm classifier is learned on the projected features. Subsequently the coefficient matrices are updated. This procedure is repeated until convergence. The objective is to find a base matrix (or a subspace) such that the projected features maximize the margin of the classifier in the subspace

problem ($\lambda\|\mathbf{X} - \mathbf{GH}\|_F^2$) is a classical NMF-type reconstruction error, while the second ($\frac{1}{2}\mathbf{w}^T\mathbf{w}$) and the third term ($C\sum_{i=1}^{n}\xi_i$) is an SVM-type cost. Notice that the slack variables ξ_i in third term control the misclassification errors. Notice also that the inequality constraints $y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i$ that involve the slack variables, depend on both the parameters $\mathbf{w}$ of the classifier and on the data projection matrix $\mathbf{G}^T$ that is used to extract the features $\mathbf{G}^T\mathbf{x}$. In this way, we jointly optimize with respect to both the Semi-NMF data projections and the maximum margin classifier.

Notice, that classical NMF-based algorithms use $\mathbf{G}^\dagger = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T$ as the projection matrix, that is the features that are extracted for a data example $\mathbf{x}$ are given by $\acute{\mathbf{x}} = \mathbf{G}^\dagger\mathbf{x}$. By contrast we follow [14, 50, 133] and use $\mathbf{G}^T$ as the projection matrix. Both NMF and our MNMF find a base matrix $\mathbf{G}$ and express an arbitrary $\mathbf{x}$ as a (non negative) linear combination (with coefficients $\mathbf{h}$) of the column vectors of $\mathbf{G}$. Using $\mathbf{G}^\dagger$ as the projection matrix results in features that are the (non-negative) coefficients $\mathbf{h}$ that minimize the MSE. The projection $\mathbf{G}^T\mathbf{x}$ that we propose, uses as features the projection of the vector $\mathbf{x}$ on the basis vectors. Both choices are equally valid. Ours is easier to work with and results to a formulation that is quadratic with respect to $\mathbf{G}$.

In order to optimize the cost function in Eq. 3.8, we follow an iterative optimization procedure. More precisely, at each iteration we solve for subsets of the unknown parameters $\mathbf{G}$, $\mathbf{H}$ and $\mathbf{w}, b, \xi_i$ by keeping the remaining parameters fixed. The optimization procedure is described below, and the steps followed in the proposed max-margin Semi-NMF framework are summarized in Algorithm 1.

Solve for G, $\boldsymbol{\xi}$ by keeping H, w and b fixed: Since $\mathbf{w}$ is fixed, the optimization problem in Eq. 3.8 is simplified as

$$\begin{aligned} \underset{\mathbf{G},\xi_i}{\arg\min} \quad & \lambda\|\mathbf{X}-\mathbf{GH}\|_F^2 + C\sum_{i=1}^{n}\xi_i \\ & \text{s.t.} \;\; y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i + b) \geqslant 1-\xi_i \\ & \xi_i \geqslant 0, \;\; 1 \leq i \leq n. \end{aligned} \tag{3.9}$$

The above formulation can be derived from Eq. 3.8 if the second term is omitted. It is a weighted combination of the reconstruction error ($\|\mathbf{X}-\mathbf{GH}\|_F^2$) and soft constraints or penalizations for the examples that do not maintain the appropriate distance (margin) from the separating hyperplane ($\sum_{i=1}^{n}\xi_i$). In this step, we solve for a projection matrix that projects the input examples to a lower dimensional feature space and the slack variables ξ_i's so as to jointly optimise for the projection and the classifier margin error. This results in a set of basis vectors $\mathbf{G}$ that simultaneously reduce the reconstruction error while ensuring a low misclassification error. Solving Eq. 3.9 is equivalent to solving

$$\begin{aligned} \underset{\mathbf{G},\xi_i}{\arg\min} \quad & \|\mathbf{X}-\mathbf{GH}\|_F^2 + \theta\sum_{i=1}^{n}\xi_i \\ & \text{s.t.} \;\; y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i + b) \geqslant 1-\xi_i \\ & \xi_i \geqslant 0, \;\; 1 \leq i \leq n. \end{aligned} \tag{3.10}$$

where $\theta = C/\lambda$. We should note that Eq. 3.10 is an optimisation problem of a function that is either quadratic or linear with respect to the unknowns, subject to linear inequality constraints. The Lagrangian of Eq. 3.10 is given by

$$\begin{aligned} L(\mathbf{G},\xi_i,\alpha_i,\beta_i) \;=\; & Tr\Big[(\mathbf{X}-\mathbf{GH})(\mathbf{X}-\mathbf{GH})^T\Big] + \\ & \theta\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i+b)-1+\xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i \\ & \alpha_i,\beta_i \geqslant 0, \quad 1 \leq i \leq n \end{aligned} \tag{3.11}$$

where α_i, β_i are the Lagrangian multipliers. Taking the derivative w.r.to the primal variables and equating to 0, we get

$$\mathbf{G} = \left(2\mathbf{X}\mathbf{H}^T + \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \mathbf{w}^T\right)(2\mathbf{H}\mathbf{H}^T)^{-1} \tag{3.12}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad 0 \leq \alpha_i \leq \theta, \quad 1 \leq i \leq n \tag{3.13}$$

where $\theta = C/\lambda$. Substituting the value of $\mathbf{G}$ in Eq. 3.11 and simplifying, we get the dual problem

$$\begin{aligned} \arg\max_{\boldsymbol{\alpha}} \;\; & \boldsymbol{\alpha}^T(\mathbf{T}_1 - \mathbf{T}_2)\boldsymbol{\alpha} + (\mathbf{t}_3 - \mathbf{t}_4 - \mathbf{t}_5 - \mathbf{t}_6 + \mathbf{t}_7)\boldsymbol{\alpha} \\ & \text{s.t.} \;\; 0 \leq \alpha_i \leq \theta \end{aligned} \tag{3.14}$$

where

$$\begin{aligned} & \boldsymbol{\alpha} \in \mathbb{R}^n, \;\; \mathbf{T}_1, \mathbf{T}_2 \in \mathbb{R}^{n \times n}, \;\; \mathbf{t}_3, \mathbf{t}_4, \mathbf{t}_5, \mathbf{t}_6, \mathbf{t}_7 \in \mathbb{R}^{1 \times n}, \\ & \mathbf{T}_{1ij} = \left[\sum_{z=1}^{n} y_i y_j \mathbf{h}_z^T \mathbf{B}\mathbf{M}_i^T \mathbf{M}_j \mathbf{B}\mathbf{h}_z\right]_{ij} \\ & \mathbf{T}_{2ij} = \left[y_i y_j \mathbf{w}^T \mathbf{B}\mathbf{M}_j^T \mathbf{x}_i\right]_{ij} \\ & \mathbf{t}_{3i} = \left[4\sum_{z=1}^{n} y_i \mathbf{h}_z^T \mathbf{B}\mathbf{H}\mathbf{X}^T \mathbf{M}_i \mathbf{B}\mathbf{h}_z\right]_{1i} \\ & \mathbf{t}_{4i} = \left[2\sum_{z=1}^{n} y_i \mathbf{h}_z^T \mathbf{B}\mathbf{w}\mathbf{x}_i^T \mathbf{x}_z\right]_{1i} \\ & \mathbf{t}_{5i} = \left[2y_i \mathbf{w}^T \mathbf{B}\mathbf{H}\mathbf{X}^T \mathbf{x}_i\right]_{1i}, \;\; \mathbf{t}_{6i} = b\left[y_i\right]_{1i} \\ & \mathbf{t}_7 = \left[111\cdots1\right]_{1\times n}, \;\; \mathbf{B} = (2\mathbf{H}\mathbf{H}^T)^{-1}, \;\; \mathbf{M}_i = \mathbf{x}_i \mathbf{w}^T, \end{aligned} \tag{3.15}$$

and $\mathbf{h}_z$ is the z^{th} column of the matrix $\mathbf{H}$.

The above problem is quadratic in $\boldsymbol{\alpha}$, thus can be solved by using conventional quadratic programming tools. The estimated $\boldsymbol{\alpha}$ is then used to compute $\mathbf{G}$ using Eq. 3.12. The constant term θ in Eq. 3.14 is used as a tuning parameter. Large values of λ (when compared to C), result in low values of θ something that leads to small α_i. This in turn causes the second term in Eq. 3.12 to disappear making the update rule of $\mathbf{G}$ to be the one used in semi-NMF, as given in Eq. 3.6. Hence for large values of λ, the

update rule for $\mathbf{G}$ tends to approach the update rule of semi-NMF, something that is also evident in Eq. 3.9.

Solve for $\mathbf{w}, b, \boldsymbol{\xi}$ by keeping G and H fixed: In the previous step, we computed the updated bases matrix $\mathbf{G}$. We now proceed in solving for the hyperplane that maximizes the margin of the classifier keeping the bases matrix $\mathbf{G}$ and weights matrix $\mathbf{H}$ fixed. The features are obtained by projecting the data points onto the updated basis matrix $\mathbf{G}$ as calculated in the previous step. Since $\mathbf{G}$ and $\mathbf{H}$ are fixed, the optimization problem in Eq. 3.8 is simplified to a form that strongly resembles that of a classical SVM:

$$\begin{aligned} &\arg\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t. } y_i(\mathbf{w}^T\mathbf{G}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i \\ &\xi_i \geqslant 0, \;\; 1 \leq i \leq n. \end{aligned} \tag{3.16}$$

The hyperplane parameters $\mathbf{w}$, and b and the slack variable vector $\boldsymbol{\xi}$ are obtained using an off-the-shelf SVM classifier.

Solve for H by keeping G, w, $\boldsymbol{\xi}$ and b fixed: Having already acquired the values for $\mathbf{G}$,$\mathbf{w}$, b and $\boldsymbol{\xi}$ from the previous steps, we proceed with solving for the weights matrix $\mathbf{H}$ by keeping all the other variables fixed. Since only the reconstruction error term of the optimization problem (Eq. 3.8) depends on $\mathbf{H}$, the objective function is simplified as

$$\begin{aligned} &\arg\min_{\mathbf{H}} \|\mathbf{X} - \mathbf{G}\mathbf{H}\|_F^2, \\ &\text{s.t. } \mathbf{H} \succeq 0. \end{aligned} \tag{3.17}$$

The i^{th} column of $\mathbf{H}$, $\mathbf{h}_i$ contributes only to the i^{th} data point $\mathbf{x}_i$ and hence the columns of $\mathbf{H}$ can be solved independent of each other. The above optimization problem can be solved using quadratic programming or using the update equation Eq. 3.7. Here, we adopt the update rule to solve for $\mathbf{h}_i$. In particular, the objective function in Eq. 3.17 can be rewritten as

$$\begin{aligned} &\arg\min_{\mathbf{h}_i} \;\; (\mathbf{x}_i - \mathbf{G}\mathbf{h}_i)^T(\mathbf{x}_i - \mathbf{G}\mathbf{h}_i), \\ &\text{s.t. } \mathbf{h}_i \succeq 0 \;\; \forall i. \end{aligned} \tag{3.18}$$

The Lagrangian of the above cost function is

$$L(\mathbf{h}_i) = (\mathbf{x}_i - \mathbf{G}\mathbf{h}_i)^T(\mathbf{x}_i - \mathbf{G}\mathbf{h}_i) - \boldsymbol{\gamma}^T\mathbf{h}_i, \quad \gamma > 0 \tag{3.19}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^k$ is a vector of positive Lagrangian multiplier.

quadratic programming: Differentiating the above equation w.r.t. $\mathbf{h}_i$ and equating to zero, we get

$$\mathbf{h}_i = (2\mathbf{G}^T\mathbf{G})^{-1}(2\mathbf{G}^T\mathbf{x}_i + \boldsymbol{\gamma}). \tag{3.20}$$

The dual formulation for Eq. 3.19 is given by

$$\arg\max_{\boldsymbol{\gamma}>0} \frac{1}{2}\boldsymbol{\gamma}^T\mathbf{B}\boldsymbol{\gamma} + 2\boldsymbol{\gamma}^T\mathbf{B}\mathbf{G}^T\mathbf{x}_i \tag{3.21}$$
$$\text{where } \mathbf{B} = (2\mathbf{G}^T\mathbf{G})^{-1}.$$

The above problem is quadratic in $\boldsymbol{\gamma}$, so a quadratic programming solver can be used to solve it. The weight vector $\mathbf{h}_i$ is obtained by substituting the computed value of $\boldsymbol{\gamma}$ in Eq. 3.20. This procedure is repeated for all columns of $\mathbf{H}$. Note that the Eq. 3.20 will not result in a non negative $\mathbf{h}_i$ for an arbitrary γ, but it is warranted that Eq. 3.20 gives a non negative $\mathbf{h}_i$ for the γ that results from Eq. 3.21.

Update Rules: Taking under consideration the KKT conditions [11], we get:

$$\nabla L(\mathbf{h}_i) = 0, \tag{3.22}$$
$$\gamma_j h_{ij} = 0 \quad \text{and} \tag{3.23}$$
$$\gamma_j \geq 0, \tag{3.24}$$

where h_{ij} is the j-th element of $\mathbf{h}_i$. Let $F(\mathbf{h}_i) = \|\mathbf{x}_i - \mathbf{G}\mathbf{h}_i\|^2$, From Eq. 3.22 we get:

$$\nabla L(\mathbf{h}_i) = 0 \Rightarrow \nabla F(\mathbf{h}_i) - \gamma = 0 \Rightarrow [\nabla F(\mathbf{h}_i)]_j = \gamma_j \tag{3.25}$$

Since $\gamma_j h_{ij} = 0 \Rightarrow [\nabla F(\mathbf{h}_i)]_j h_{ij} = 0$. Therefore

$$\nabla F(\mathbf{h}_i) = -2\mathbf{G}^T\mathbf{x}_i + 2\mathbf{G}^T\mathbf{G}\mathbf{h}_i \tag{3.26}$$
$$= -\left([2\mathbf{G}^T\mathbf{x}_i]^+ - [2\mathbf{G}^T\mathbf{x}_i]^-\right) + \left([2\mathbf{G}^T\mathbf{G}\mathbf{h}_i]^+ - [2\mathbf{G}^T\mathbf{G}\mathbf{h}_i]^-\right) \tag{3.27}$$
$$= -\left([2\mathbf{G}^T\mathbf{x}_i]^+ + [2\mathbf{G}^T\mathbf{G}\mathbf{h}_i]^-\right) + \left([2\mathbf{G}^T\mathbf{G}\mathbf{h}_i]^+ + [2\mathbf{G}^T\mathbf{x}_i]^-\right). \tag{3.28}$$

Hence

$$\mathbf{h}_i = \mathbf{h}_i \odot \frac{\left([\mathbf{G}^T\mathbf{x}_i]^+ + [\mathbf{G}^T\mathbf{G}\mathbf{h}_i]^-\right)}{\left([\mathbf{G}^T\mathbf{G}\mathbf{h}_i]^+ + [\mathbf{G}^T\mathbf{x}_i]^-\right)}. \tag{3.29}$$

This procedure is repeated for all columns of $\mathbf{H}$.

Algorithm 1: Algorithm for MNMF

```
input  : X, Ginit, Hinit, MAXITER, λ, C
output: G, H, w, b
begin
    G = Ginit;
    H = Hinit;
    repeat
        S1 : Solve for α in Eq. 3.14
        S2 : Compute G using Eq. 3.12
        S3 : Find the classifier parameters, w, b, ξ_i for the updated G
        S4 : foreach column h_i of H do
             Compute h_i using Eq. 3.29
        end
    until iter ≤ MAXITER   or   convergence;
end
```

During testing, the input test vector $\mathbf{x}_{test}$ is projected onto the basis matrix to obtain the feature vector, $\mathbf{f}_{test} = \mathbf{G}^T\mathbf{x}_{test}$. This feature vector is then given as input to the max-margin classifier which predicts the class $\hat{\mathbf{y}}_{test} = sign(\mathbf{w}^T\mathbf{f}_{test} + b)$ where $\mathbf{w}, b, \mathbf{G}$ are computed during training.

3.3.4. Experiments on a Synthetic Toy Dataset

In order to provide an insight to the way the proposed MNMF algorithm works, we first conduct experiments on a toy dataset consisting of two classes each of which contains 100 points that are sampled from 50-dimensional Gaussian distributions. For visualization purposes, we restrict the number of bases taken under consideration to be equal to two ($k = 2$). The bases matrix $\mathbf{G}$ and the weights matrix $\mathbf{H}$ are computed using the Semi-NMF algorithm and the input data points are projected onto the lower dimensional subspace using the acquired $\mathbf{G}$. In Fig. 3.3(a) we show the projections of the points after applying a common dimensionality reduction technique, namely Principal Component Analysis (PCA) [103]. Fig. 3.3(b) depicts the projections of the input datapoints using

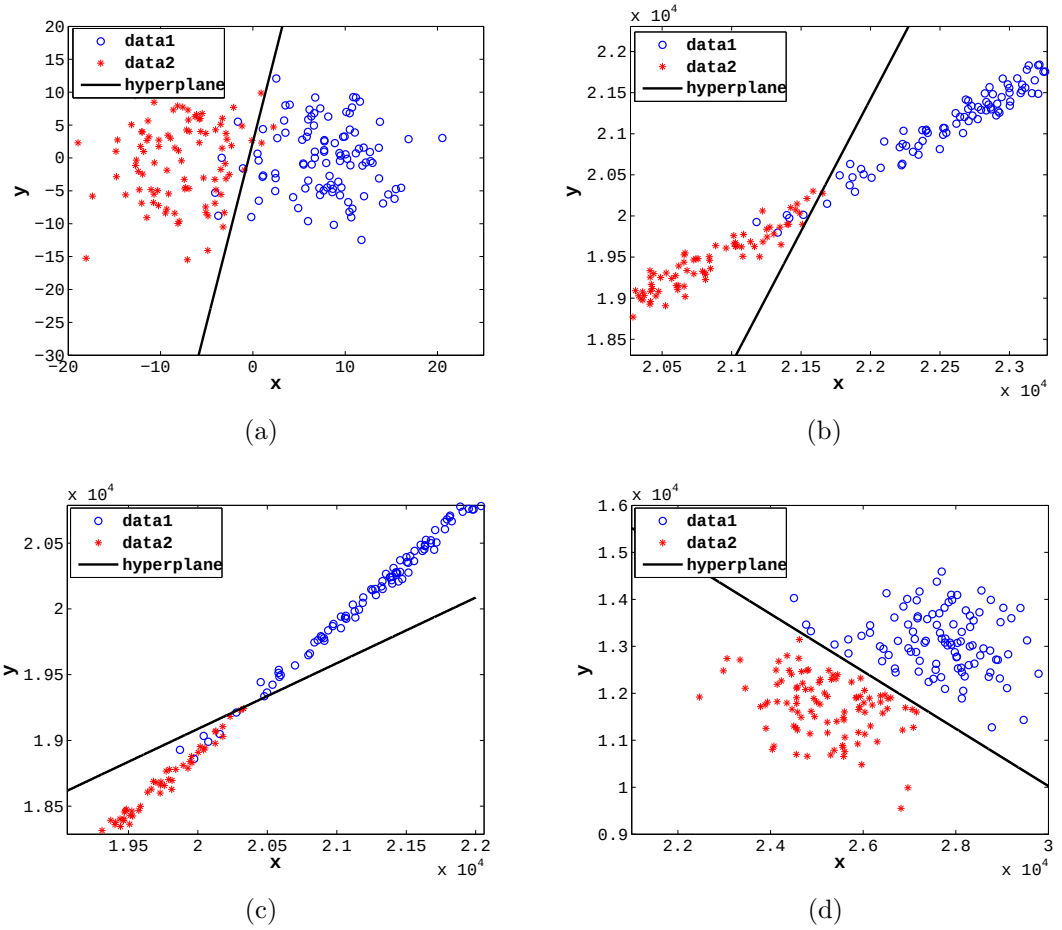


Figure 3.3.: The projections and the SVM separating hyperplane using: 3.3(a) PCA; 3.3(b) Semi-NMF bases; 3.3(c) Max-margin NMF bases (1^{st} iteration); 3.3(d) MNMF bases (6^{th} iteration)

the bases extracted using Semi-NMF. Fig. 3.3(c) and Fig. 3.3(d) show the projections of the proposed MNMF algorithm after the first and the sixth iterations, respectively. It is clear that the projections acquired from the proposed MNMF algorithm make the classes more separable.

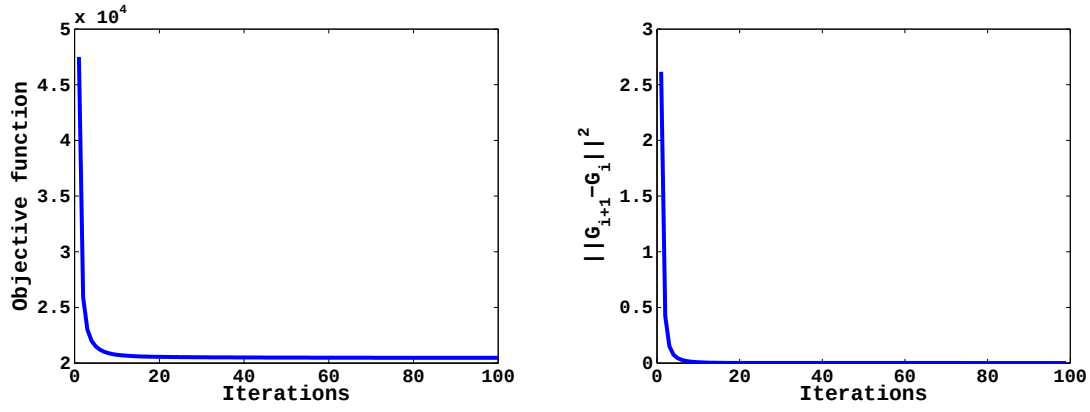
In order to examine the discriminative power of the features extracted by each of the above mentioned methods we trained an SVM classifier on the acquired projections and report the obtained classification accuracies. When PCA, Semi-NMF (at convergence, *i.e.*, after 2000 iterations) and the proposed MNMF algorithm (at convergence, *i.e.*, after only 6 iterations) were applied, the accuracies obtained were equal to 96.5%, 97% and 100%, respectively. This verifies the fact that the proposed algorithm updates the

bases in such a way that the margin of the classifier in the projected space is maximized, thus achieving lower misclassification error.

3.3.5. Convergence Issues

In this section, we discuss the convergence of the proposed MNMF framework. The objective function proposed in Eq. 3.8 is a weighted combination of the NMF cost and the classifier (SVM) cost. We optimize this objective function using a block coordinate descent where at each step we solve a set of convex sub-problems each of which is guaranteed to converge. Therefore, the whole procedure converges to a (local) minimum. A proof of the convergence is provided in B.

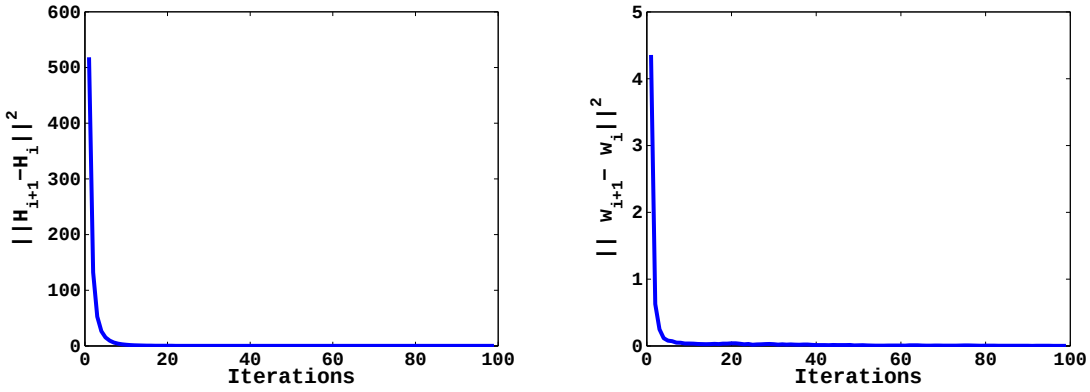
We verified, experimentally, that the proposed optimization procedure does reduce at each step the objective function in Eq. 3.8 and that the Frobenius norm of the differences (between subsequent iterations) in $\mathbf{G}, \mathbf{H}$ and hyperplane parameter $\mathbf{w}$ converge to zero. In order to demonstrate this, we use a subset of the KTH dataset consisting of the action classes Run and Walk. The details of the experiment are given in Section 3.5.4. The plot in Fig. 3.4(a) shows that the objective function decreases and converges after few iterations. The convergence of the parameters $\mathbf{G}$, $\mathbf{H}$ and $\mathbf{w}$ is also evident from Fig 3.4(b), 3.5(a), and 3.5(b), respectively.



(a) (b)

Figure 3.4.: (a) Objective function in Eq. 3.8 vs Iterations, (b) $\|\mathbf{G}_{i+1} - \mathbf{G}_i\|_F^2$ vs Iterations

Note that in the proposed framework, we solve for $\boldsymbol{\xi}$ in two places, *i.e.*, in Eq. 3.9 and Eq. 3.16. Alternatively we could for example solve for $\boldsymbol{\xi}$ only in Eq. 3.9 and substitute the obtained values in Eq. 3.16. As optimization strategies, both are valid block-coordinate



(a) (b)

Figure 3.5.: (a) $\|\mathbf{H}_{i+1} - \mathbf{H}_i\|_F^2$ vs Iterations, (b) $\|\mathbf{w}_{i+1} - \mathbf{w}_i\|^2$ vs Iterations

descent optimizations, which at each step reduce the objective function and therefore lead to local minima. However, we notice that a change in either the projection matrix $\mathbf{G}$ or the hyperplane parameters can violate the inequality $\mathbf{w}^T\mathbf{G}^T x_i + b \geq 1 - \xi_i$. Fixing the slack variables when solving for, say, $\mathbf{w}$ in Eq. 3.16 would make the inequality constraints **hard**, therefore lead to worse solutions of $\mathbf{w}$ (in terms of the objective function). By contrast, if we solve for the penalty term $\boldsymbol{\xi}$ in both Eq. 3.9 and Eq. 3.16, the constraints are **soft** in both cases. Fig. 3.6 shows the plot of $\sum_i \|\xi_i - \xi_i'\|^2$ where ξ_i is obtained by solving for ξ_i in Eq. 3.9 and ξ_i' is obtained by solving for ξ_i in Eq. 3.16. From Fig. 3.6, we see that the sum of the differences converges to zero, that is, ξ_i and ξ_i' converge to the same values.

3.4. Max-Margin Kernel NMF (KMNMF)

In the previous Section, we presented our proposed framework assuming that a linear Semi-NMF was used. However, linear NMF algorithms cannot properly capture the non-linear structure that the data may follow. To tackle this problem the authors in [134] proposed the Kernel extension of NMF (KNMF) and showed that it significantly improves the performance over NMF in classification applications. In the following sections, we extend the max-margin framework described in the previous section to include KNMF.

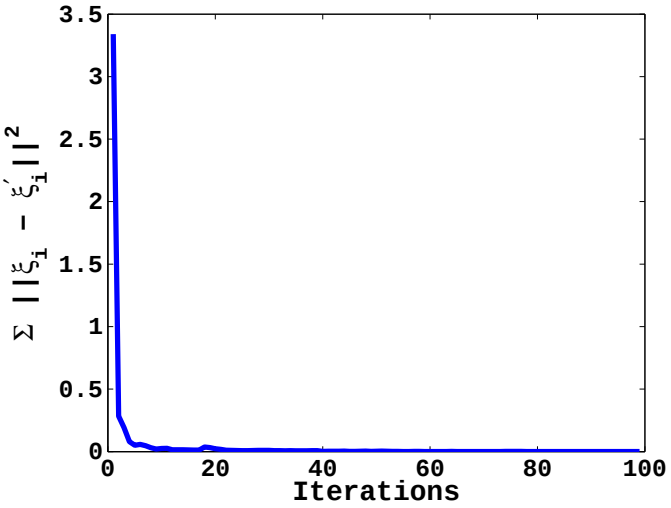


Figure 3.6.: Plot of $\sum_i \|\xi_i - \xi_i'\|^2$ vs iterations where ξ_i is obtained by solving Eq. 3.9 and ξ_i' is obtained by solving Eq. 3.16

3.4.1. Overview of KNMF

Let Φ denote a non-linear transformation that maps data $\mathbf{x} \in \mathbb{R}^m$ in the input space to a higher dimensional feature space, *i.e.*, $\Phi : \mathbf{x} \in \mathbb{R}^m \rightarrow \Phi(\mathbf{x}) \in \mathbb{R}^f$, typically $f \gg m$. Let $\Phi(\mathbf{X}) = [\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \cdots \Phi(\mathbf{x}_n)]$ denote the data matrix where each example $\Phi(\mathbf{x}_i) \in \mathbb{R}^f$. KNMF decomposes the data matrix as

$$\Phi(\mathbf{X}) \approx \mathbf{G}_\Phi \mathbf{H} \tag{3.30}$$

where the base matrix $\mathbf{G}_\Phi \in \mathbb{R}^{f \times k}$ contains the basis vectors in the feature space and the coefficients matrix $\mathbf{H} \in \mathbb{R}^{k \times n}$ indicates the contribution of each basis vector in the reconstruction of the example. In practice, the computation of $\Phi(\mathbf{X})$ and $\mathbf{G}_\Phi$ are impractical and thus the kernel trick [12] is employed to efficiently compute the similarities in the feature space,

$$\mathbf{K} \approx \mathbf{Y}\mathbf{H},$$

where $\mathbf{K} = \Phi^T(\mathbf{X})\Phi(\mathbf{X})$ is the kernel matrix and $\mathbf{Y} = \Phi^T(\mathbf{X})\mathbf{G}_\Phi$. The coefficient vector $\mathbf{h}_{test}$ for a test example $\mathbf{x}_{test}$ is given by,

$$\mathbf{h}_{test} = \mathbf{Y}^\dagger \mathbf{K}_{test},$$

where $\mathbf{K}_{test} = \Phi^T(\mathbf{X})\Phi(\mathbf{x}_{test})$ and $\dagger$ denotes the pseudo-inverse.

3.4.2. Max-Margin Kernel NMF (KMNMF)

In this section, we formulate our proposed framework by imposing max-margin constraints within KNMF. We aim at finding a set of basis vectors in the feature space, derived using KNMF, that maximizes the margin of an SVM classifier in the reconstructed feature space. This is illustrated in Fig. 3.7

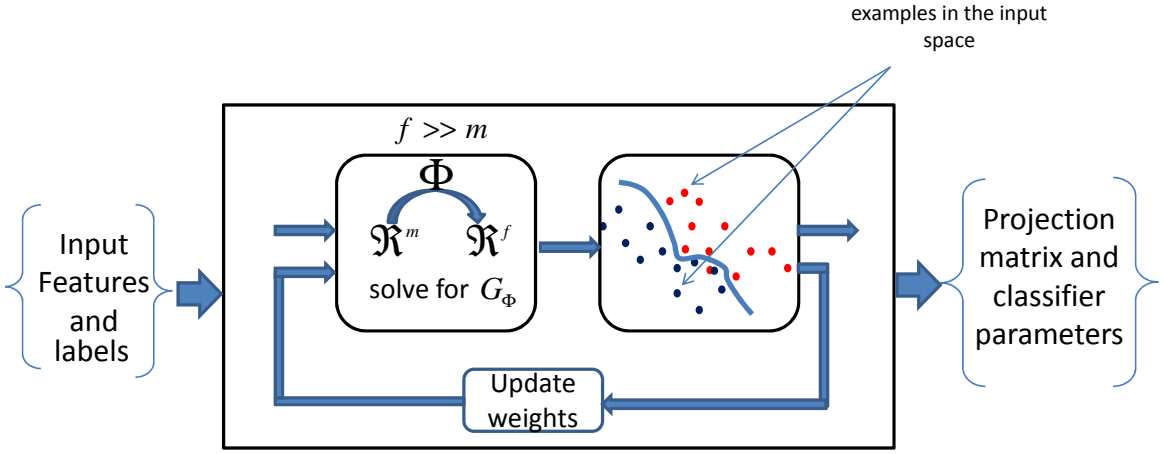


Figure 3.7.: Framework for KMNMF: The input descriptors are reconstructed in the high dimensional feature space where a linear svm classifier separates the samples. The objective is to find a base matrix such that the reconstructed samples in the high dimensional feature space maximize the margin of the classifier

Cost Function for KMNMF

Let $\{\Phi(\mathbf{x}_i), y_i\}_{i=1}^n$ denote a set of data vectors in the feature space and their corresponding labels, where $\Phi(\mathbf{x}_i) \in \mathbb{R}^f$, $y_i \in \{-1, 1\}$. The objective is to determine a set of basis vectors acquired using KNMF that can be used to reconstruct the data in the feature space in such a way that they are optimal under a max-margin classification criterion. This is accomplished by imposing constraints on the reconstructed data computed using the bases matrix $\mathbf{G}_\Phi$. Let the reconstructed vector for a data example $\Phi(\mathbf{x}_j)$ be given as $\Phi(\tilde{\mathbf{x}}_j) \approx \mathbf{G}_\Phi \mathbf{h}_j$ where $\mathbf{h}_j$ is the coefficient vector.

The optimization problem for the proposed criterion is given by

$$\begin{aligned} \underset{\mathbf{G}_{\Phi},\mathbf{H},\mathbf{w}_{\Phi},b,\xi_i}{\arg\min}\ & \lambda\|\Phi(\mathbf{X}) - \mathbf{G}_{\Phi}\mathbf{H}\|_F^2 + \frac{1}{2}{\mathbf{w}_{\Phi}}^T\mathbf{w}_{\Phi} + C\sum_{i=1}^{n}\xi_i \\ \text{s.t. }\ & y_i({\mathbf{w}_{\Phi}}^T\mathbf{G}_{\Phi}\mathbf{h}_i + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0,\ \ 1 \leq i \leq n,\ \mathbf{H} \succeq 0 \end{aligned} \tag{3.31}$$

where $\Phi(\mathbf{X}) = \{\Phi(\mathbf{x}_i)\}_{i=1}^n$ and λ is the weight factor for the KNMF cost. The first term ($\lambda\|\Phi(\mathbf{X}) - \mathbf{G}_{\Phi}\mathbf{H}\|_F^2$) corresponds to the KNMF reconstruction error, the second term ($\frac{1}{2}{\mathbf{w}_{\Phi}}^T\mathbf{w}_{\Phi}$) corresponds to the maximum margin classifier in the reconstructed space and the third term ($C\sum_{i=1}^{n}\xi_i$) is the common term shared by the two previously mentioned terms, used to penalize the misclassified examples with respect to the input projections acquired from KNMF. The above formulation aims at maximizing the margin of the support vectors while at the same time minimizing the reconstruction and misclassification errors. The classifier is trained on the reconstructed data points $\mathbf{G}_{\Phi}\mathbf{h}$, obtaining in this way the hyperplane parameter $\mathbf{w}_{\Phi} \in \mathbb{R}^f$. We follow the same procedure described in the previous section and iteratively solve for one of the terms $\mathbf{G}_{\Phi}$, $\mathbf{H}$ and $\mathbf{w}_{\Phi}$, b, $\boldsymbol{\xi}$ while keeping the remaining parameters fixed.

We should note here that since the columns of the bases matrix $\mathbf{G}_{\Phi}$, the data matrix $\Phi(\mathbf{X})$ and the SVM hyperplane parameter $\mathbf{w}_{\Phi}$ lie in the feature space, their explicit computation is not necessary. Instead we solve explicitly for the parameters of the dual formulations of their corresponding constrained optimization problems and use them in order to calculate quantities in the form of dot products in the feature space. More specifically, when we calculate $\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}$ and $\mathbf{G}_{\Phi}^T\Phi(\mathbf{X})$, while when we solve for the max-margin hyperplane $\mathbf{w}_{\Phi}$ we calculate $\mathbf{w}_{\Phi}^T\mathbf{w}_{\Phi}$ and $\mathbf{w}_{\Phi}^T\Phi(\mathbf{X})$. For the data kernel matrix $\Phi(\mathbf{X})^T\Phi(\mathbf{X})$ we use the Gaussian kernel [12],

$$k(\mathbf{x},\mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right). \tag{3.32}$$

The steps followed in the proposed max-margin KNMF framework are summarized in Algorithm 2.

Solve for $\mathbf{G}_{\Phi}$ and $\boldsymbol{\xi}$ by keeping $\mathbf{H}, \mathbf{w}_{\Phi}$ and b fixed: Since $\mathbf{w}_{\Phi}$ remains fixed, the optimization problem in Eq. 3.31 is simplified as

$$\begin{aligned} \underset{\mathbf{G}_{\Phi},\xi_i}{\arg\min} \quad & \lambda\|\Phi(\mathbf{X}) - \mathbf{G}_{\Phi}\mathbf{H}\|_F^2 + C\sum_{i=1}^{n}\xi_i \\ & \text{s.t.} \;\; y_i({\mathbf{w}_{\Phi}}^T\mathbf{G}_{\Phi}\mathbf{h}_i + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0, \;\; 1 \leq i \leq n. \end{aligned} \tag{3.33}$$

The above formulation is derived from Eq. 3.31 if the second term is omitted. It is a weighted combination of the reconstruction error caused by the KNMF (1^{st} term) and the soft constraints/penalizations for the examples that do not maintain the appropriate distance (margin) from the separating hyperplane (3^{rd} term). Our aim is therefore to find a set of bases $\mathbf{G}_{\Phi}$ that simultaneously reduce the reconstruction and misclassification errors. The cost function in Eq. 3.33 can also be written as

$$\begin{aligned} \underset{\mathbf{G}_{\Phi},\xi_i}{\arg\min} \quad & \|\Phi(\mathbf{X}) - \mathbf{G}_{\Phi}\mathbf{H}\|_F^2 + \theta\sum_{i=1}^{n}\xi_i \\ & \text{s.t.} \;\; y_i({\mathbf{w}_{\Phi}}^T\mathbf{G}_{\Phi}\mathbf{h}_i + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0, \;\; 1 \leq i \leq n. \end{aligned} \tag{3.34}$$

where $\lambda = C/\theta$. We should note that the cost function in Eq. 3.34 is either a quadratic or linear function that imposes linear inequality constraints on the set of unknowns. We proceed with solving it using its dual formulation. The Lagrangian of Eq. 3.34 is given by

$$\begin{aligned} & L(\mathbf{G}_{\Phi}, \xi_i, \alpha_i, \beta_i) = Tr\Big(\big(\Phi(\mathbf{X}) - \mathbf{G}_{\Phi}\mathbf{H}\big)\big(\Phi(\mathbf{X}) - \mathbf{G}_{\Phi}\mathbf{H}\big)^T\Big) + \\ & \theta\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i({\mathbf{w}_{\Phi}}^T\mathbf{G}_{\Phi}\mathbf{h}_i) + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i \\ & \alpha_i, \beta_i \geq 0, \quad 1 \leq i \leq n \end{aligned} \tag{3.35}$$

where α_i, β_i are the Lagrangian multipliers. Taking the derivative w.r. to the primal variables and equating to 0, we have

$$\mathbf{G}_{\Phi} = \left(2\Phi(\mathbf{X})\mathbf{H}^T + \sum_{i=1}^{n}\alpha_i y_i \mathbf{w}_{\Phi}\mathbf{h}_i^T\right)(2\mathbf{H}\mathbf{H}^T)^{-1} \tag{3.36}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad 0 \leq \alpha_i \leq \theta, \quad 1 \leq i \leq n \tag{3.37}$$

Substituting the value of $\mathbf{G}_\Phi$ in Eq. 3.35 and simplifying, we get the dual problem

$$\arg\max_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha}^T(\mathbf{T}_1 - \mathbf{T}_2)\boldsymbol{\alpha} + (\mathbf{t}_3 - \mathbf{t}_4 - \mathbf{t}_5 - \mathbf{t}_6 + \mathbf{t}_7)\boldsymbol{\alpha} \quad \text{s.t. } 0 \leq \alpha_i \leq \theta \tag{3.38}$$

where

$$\begin{aligned}
&\boldsymbol{\alpha} \in \mathbb{R}^n, \ \ \mathbf{T}_1, \mathbf{T}_2 \in \mathbb{R}^{n\times n} \ , \ \mathbf{t}_3, \mathbf{t}_4, \mathbf{t}_5, \mathbf{t}_6, \mathbf{t}_7 \in \mathbb{R}^{1\times n}, \\
&\mathbf{T}_{1ij} = \left[\sum_{z=1}^{n} y_i y_j \mathbf{h}_z^T \mathbf{B}\mathbf{h}_i \mathbf{w}_\Phi^T \mathbf{w}_\Phi \mathbf{h}_j \mathbf{B}\mathbf{h}_z\right]_{ij} \\
&\mathbf{T}_{2ij} = \left[y_i y_j \mathbf{w}_\Phi^T \mathbf{w}_\Phi \mathbf{h}_i^T \mathbf{B}\mathbf{h}_j\right]_{ij} \\
&\mathbf{t}_{3i} = \left[4\sum_{z=1}^{n} y_i \mathbf{h}_z^T \mathbf{B}\mathbf{H}\Phi(\mathbf{X})^T \mathbf{w}_\Phi \mathbf{h}_i \mathbf{B}\mathbf{h}_z\right]_{1i} \\
&\mathbf{t}_{4i} = \left[2\sum_{z=1}^{n} y_i \mathbf{h}_z^T \mathbf{B}\mathbf{h}_i \mathbf{w}_\Phi^T \Phi(\mathbf{X})\right]_{1i} \\
&\mathbf{t}_{5i} = \left[2y_i \mathbf{w}^T \Phi(\mathbf{X})\mathbf{H}^T \mathbf{B}\mathbf{h}_i\right]_{1i}, \ \ \mathbf{t}_{6i} = b\left[y_i\right]_{1i} \\
&\mathbf{t}_7 = \left[111\cdots 1\right]_{1\times n}, \ \ \mathbf{B} = (2\mathbf{H}\mathbf{H}^T)^{-1}
\end{aligned} \tag{3.39}$$

and $\mathbf{h}_z$ is the z^{th} column of the matrix $\mathbf{H}$. The above problem is quadratic in $\boldsymbol{\alpha}$, thus enabling us to use conventional quadratic programming tools to solve it. Once $\boldsymbol{\alpha}$ is estimated we can compute

$$\begin{aligned}
\mathbf{G}_\Phi^T \mathbf{G}_\Phi = B^T \Bigg(&4\mathbf{H}\Phi(\mathbf{X})^T \Phi(\mathbf{X})\mathbf{H}^T + 4\sum_{i=1}^{n} \boldsymbol{\alpha}_i y_i \mathbf{H}\Phi(\mathbf{X})^T \\
&\mathbf{w}_\Phi \mathbf{h}_i^T + \mathbf{w}_\Phi^T \mathbf{w}_\Phi \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \mathbf{h}_i \mathbf{h}_j^T \Bigg) B
\end{aligned} \tag{3.40}$$

and

$$\mathbf{G}_\Phi^T \Phi(\mathbf{X}) = B\left(\mathbf{H}\Phi(\mathbf{X})^T \Phi(\mathbf{X}) + \sum_{i=1}^{n} \boldsymbol{\alpha}_i y_i \mathbf{h}_i w_\Phi^T \Phi(\mathbf{X})\right) \tag{3.41}$$

that are used in the subsequent optimization problems (e.g. Eq. 3.44).

The constant term θ in Eq. 3.38 is used as a tuning parameter. Large values of λ (when compared to C), result in low values of θ which cause α_i to decrease and the

second term of Eq. 3.36 to disappear. Hence for large values of λ, the KMNMF cost function resembles that of KNMF.

Solve for $\mathbf{w}_{\Phi}, b, \boldsymbol{\xi}$ by keeping $\mathbf{G}_{\Phi}$ and $\mathbf{H}$ fixed: Having computed the updated bases matrix $\mathbf{G}_{\Phi}$ from Eq. 3.36 and keeping the weights matrix $\mathbf{H}$ fixed, we proceed with calculating the maximum margin of the classifier. The features are obtained by reconstructing the data points in the feature space using the updated bases matrix. The optimization problem in Eq. 3.31 in that case strongly resembles that of a classical SVM:

$$\begin{aligned} &\arg\min_{\mathbf{w}_{\Phi}, b, \xi_i} \frac{1}{2}\mathbf{w}_{\Phi}^T\mathbf{w}_{\Phi} + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t. } y_i(\mathbf{w}_{\Phi}^T\mathbf{G}_{\Phi}\mathbf{h}_i + b) \geqslant 1 - \xi_i \\ &\xi_i \geqslant 0, \;\; 1 \leq i \leq n. \end{aligned} \tag{3.42}$$

The above optimization problem intends to maximize the margin of the classifier in the feature space while reducing the misclassification error appearing when using the projections acquired from KNMF. The hyperplane parameters $\mathbf{w}_{\Phi}$, and b are obtained using a off-the-shelf SVM classifier. The later takes as input the kernel matrix in the feature space, that is $\mathbf{H}^T\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}\mathbf{H}$. This can be calculated using $\mathbf{H}$ and the $\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}$ that is explicitly computed in Eq. 3.40. After obtaining the support vectors and the solution of the dual formulation of the problem, we can explicitly compute $\mathbf{w}_{\Phi}^T\mathbf{w}_{\Phi}$ and $\mathbf{w}_{\Phi}^T\Phi(\mathbf{X})$.

Solve for $\mathbf{H}$ by keeping $\mathbf{G}_{\Phi}$, $\mathbf{w}_{\Phi}, b$, and $\boldsymbol{\xi}$ fixed: We proceed with solving for the weights matrix $\mathbf{H}$ by keeping all the remaining variables fixed ($\mathbf{G}_{\Phi}$, $\mathbf{w}_{\Phi}, b$, and $\boldsymbol{\xi}$). Since only the reconstruction error term of the optimization problem (Eq. 3.31) depends on $\mathbf{H}$, the objective function in Eq. 3.31 is simplified as

$$\begin{aligned} &\arg\min_{\mathbf{H}} \|\Phi(\mathbf{X}) - \mathbf{G}_{\Phi}\mathbf{H}\|_F^2, \\ &\text{s.t. } \mathbf{H} \succeq 0. \end{aligned} \tag{3.43}$$

The i^{th} column of $\mathbf{H}$, $\mathbf{h}_i$, contributes only to the i^{th} data point $\Phi(\mathbf{x}_i)$ and hence the columns of $\mathbf{H}$ can be solved independent of each other. We adopt the update rule similar to the one used in MNMF to solve for $\mathbf{h}_i$:

$$\mathbf{h}_i = \mathbf{h}_i \odot \frac{\left([\mathbf{G}_{\Phi}^T\Phi(\mathbf{x}_i)]^+ + [\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}\mathbf{h}_i]^-\right)}{\left([\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}\mathbf{h}_i]^+ + [\mathbf{G}_{\Phi}^T\Phi(\mathbf{x}_i)]^-\right)}. \tag{3.44}$$

This procedure is repeated for all columns of $\mathbf{H}$.

Algorithm 2: Algorithm for KMNMF

input : $\mathbf{X}$, $\mathbf{H}init$, $MAXITER$, λ, C,σ
output: $\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}$, $\mathbf{H}$, $\mathbf{w}_{\Phi}$, b
begin
  $\mathbf{H} = \mathbf{H}init$;
  repeat
    $S1$: Solve for $\boldsymbol{\alpha}$ in Eq. 3.38
    $S2$: Compute $\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}$ using the values of $\boldsymbol{\alpha}$ as in Eq. 3.40
    $S3$: Compute the kernel matrix $\mathbf{H}^T\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}\mathbf{H}$
    $S4$: Use the computed kernel matrix to find the classifier parameters, $\mathbf{w}_{\Phi}$,b.
    $S5$: **foreach** *column* $\mathbf{h}_i$ *of* $\mathbf{H}$ **do**
      Compute $\mathbf{h}_i$ using Eq. 3.44
    end
  until $iter \leq MAXITER$ ***or*** *convergence*;
end

During testing, the coefficeint vector $\widehat{\mathbf{h}}_t$ for the test data $\mathbf{x}_t$ is computed as

$$\widehat{\mathbf{h}}_t = \arg\min_{\mathbf{h}_t} \|\Phi(\mathbf{x}_t) - \mathbf{G}_{\Phi}\mathbf{h}_t\|^2 \qquad s.t \quad \mathbf{h}_t \geqslant 0 \tag{3.45}$$

The above equation can be solved using quadratic programming. Eq. 3.45 requires the computation of $\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}$ which is computed as in Eq. 3.40 and also $\mathbf{G}_{\Phi}^T\Phi(\mathbf{x}_t)$ which can be computed by substituting $\mathbf{X}$ with $\mathbf{x}_t$ in Eq. 3.41. The kernel matrix between the training and test samples is computed as $\mathbf{H}^T\mathbf{G}_{\Phi}^T\mathbf{G}_{\Phi}\widehat{\mathbf{h}}_t$ and is used as input to the SVM classifier that classifies the given test sample.

3.5. Experimental Results

In the following sections, we demonstrate the performance of the proposed framework using real, publicly available datasets. More specifically, we use the INRIA-pedestrian dataset [26], the BU-3DFE [129] facial expression dataset, the Mediamill [106] dataset and the KTH actions dataset [101]. To allow comparisons with previously reported methods, we report results obtained by SVM classifiers trained with the features extracted using the Semi-NMF [29] and the KNMF [134] algorithms. We also report results with the DNMF algorithm [133] followed by a K Nearest Neighbors (KNN) classifier. We show that the classification performance of the proposed algorithms that jointly learn

the classifier parameters and the matrix factorization is consistently higher, especially when only few dimensions are retained. In our experiments, the parameter σ was set equal to the standard deviation of the data, that is $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \overline{\mathbf{x}}\|^2$ [134]. We set the parameter values $C = 100$ and $\lambda = 100$ for MNMF and $C = 100$ and $\lambda = 10^5$ for KMNMF.

3.5.1. INRIA Dataset

First, we tested our algorithm on the INRIA pedestrian dataset [26] described in Section 2.3. In order to handle possible illumination changes we used as features Histogram of Oriented Gradients (HOGs) [26]. The HOGs were extracted by creating a non-overlapping spatial grid size of 8×8, using 9 orientation bins per histogram. For each histogram four different normalizations were computed using adjacent histograms. This procedure resulted in a vector of length equal to 36 per region. For color images, the gradient was separately computed for each channel and the one with maximum magnitude was chosen, resulting in feature vectors of size 1440. In total we used 3548 positive examples and 3795 negative examples. For training, we randomly chose 200 positive and 200 negative images from the positive and negative image sets and used the remaining images for testing. This procedure was repeated several times and we averaged the acquired accuracies to calculate the accuracy of the classifier. In Fig. 3.8 a set of positive (pedestrian) and negative (background) examples extracted from the INRIA dataset are depicted.

Figure 3.8.: A sample of the positive and negative image examples used from the INRIA dataset.

A set of bases images obtained using the proposed MNMF are shown in Fig. 3.9. For comparison reasons, we also depict the equivalent bases images acquired when using classic NMF. As we can see, the proposed algorithm results in bases images that have good localization characteristics.

(a) (b)

Figure 3.9.: An example of the bases acquired for (a) NMF and (b) proposed MNMF algorithms.

In order to compare the performance of our algorithm with semi-NMF and KNMF, we report the classification performance of the unsupervised scenarios , *i.e.*, SVM classifiers trained with the features extracted using the Semi-NMF and the KNMF. We also report the classification performance of DNMF combined with a KNN classifier. The classification performance for different number of bases considered, that is for various values of k, is summarized in Fig. 3.10. We note that for each k we repeat the experiments with different training sets sampled from the main dataset and report the average accuracy over all runs. For simplicity, for each value of k, we used the value of C that provided the best results for the semi-NMF+SVM algorithm as input for all the rest of the methods tested (including ours). It can be seen that the proposed method clearly outperforms all other methods in terms of the recognition accuracy for all values of k.

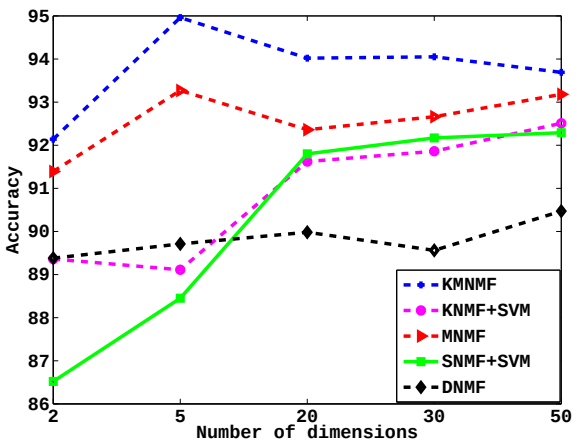


Figure 3.10.: The accuracy obtained for the INRIA dataset.

Table 3.1.: Comparison of MNMF and KMNMF with baseline techniques (INRIA dataset).

Method	SNMF + SVM	DNMF + KNN	KNMF + SVM	MNMF	KMNMF
Accuracy	92.29%	90.47%	92.51%	93.27%	94.96%

In Table 3.1 we report the performance of the proposed MNMF and KMNMF algorithms as well as that of the baseline techniques in terms of recognition accuracy. As we can see, both the proposed MNMF and KMNMF outperform the baseline techniques.

3.5.2. BU-3DFE Dataset

We then applied the proposed algorithms on the BU-3DFE [129] facial expression dataset. To extract features, we calculated the difference image for each subject, by subtracting the image corresponding to the neutral state from the equivalent image corresponding to the fully formed expression (highest intensity). A Gabor filter [72] of 2 scales and 4 orientations was applied on these difference images to yield feature vectors of size 7680×1.

For testing, we adopted a five-fold cross validation protocol. The dataset of 100 subjects was divided into 5 non overlapping groups of 20 subjects each. We used images from one group to form the test set and the images corresponding to the remaining 4 groups to create the training set. This procedure was repeated 5 times so that each group would be used for testing and the average classification accuracy was considered for the classifier. The number of bases (k) was set to 100. In order to perform multi-class classification, we employed a All-versus-All approach, *i.e.*, a binary classifier was built for each pair of classes. During testing, the class that received the maximum number of votes won.

In Table 3.2 we report the classification accuracy of the proposed MNMF and KMNMF, as well as that of some baseline techniques (Semi-NMF and KNMF followed by SVM and DNMF followed by KNN). As we can see, the proposed MNMF and KMNMF both outperform the baseline techniques SNMF + SVM, DNMF + KNN and KNMF + SVM, with accuracy rates equal to 69.00%, 71.67%, 66.5%, 65.00% and 70.00%, respectively. Therefore, the introduction of the maximum margin constraints within the factorization procedure efficiently leads to better performance in terms of recognition accuracy.

Table 3.2.: Comparison of MNMF and KMNMF with baseline techniques (BU-3DFE dataset).

Method	SNMF + SVM	DNMF + KNN	KNMF + SVM	MNMF	KMNMF
Accuracy	66.5%	65.00%	70.00%	69.00%	71.67%

Table 3.3.: Confusion matrices for the proposed MNMF and KMNMF.

%	anger	disgust	fear	happiness	sadness	surprise
anger	60	7.5	5	1.5	24.5	1.5
disgust	11.5	66	10	3.5	6	3
fear	7.5	7.5	57	10.5	11	6.5
happiness	1.5	6	6.5	83	1.5	1.5
sadness	21	2.5	9	1	65.5	1
surprise	4.5	4.5	5	1	3	82

%	anger	disgust	fear	happiness	sadness	surprise
anger	59	8	4.5	1	27	0.5
disgust	10	68	8	3.5	7	3.5
fear	7.5	5.5	66	6.5	10.5	4
happiness	1	4.5	8.5	84.5	1	0.5
sadness	16.5	3.5	9	0.5	69	1.5
surprise	4.5	3.5	5.5	0.5	2.5	83.5

3.5.3. Mediamill Dataset

We next studied the performance of the proposed algorithms on object classification using the Mediamill [106] dataset. For training, we randomly chose 200 images from each class and used the remaining images for testing. This procedure was repeated several times and the average classification accuracy was regarded as the classification accuracy of the classifier.

The nonlinear kernel parameter σ in Eq. 3.32 was chosen to be same for both the proposed KMNMF and KNMF [134] algorithms, to ensure a fair comparison. As baseline techniques, we again used the Semi-NMF and KNMF algorithms followed by a SVM classifier and the DNMF algorithm followed by a KNN classifier. We should note here that for each k, we repeat the experiments with different training sets sampled from the main dataset and report the average accuracy over all the runs. For all experiments, for each value of k, we used the value of C (100) that provided the best results for the KNMF+SVM algorithms.

The classification performance for different number of bases (k) is summarized in Fig. 3.11. It can be seen that the proposed method clearly outperforms all other methods in terms of the classification error for all values of k. In particular, we notice that the proposed method significantly outperforms other methods when the number of basis vectors is small.

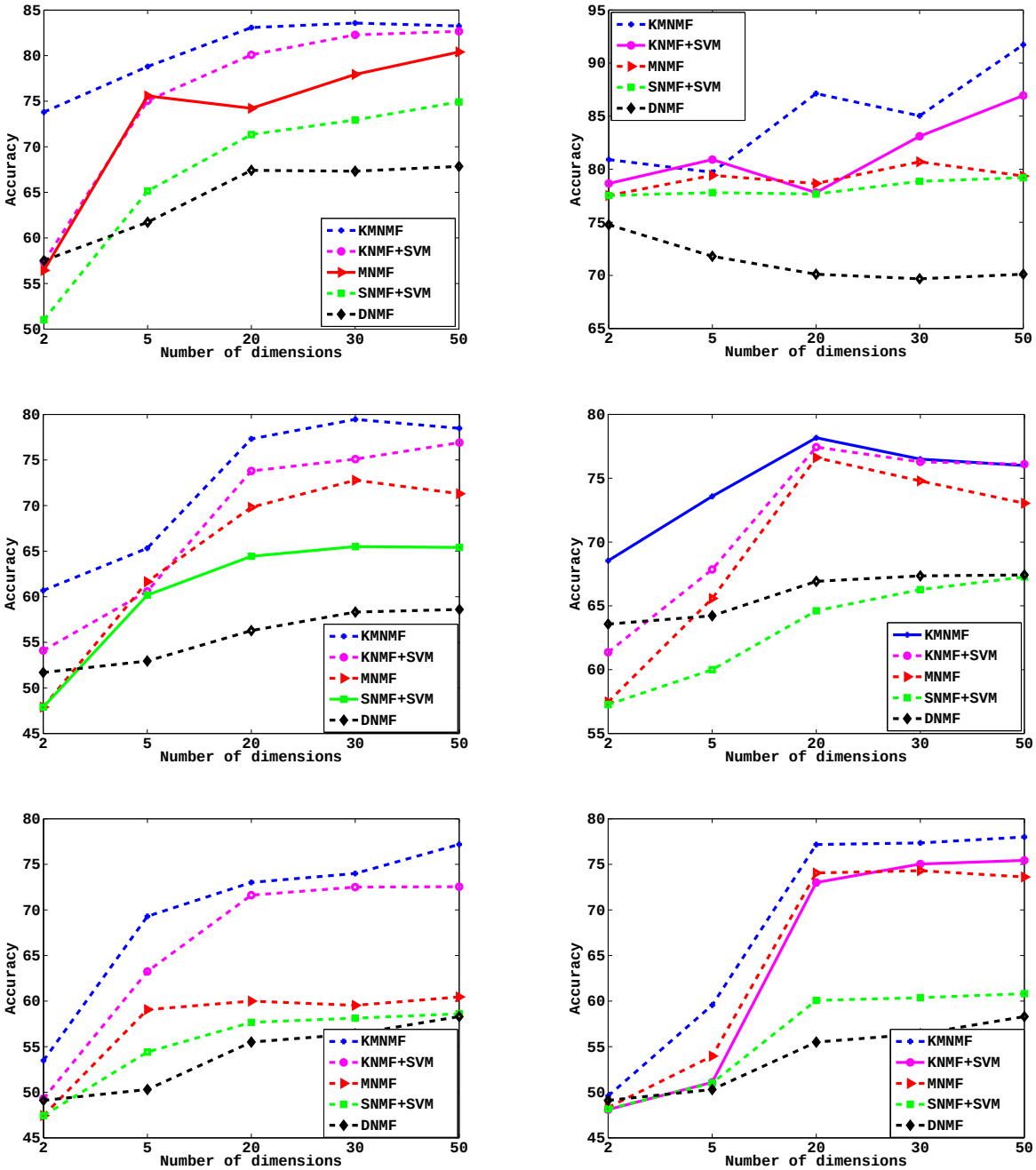


Figure 3.11.: Comparison of the performance of the MNMF and KMNMF algorithms with DNMF +KNN [133] , Semi-NMF [29] + SVM, KNMF [134] + SVM versus the number of bases k for different categories of Mediamill dataset.

Table 3.4.: Confusion matrices of the proposed MNMF and KMNMF on the KTH dataset

%	box	clap	wave	jog	run	walk
box	90	2	2	2	2	2
clap	4	88	7	1	0	0
wave	5	3	92	0	0	0
jog	0	1	1	87	11	0
run	1	0	0	8	89	2
walk	1	0	0	1	3	95

%	box	clap	wave	jog	run	walk
box	92	2	2	2	0	2
clap	1	91	7	1	0	0
wave	1	6	92	1	0	0
jog	0	1	0	84	13	2
run	0	0	0	4	94	2
walk	0	1	1	0	0	98

3.5.4. KTH Action Dataset

Subsequently, we studied the performance of the proposed algorithm on the KTH dataset. For each action, we considered a period of 9 'naively' chosen frames (not time-scaled). In order to have a better alignment for the training data we extracted bounding boxes of size 60×80 around the objects. Possible illumination changes were handled using as features HOGs. The HOGs were extracted as described in Section 3.5.1. The number of bases k was set 135. In order to perform multi-class classification we employed the same all-versus-all strategy that we adopted in the experiments conducted for the BU-3DFE database. The leave-one-person-out cross validation approach was used to test the performance of the algorithms.

Table 3.4 shows the confusion matrices for the proposed MNMF and KMNMF on the KTH action dataset. From Table 3.4, we notice that both the algorithms achieve low performance for the action *jog* and *run*. This is due to the similarity between the actions *jog* and *run*. Since we only consider the appearance of the actions (intra frame information) and ignore the temporal information (inter frame information), the algorithm achieves low accuracy for theses classes. We also note that KMNMF outperforms the MNMF for all classes except *jog* which suggests that the classes in the KTH dataset are not linearly separable.

In Table 3.5 we report the performance of the proposed MNMF and KMNMF algorithms as well as that of the baseline techniques (SNMF + SVM, KNMF + SVM and DNMF + KNN) in terms of recognition accuracy. As we can see, both the proposed MNMF and KMNMF outperform the baseline techniques. KMNMF also introduces a 1.66% increase in recognition accuracy over MNMF, something that implies that the

Table 3.5.: Comparison of MNMF and KMNMF with baseline techniques (KTH dataset).

Method	SNMF + SVM	DNMF + KNN	KNMF+SVM	MNMF	KMNMF
Accuracy	87.50%	84.00%	91.30%	90.17%	91.83%

classes included in the KTH dataset are better separated when non-linear techniques are used.

3.5.5. Effect of Parameter λ

Having already shown that the proposed MNMF and KMNMF algorithms outperform schemes that employ factorization techniques followed by a SVM classifier, we further study the effect of the parameter λ on the cost function Eq. 3.8 and on ξ_i. As we have defined in Eq. 3.13, the weight for the NMF cost λ is inversely proportional to the tuning parameter θ, *i.e.*, $\theta = C/\lambda$. High values of λ result in low values of α_i, since α_i is upper bounded by θ. Thus for high values of λ the second term in Eq. 3.12 approaches zero and the update rules for G are similar to the update rules of Semi-NMF [29] while for lower values of λ the second term in Eq. 3.12 is introduced formulating in that way our framework. In Fig. 3.12 we plot the classification accuracy achieved for the train and test sets of the KTH dataset versus the value $\log(\lambda)$. As we can see the proposed MNMF achieves a higher training and testing classification accuracy when a smaller value for the parameter λ is considered. For larger values of λ the classification accuracy of MNMF converges to the same value with that of SNMF followed by SVM.

3.6. Conclusions

In this chapter, we propose a maximum margin framework for the linear and non-linear Non-negative Matrix Factorization algorithm. Our aim is to impose soft max-margin constraints on the cost function of NMF in order to calculate the decomposition matrices that will enable us to perform classification while reducing simultaneously both the reconstruction and misclassification errors. In that way, we obtain the maximum margin of the classifier in the low or high dimensional space (for the case of linear and non linear NMF, respectively) while ensuring that it achieves the highest classification accuracy. To accomplish this, we formulate a novel cost function that combines the reconstruc-

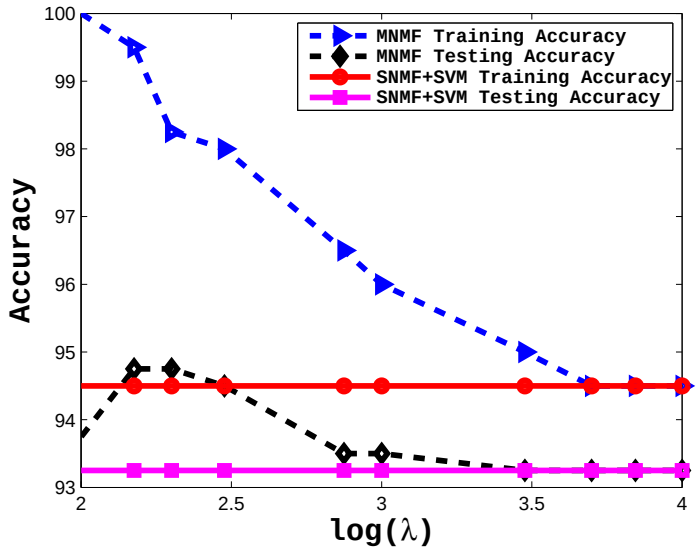


Figure 3.12.: Classification accuracy versus $\log(\lambda)$.

tion error term introduced by the matricization algorithm and the misclassification error introduced by the maximum margin classifier, bound together under SVM-type linear inequality constraints. The introduced cost function formulates a non-convex constrained optimization problem with respect to the bases and the separating hyperplane, which we solve following an iterative optimization procedure. At each iteration we solve for a set of convex (constrained quadratic or Support Vector Machine-type) sub-problems employing typical quadratic programming tools. We demonstrate the performance of the proposed algorithms on several computer vision problems such as pedestrian detection, image retrieval, facial expression recognition and action recognition using not only toy datasets but also publicly available ones (the INRIA, the BU-3DFE, the KTH action and the Mediamill datasets).

Chapter 4.

Learning Dictionary Weights for Action Localization

4.1. Introduction

The task of detecting actions in video sequences is challenging as the system is not only required to distinguish the action from other action categories but also locate the actions in the video. A good system for such task should have a model of the action with low intra-class and high inter-class variability. In addition a robust system should be able to perform well under various real world conditions such as illumination changes, variations in the size and speed of the action, occlusion, cluttered background etc. While the appearance of the action can dramatically change for different instances of the same class under above conditions, it has been observed that the appearance of the small local parts are less variable. This calls for a part-based model similar to [4, 61, 85] where the object is decomposed into parts or components and the local appearance of theses parts are modeled independently. The Implicit Shape Model [61] is a part-based model where the parts of the object provide information about the spatial location, scale and size of the object. i.e during training, the spatial configuration of the parts are learned relative to the object center. Although the spatial configuration of different training instances may vary significantly, the model is flexible enough to combine parts from different training examples to represent a test object with novel articulation and thus the model can be learned from a few examples. In this work we employ the Implicit Shape Model (ISM) [61] for detecting actions in video sequences. As the generative approach of the ISM does not consider the discriminative information for separating the object from

the background, we introduce discriminative weights for the dictionary learned from the training sequences.

The rest of the chapter is organized as follows. We briefly describe the ISM in the section Section 4.2. We give an overview of the proposed method in Section 4.3. In Section 4.4, we discuss the global weights, local weights and the discriminative votemaps. Section 4.5 describes the resulting objective function and the training and testing algorithms. We discuss the experimental results in Section 4.6 and finally we conclude in Section 4.7.

4.2. Implicit Shape Model

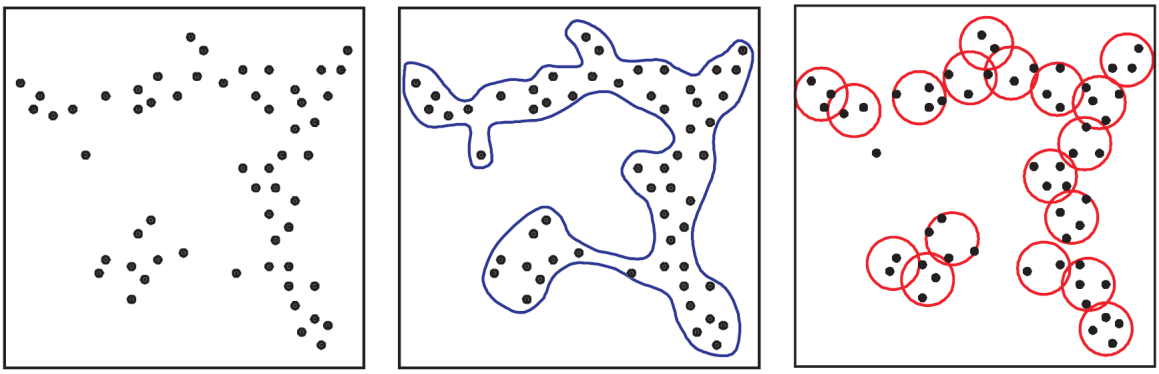
The Implicit Shape Model proposed by Leibe *et al.* [60,61] for combined object detection and segmentation is one of the popular part-based approaches. As we are interested in detection aspect of the ISM in this thesis, we will only describe how the shape models are learned and used for object detection task. The ISM framework can be divided into two stages namely

- Codebook generation
- Learning spatial occurrence distribution

4.2.1. Codebook generation

The wide usage of part-based methods for object detection can be devoted to the following facts [43]: (i) Most of the object categories can be represented by a few characteristic object parts and their geometrical relationships; (ii) the appearance of some object parts vary less under pose changes as compared to the whole object; and (iii) part-based approaches are less susceptible to occlusion than the whole object .

These methods can be broadly classified into two categories based on the approach used to select parts. The methods in [43,85,95] follow a part-classifier based approach in which a limited number of hand-picked components of the object are treated as parts and a complex classifier is learned for each part. These methods attempt to learn a robust detector for semantically meaningful parts so that objects' presence can be detected from a few parts. Since a single detector is learned for each part, these methods require a large number of training samples (positive and negative) to capture all the variations



(a) (b) (c)

Figure 4.1.: Codebook representation [59]. (a) The points represent the appearance distribution of some object part. (b) The methods in [43, 85, 95] try to find complex decision surface that separates all part appearances from non-part appearances. (c) The codebook approach represents the appearance distribution by a set of compact prototypes

in the parts. In contrast to this, the methods in [4, 34, 36, 61, 120] follow a codebook [1] approach for building the part model. The codebook is a vocabulary of local descriptors (or parts). For a given training set, the local descriptors of the objects are extracted at interest points and these local descriptors are grouped based on the visual similarity using an unsupervised clustering algorithm. The codebook approach results in a large number of simple and compact appearance prototypes that represent the complex appearance distribution of the part.

Fig. 4.1 describes the method employed to model the parts in the above approaches. Let 4.1 (a) represent a complex appearance distribution of some object. Instead of trying to find a complex distribution that tries to discriminate the part and non-part appearances [43, 85, 95] as in 4.1 (b), the codebook approach represents the appearance distribution of parts by compact prototypes as shown in 4.1 (c). A point is classified as belonging to a certain prototype if the distance between the point and the prototype is less than a threshold. Learning a large number of prototypes in an unsupervised manner also enables the codebook approach to use far more object parts than when the parts need to be manually annotated. This is very useful when the object contains a large number of semantically meaningful parts.

To learn a codebook during the training stage, a large number of local patches are extracted from the images using interest point detectors. Extracting the patches at the interest point detectors results in the parts or components that can characterize

[1] We will use the terms *dictionary* and *codebook* interchangeably in the rest of the chapter.

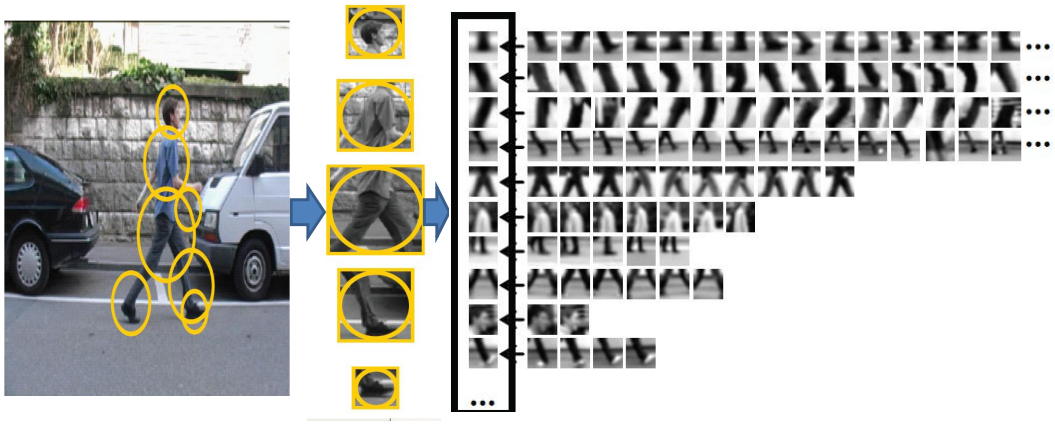


Figure 4.2.: Codebook Generation [61]: The patches are extracted from the training images and clustered using an unsupervised clustering algorithm. The resultant codewords or cluster centers are shown

the objects and also reduces the number of patches to be processed. In addition the repeatability property of the interest point detectors enables us to extract similar patches in different objects. These extracted patches are grouped based on the visual similarity using an unsupervised clustering algorithm to form a small compact prototypes of local appearances (codebook). Fig. 4.2 shows the steps involved in the codebook generation process.

4.2.2. Learning Spatial Occurrence Distribution

The ISM uses the codebook to discriminate the patch appearances and to learn the structural information of the parts. This is done by comparing the training patches against the clusters (also called codewords) in the codebook. The conventional way of matching is assigning the patch to the nearest neighbor, however, it is important consider the uncertainty in matching and propagating the uncertainty forward to the later stages. An important assumption in codebook approach is that a codeword is a characteristic representative of the image patch [114]. Due to continuous nature of the visual appearance, it is difficult to choose a single representative codeword for a patch. In realistic situations, most of the patches are matched to more than one codeword. If the patch is matched to only one codeword then there is no ambiguity but if the patch is matched to more than one codeword, then there is uncertainty called codeword uncertainty [114].

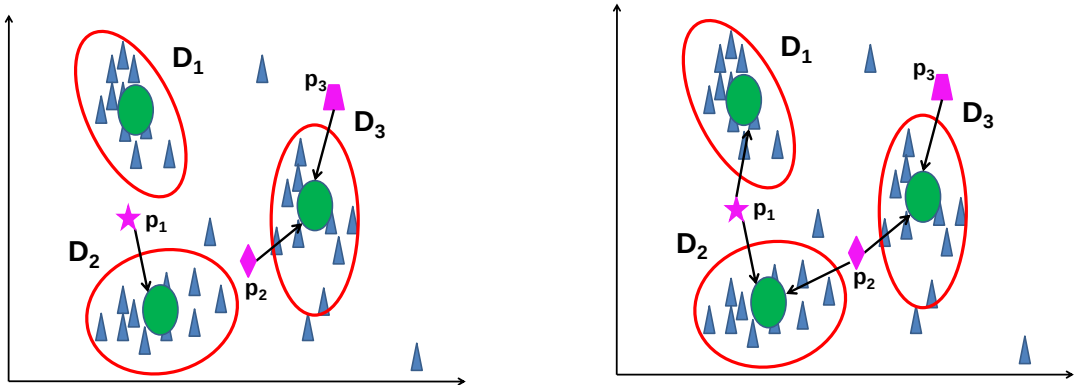


Figure 4.3.: Codeword uncertainty [59]. (a) The local patches $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$ are assigned to the closest codewords $\mathbf{D_2}, \mathbf{D_3}, \mathbf{D_3}$ respectively. This assignment is unstable as a slight change in the patch can result in different codewords being assigned. (b) The patches are assigned to all the codewords within a threshold distance. This is more robust as a slight variation in the patch does not alter the codeword assignment

This illustrated in the Fig. 4.3. In Fig. 4.3 (a), each patch is matched to the closest codeword. The patch $\mathbf{p_3}$ represented by the trapezoid does not have the codeword uncertainty as it is close to only one codeword, *i.e.*, codeword $\mathbf{D_3}$ whereas the patches represented by star $\mathbf{p_1}$ (close to $\mathbf{D_1}$ and $\mathbf{D_2}$) and diamond $\mathbf{p_2}$ (close to $\mathbf{D_2}$ and $\mathbf{D_3}$) have the codeword uncertainty. The matching process in Fig. 4.3 (a) is unstable as a small change in appearance of the patch $\mathbf{p_1}$ will result in matching with the codeword $\mathbf{D_1}$ instead of $\mathbf{D_2}$. Similarly a slight change in the appearance of the patch $\mathbf{p_2}$ will activate the codeword $\mathbf{D_2}$. In order to obtain a stable codebook representation, the matching should consider the codeword uncertainty and propagate the uncertainty to the next stage.

The ISM [60] uses the Normalized cross correlation as the similarity measure for matching, *i.e.*, a codeword is matched to a patch if the similarity between the patch and the codeword is greater than a threshold. If a patch is matched to a codeword, then we store the relative location of the patch w.r.t. the object center and the scale at which the interest point was detected. These stored occurrence locations reflect the spatial distribution of a codeword over the object area in a non-parametric form. The spatial occurrence distribution of a codeword specifies the spatial locations within the object, where the patches associated with the codeword in question can be found. The spatial

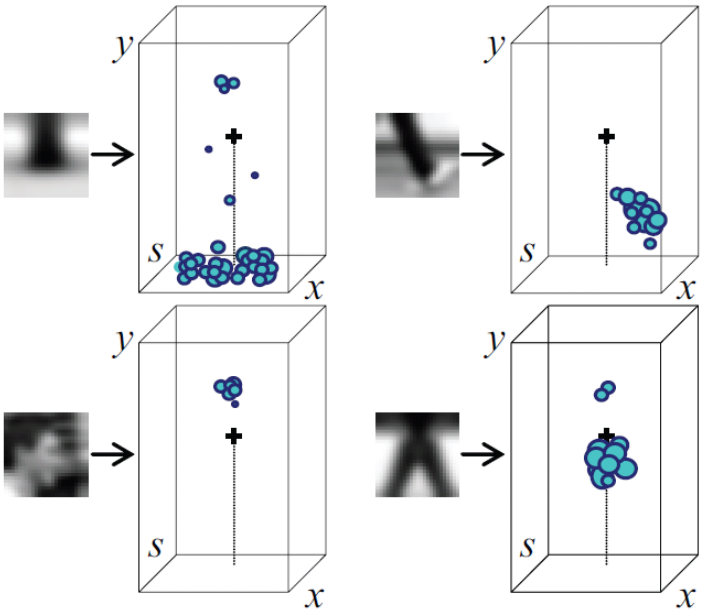


Figure 4.4.: Spatial occurrence distribution for four sample codewords [61]. The spatial occurrence distribution are plotted with x, y and *scale* axes

occurrence distribution for each codeword is estimated in a non-parametric manner. Fig. 4.4 shows the spatial occurrence distribution for four sample codewords.

Algorithm 3: Algorithm to compute the spatial occurrence distribution [61]

```
begin
    foreach Codeword D_i do
        Occ[i] ← ∅    // Initialize occurrence of codeword D_i
    end
    foreach interest regions l_k = (l_x, l_y, l_s) with descriptor f_k do
        foreach Codeword D_i do
            if sim(D_i, f_k) > κ  then
                // Record an occurrence of codeword D_i
                Occ[i] ← Occ[i] ∪ (c_x − l_x, c_y − l_y, l_s)
            end
        end
    end
end
```

4.2.3. Object Detection with Implicit Shape Model

Let $\mathbf{D}$ denote a codebook obtained using k-means algorithm and let $\mathbf{D}_j$ denote j^{th} codeword which corresponds to a cluster center. The spatial occurrence distribution is learned by matching the training patches against the codebook. For all matched patches the scale and spatial information are stored in the codeword as described in Algorithm 3. During testing, the patches of fixed size are extracted from the test images at the locations indicated by the interest point detectors. These patches are compared against codebook with the criterion mentioned above and activate codewords. The votes for the possible center and scale of the object is casted using generalized Hough transform framework [9, 71], *i.e.*, the activated codewords use the spatial and scale information stored during training to cast probabilistic votes in an output space called Hough space. The consistent hypothesis are searched as local maxima in the Hough space using a meanshift mode estimation algorithm [24, 25].

Probabilistic Hough Voting

Let $\mathbf{x}_i$ be a feature located at location l_i of the input image. The probability of matching the patch with j^{th} codeword $\mathbf{D}_j$ can be given by $p(\mathbf{D}_j|\mathbf{x}_i, l_i)$. Every matched codeword can vote for different objects O and locations y. Let $S(O, y|\mathbf{x}_i, l_i)$ denote the probabilistic Hough score collected at location y of the object O from the feature $(\mathbf{x}_i, l_i)$. Then

$$S(O, y|\mathbf{x}_i, l_i) = \sum_j p(O, y|\mathbf{D}_j, \mathbf{x}_i, l_i) p(\mathbf{D}_j|\mathbf{x}_i, l_i) \tag{4.1}$$

Since the matching is independent of the location of the patch, we have $p(\mathbf{D}_j|\mathbf{x}_i, l_i) = p(\mathbf{D}_j|\mathbf{x}_i)$. Also, note that the probabilistic Hough vote from a codeword is independent of the appearance of the input feature, *i.e.*, $p(O, y|\mathbf{D}_j, \mathbf{x}_i, l_i) = p(O, y|\mathbf{D}_j, l_i)$.

$$S(O, y|\mathbf{x}_i, l_i) = \sum_j p(O, y|\mathbf{D}_j, l_i) p(\mathbf{D}_j|\mathbf{x}_i) \tag{4.2}$$

$$= \sum_j p(y|O, \mathbf{D}_j, l_i) p(O|\mathbf{D}_j, l_i) p(\mathbf{D}_j|\mathbf{x}_i) \tag{4.3}$$

The term $p(O, y|\mathbf{D}_j, l_i)$ in Eq. 4.2 is the probabilistic Hough vote for the object location given the codeword and the feature. The term $p(O|\mathbf{D}_j, l_i)$ specifies the probability that the codeword $\mathbf{D}_j$ is matched to the object O as opposed to background and the third term represents the quality of the match between the codeword and the patch.

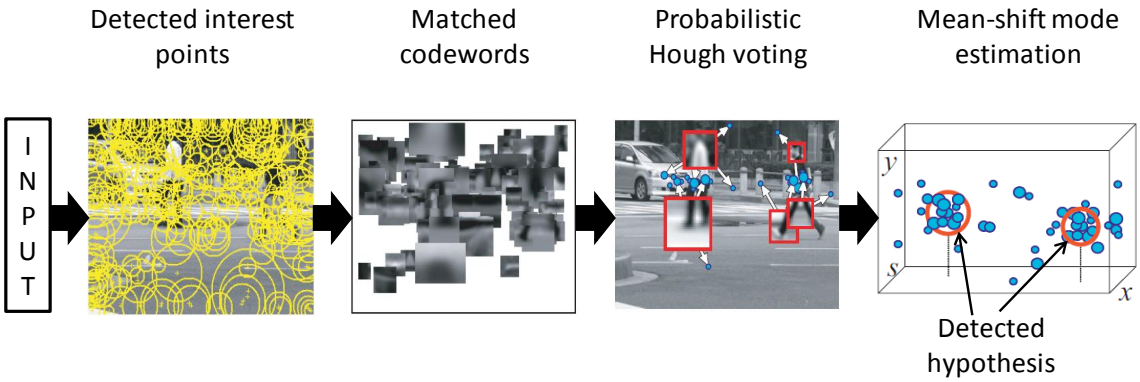


Figure 4.5.: The Recognition procedure with the ISM: The image patches are extracted using interest point detectors and matched against codebook. The activated codewords cast probabilistic Hough votes in the output Hough voting space. A Mean-shift mode estimation algorithm is applied to detect the hypothesis

In order to handle multiple scale, the scale is included as a third dimension in the Hough space. Let the t^{th} patch of an image be located at (x_t, y_t, s_t) where s_t is the scale at which the patch was found. If the patch is matched to the codeword with entry (x_{cw}, y_{cw}, s_{cw}), then the voting location in the Hough space is given by

$$x_{vote} = x_t - x_{cw}(s_t/s_{cw}) \tag{4.4}$$

$$y_{vote} = y_t - y_{cw}(s_t/s_{cw}) \tag{4.5}$$

$$s_{vote} = s_t/s_{cw} \tag{4.6}$$

The above votes are computed for all the learned entries in a codeword. The votes are collected in an output space called Hough space that has the same dimension as the number of parameters to be estimated. The hypothesis corresponds to the maxima of the Hough space. The ISM in [61] employs a uniform distribution for the last term in Eq. 4.2, *i.e.*, $p(\mathbf{D}_j|\mathbf{x}_i) = \frac{1}{\sum_{j=1}^{k} |d(D_j, \mathbf{x}_i) > \kappa|}$ where $|.|$ represents is the indicator function. The probabilistic Hough vote from a codeword is also assumed to follow an uniform distribution, *i.e.*, $p(y|O, \mathbf{D}_j, l_i) = \frac{1}{b_j}$ where b_j is the total number of patches matched to the j^{th} codeword during training. The final voting score is obtained by combining the evidence from all the patches in the test image:

$$score(O, \mathbf{y}) = \sum_i p(O, y|\mathbf{x}_i, l_i) \tag{4.7}$$

The overall steps involved in the detection process is shown in Fig. 4.5

4.3. Learning Discriminative Weights for the Codebook

In practice the nature of the generative models makes it robust to various real world conditions such as partial occlusion, illumination changes, view-point changes and so on. The ISM has a generative probabilistic model which can represent a category of objects in an efficient manner. Since each patch of the image can vote independently of the other patches for the center of the object, the model is flexible enough to combine the local parts seen on different training images. Thus the model can achieve high recall rates for any general object category but this flexibility leads to an increase in the false positive rate [37, 79]. This motivates the use of a discriminative methods along with the ISM. Recently, a number of methods have been proposed that combines the advantages of a generative model with a discriminative model [21, 37, 79, 81, 127, 136, 137].

The method in [81] introduced a spatial weighting approach to discriminate the background from the foreground for object detection. The weights of the features that agree on both the location and the shape object are boosted while the weights of the background features are suppressed thus making the method more robust to background clutter. Zhang *et al.* [136] proposed a boosting approach for the codebook construction where codebooks are learned in a sequential manner by using the discriminative information that was not learned by the previous codebooks and their corresponding classifiers. Cai *et al.* [21] used a codebook weighting approach for image classification based on the criteria that the weighted similarity between the same labeled images is greater than that between differently labeled images with largest margin.

In particular the flexibility of the ISM has motivated the researches to develop methods [37, 79, 127, 137] that combine the representative power of ISM with a discriminative model. In [127] a set of decision trees were used to learn a mapping between the densely sampled feature patches and the corresponding votes in the spatio-temporal Hough space. The leaf nodes of the trees are the discriminative codebooks that store information about the location of the object of interest. A common appearance codebook for the generative and discriminative models is learned in [37]. For a query image, the generative part of the algorithm produces several hypothesis using the appearance codebook and these generated hypothesis are verified by the discriminative part of the algorithm using the same codebook activations. The method in [137] puts the Hough transform into a max-margin framework such that the Hough transform detector can be

used to obtain the decision scores of the svm classifier at every location and scale of the image. Maji *et al*. [79] used a max-margin approach to learn weights for the codewords by maximizing the Hough voting response at the correct locations of the object over the incorrect ones. Motivated by these works, we propose a codebook weighting approach in which the local parts cast weighted votes to the center and scale of the action. In addition the success of the soft quantization over hard quantization also motivated us to develop a local weight for the codewords. The objective of the localization is to discriminate the object locations from the non-object locations. Since the spatio-temporal location of the action is known during training, we can compute the weighted votes for the location of the action and the background for each training feature and use this information to efficiently compute the global weights for the codebook. Summarizing, the main contributions of this chapter are:

- We present an approach to learn local weights for the matched codewords of each feature based on the degree of match between the codeword and the feature. This is achieved in a principled way by incorporating the Hough voting scheme into Locality Constrained Linear Coding (LLC) [118] framework.
- We develop a framework that enables us to incorporate the discriminative output Hough space information into training, *i.e*., we compute the contribution of votes from each codeword to the Hough space of the training sequences. This information is used to quantify the contribution of votes from each codeword to the center of the action and other locations.
- Using the above framework, we propose a discriminative weighting scheme to learn global weights for the codebook that maximize the Hough voting response at the spatio-temporal location of the activity compared to background.

4.4. Proposed Framework

Most of the methods mentioned in Section 4.3 [60, 79, 137] use a codebook learned using k-means algorithm which has been the most popular and widely used codebook generation method. However, it is reported in literature [21, 46, 114] that k-means codebook results in codewords that are highly biased towards the high density regions as compared to the low density regions and also it is noted by [15, 46] that the most frequent features are not essentially the most discriminative features. This leads to codebook with

redundant codewords and can increase the number of false positives if the codewords are uniformly weighted. To overcome this, codebook weighting approaches are used that discriminate the informative codewords from the non-informative ones [21, 79]. Further Gemert *et al.* [40, 114] showed that instead of using uniform weights for the codewords, soft assignment to a few codewords based on degree of matching can improve the performance. In this work we use the k-means codebook for detection of actions in videos and propose a codebook weighting approach based on the Hough voting spaces of the training sequences.

The ISM [61] learns a set of codewords from training patches and keeps track of the location of the codewords with respect to the object centre. During testing, the test patches are matched against the codewords and the matched codewords cast probabilistic votes to the object centre. Let $\mathbf{x}_i$ be a feature located at location l_i of the input image. The patch is matched against the j^{th} codeword $\mathbf{D}_j$ with a probability $p(\mathbf{D}_j|\mathbf{x}_i, l_i)$. As described in Section 4.2.3, the Hough score $S(O, y|\mathbf{x}_i, l_i)$ collected at location y of the object O from the feature $(\mathbf{x}_i, l_i)$ is given by

$$S(O, y|\mathbf{x}_i, l_i) = \sum_j p(y|O, \mathbf{D}_j, l_i) p(O|\mathbf{D}_j, l_i) p(\mathbf{D}_j|\mathbf{x}_i) \tag{4.8}$$

Many variations of the Hough voting scheme have been proposed [79, 127, 137]. In [79] the second term is learned discriminatingly using Max-Margin classifier whereas the approach in [127] discriminatingly learns the third term using Hough forests. The method in [137] reformulate the Hough voting scheme so as to fit it to the kernel classifier. The third term is computed for each local feature of the input video. Hence we call it as the local weights for the votes. Similarly second term can be regarded as the global weight for the votes. In this work, we propose a method to learn global weights (second term) and also the local weights (third term) for the codebook.

4.4.1. Learning Local Weights

In the ISM, the local patches are compared with multiple codewords in order to cope up the ambiguity generated by mapping the continuous image patch to discrete visual codewords. Recently there has been an increasing interest in the machine learning community to learn the mapping between the local features and codewords to which the features are assigned. This can be also referred to as the coding step [17] where the input features are locally transformed into representations with some desirable properties such

as compactness, sparseness etc. The codes can also be interpreted as the coefficients obtained by decomposing the feature on a codebook or as the local weights for the codewords. Let $\mathbf{x}_i$ denote a local feature and $\mathbf{h}_i$ denote the corresponding coefficient and $\mathbf{D}_j$ denote the j^{th} codeword. There are several ways of assigning a local feature to the codeword [17, 68]. They are:

Hard-assignment coding: The feature $\mathbf{x}_i$ is matched to a single nearest codeword. Hence only one entry in $\mathbf{h}_i$ is one and all other entries are zero, *i.e.*,

$$\mathbf{h}_{ij} = \begin{cases} 1 & j = \arg\min_{j=1,2\cdots K} \|\mathbf{x}_i - \mathbf{D}_j\|^2 \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

Soft-assignment coding: In contrast to hard-assignment, the soft-assignment distribute the probability mass to multiple codewords based on the degree of match between the local feature and the codeword.

$$\mathbf{h}_{ij} = \frac{\exp(-\beta\|\mathbf{x}_i - \mathbf{D}_j\|^2)}{\sum_{k=1}^{K} \exp(-\beta\|\mathbf{x}_i - \mathbf{D}_k\|^2)} \tag{4.10}$$

where β is the smoothing factor.

Sparse Coding: In sparse coding, the local feature is expressed as a linear combination of a few codewords. The objective function minimizes a combination of the ℓ_2 norm of the reconstruction error and ℓ_1 norm of the coefficient vector $\mathbf{h}_i$ [17, 124, 125],

$$\widehat{\mathbf{h}_i} = \arg\min_{\mathbf{h}_i} \|\mathbf{x}_i - \mathbf{D}\mathbf{h}_i\|^2 + \lambda\|\mathbf{h}_i\|_1 \tag{4.11}$$

where $\|\mathbf{h}_i\|_1$ denotes the ℓ_1 norm of $\mathbf{h}_i$, λ is the regularization parameter that controls the sparsity of $\mathbf{h}_i$ and $\mathbf{D}$ denotes the codebook or dictionary of the local features.

Gemert *et al.* [114] termed the ambiguity in assigning the local feature to multiple codewords as codeword uncertainty and proposed soft-assignment coding as given in Eq. 4.10. It was also shown in [17, 114] that soft-assignment improves the recognition performance over hard-assignment. Further it was shown in [17] that sparse coding improves over soft quantization (soft-assignment coding). However, the nature of the regularization term in sparse coding makes it highly sensitive [39] to the data, *i.e.*, ℓ_1 term is not smooth [118]. This causes the similar local features to have different coefficient vectors thus losing the correlation between the codes. Yu *et al.* [131] empirically

pointed out that the sparse codes need not necessarily select the codewords that are local to the feature and presented a technique called Local Coordinate Coding (LCC) which explicitly constrains the codewords to be local. They also suggested that under certain assumptions locality is more important than sparsity [118] for successful nonlinear function learning. But similar to sparse coding [124], it is computationally expensive as it requires to solve ℓ_1 minimization problem. Recently Wang *et al.* proposed Locality Constrained Linear Coding (LLC) [118] that remarkably outperformed the sparse coding [124] techniques. LLC assigns the features to the local codewords by imposing locality constraint on the codes. The success of the local coding methods such as [118,131] justifies the assumption that local features lie on a low dimensional manifold in an ambient descriptor space [68] and also highlight the importance of assigning the features to the local codewords. In this work we employ LLC to learn local weights for the codewords. We briefly describe the LLC in the next section.

Locality Constrained Linear Coding

The LLC [118] imposes locality constraint on the codes in contrast to sparseness constraints [124, 125]. This results in low weights for the codewords that are far away from the feature. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ denote a set of input data vectors and let $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_k] \in \mathbb{R}^{m \times k}$ denote a dictionary or codebook such that the data vector can be approximately represented as a linear combination of a few columns of $\mathbf{D}$, *i.e.*, $\mathbf{x}_i \approx \mathbf{D}\mathbf{h}_i$, $\mathbf{h}_i$ being the sparse representation of the sample. The LLC algorithm in [118] uses the locality criteria to obtain a representation of the sample, *i.e.*,

$$\arg\min_{\mathbf{h}_i} \|\mathbf{x}_i - \mathbf{D}\mathbf{h}_i\|^2 + \lambda \|\mathbf{g}_i \odot \mathbf{h}_i\|^2 \quad s.t \;\; \mathbf{1}^T\mathbf{h}_i = 1, \;\; \forall i \tag{4.12}$$

where $\odot$ denotes element-wise multiplication. The weights $\mathbf{g}_i$ is given by

$$\mathbf{g}_i = \left[\exp\left(\frac{\|\mathbf{x}_i - \mathbf{D}_1\|_2^2}{\sigma}\right), \exp\left(\frac{\|\mathbf{x}_i - \mathbf{D}_2\|_2^2}{\sigma}\right), \cdots, \exp\left(\frac{\|\mathbf{x}_i - \mathbf{D}_k\|_2^2}{\sigma}\right)\right] \tag{4.13}$$

σ is used for adjusting weight decay speed. The constraint $\mathbf{1}^T\mathbf{h}_i$ is for the shift-invariant requirements of the LLC code. Fig. 4.6 shows the codeword assignment scheme for different coding techniques. Fig. 4.6a shows the codeword assignment in the case of the Hard assignment coding where the feature is assigned to a single nearest codeword. Fig. 4.6b shows the sparse coding technique in which the feature is assigned to a few codewords of the codebook. Note that the codewords need not necessarily be the local

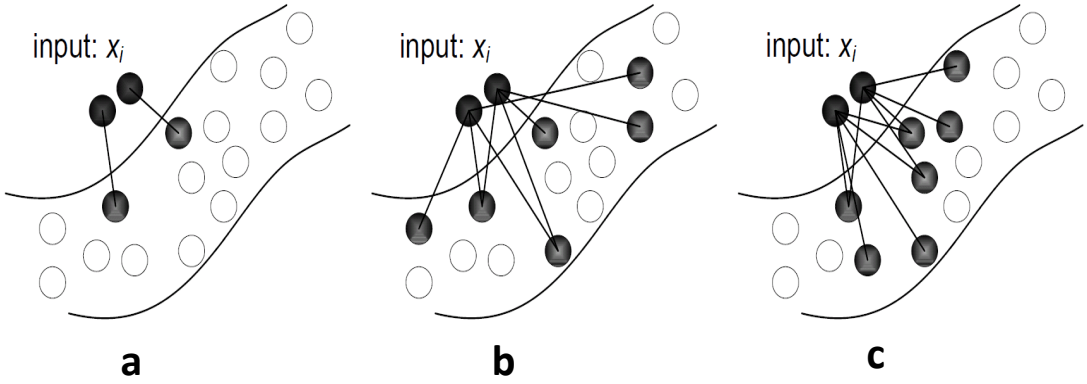


Figure 4.6.: Assignment of codewords for different coding techniques [118]: **a.** Hard-assignment coding in which a feature vector is assigned to the single nearest codeword. **b.** Sparse coding where the feature is assigned to only few (not necessarily the local) codewords of the codebook. **c.** LLC assigns the features to the nearest codewords

codewords. Fig. 4.6c shows the LLC where the codewords are assigned to the local codewords.

Solving for $\mathbf{h}_i$ in Eq. 4.12 results in coefficients that are sparse denoting that the data vectors can be approximated by a linear combination of few local dictionary atoms. Hence instead of solving for $\mathbf{h}_i \in \mathbb{R}^k$ in Eq. 4.12, the authors also proposed an approximated LLC method where data vector is represented as a linear combination of q nearest dictionary atoms and the coefficients $\mathbf{h}_i$ are obtained as

$$\widehat{\mathbf{h}_i} = \arg\min_{\mathbf{h}_i} \|\mathbf{x}_i - \mathbf{D}^i\mathbf{h}_i\|^2 \quad \text{s.t} \quad \mathbf{1}^T\mathbf{h}_i = 1, \quad \forall i \tag{4.14}$$

where $\mathbf{D}^i \in \mathbb{R}^{m\times q}$ is a matrix containing q nearest codewords of $\mathbf{x}_i$. Typically the number of nearest codewords q is much lesser than the total number of codewords k and hence reduces the computational complexity.

The probabilistic Hough score for an object O at a location y given a feature $\mathbf{x}_i$ is expressed as in Eq. 4.8. It is also evident from Eq. 4.8 that ISM can cope with codeword uncertainty as the local features are assigned to multiple codewords as shown in Fig. 4.3. However, Leibe *et al.* [60, 61] assumed a uniform distribution for $p(\mathbf{D}_j|\mathbf{x}_i)$, *i.e.*, $p(\mathbf{D}_j|\mathbf{x}_i) = \frac{1}{\sum_{j=1}^{k} |d(\mathbf{D}_j,\mathbf{x}_i)<\kappa|}$ where $|.|$ represents the indicator function. This causes all the matched codewords to vote with same weight irrespective of the degree of match between the local feature and the codeword. This is illustrated in Fig. 4.7. The diamond represents a local feature and $\mathbf{D_1}$ - $\mathbf{D_7}$ represent the codewords around the feature in

the feature space. The codewords $\mathbf{D_1}$ - $\mathbf{D_5}$ are matched to the feature. Similar to [61], we say a feature $\mathbf{x}_i$ is matched to a codeword $\mathbf{D}_j$ if the distance between the feature and the codeword is less than a pre-defined threshold κ, *i.e.*, $d(\mathbf{D}_j, \mathbf{x}_i) < \kappa$, the function d is Euclidean distance in our case. Fig. 4.7a shows the assignment of weights by the ISM where equal weights $\mathbf{m}$ are assigned to all the matched codewords ($\mathbf{D_1}$ - $\mathbf{D_5}$). Note that weighting scheme adopted by the ISM does not consider the distance between the feature and the codeword. For *E.g.* the distance between the feature and the codeword $\mathbf{D_1}$ is greater than that between the feature and the codeword $\mathbf{D_2}$, *i.e.*, $d(\mathbf{D_1}, \mathbf{x}_i) > d(\mathbf{D_2}, \mathbf{x}_i)$ but the ISM assigns equal weights to the votes from these codewords. Thus the votes from codewords that are consistent in location and appearance with the feature are equally weighted as the votes from the ones that are inconsistent with the feature. This can lead to votes at inconsistent locations in the output Hough space and can increase the false positives. To overcome this, we assign weights to the codewords based on its degree of matching with the feature.

The ISM also suggested a soft-assignment scheme for assigning weights to the codewords. However, [17, 131] showed that LCC is better than the soft-assignment scheme. In addition, the studies in [68] show that the assignment of weights by considering the local manifold leads to improvement in the performance. In this work we design a framework that assign weights in a principled way by incorporating the Hough voting scheme into LLC framework.

Let $\mathbf{h}_i \in \mathbb{R}^k$ denote the weight vector for a feature $\mathbf{x}_i$. We add a constraint $\mathbf{c}^T\mathbf{h}_i = 1$ to satisfy the local manifold assumption in [68, 110] which states that the Euclidean distance is meaningful within a local region where it can approximate the geodesic distance well. The parameter $\mathbf{c} \in \mathbb{R}^k$ is given by

$$\mathbf{c}_j = \begin{cases} 1 & \text{if } dist(\mathbf{D}_j, \mathbf{x}_i) < \kappa \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

Thus the probabilistic Hough score in Eq. 4.2 can be written as

$$S(O, y|\mathbf{x}_i, l_i) = \sum_{j=1}^{k} p(O, y|\mathbf{D}_j, l_i) p(\mathbf{D}_j|\mathbf{x}_i) \tag{4.16}$$

$$= \sum_{j=1}^{k} p(O, y|\mathbf{D}_j, l_i) h_{ij} = \mathbf{b}_i^T(y)\mathbf{h}_i \tag{4.17}$$

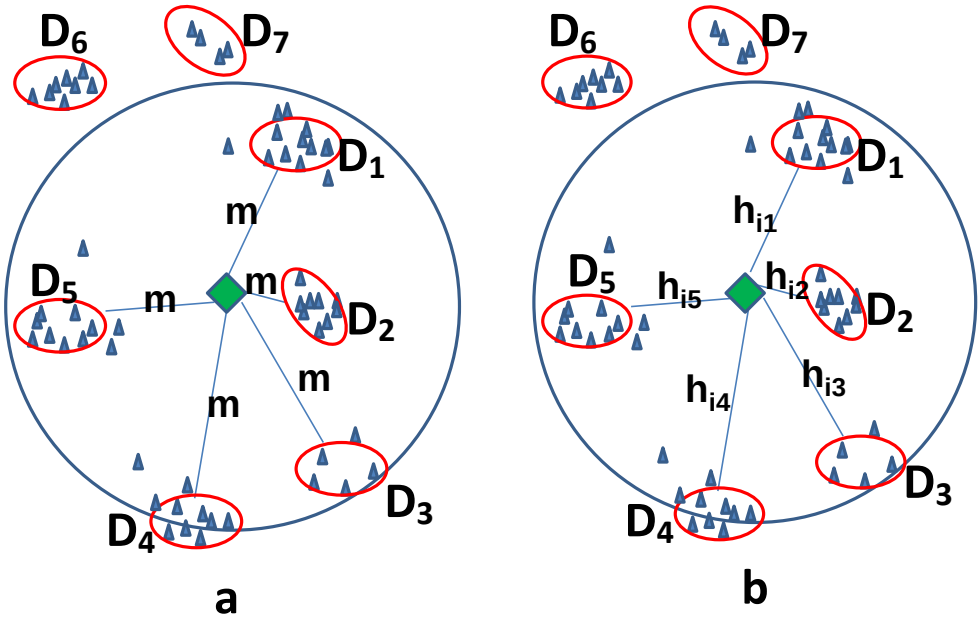


Figure 4.7.: Assignment of weights to the codewords: The diamond represents the local feature and the and $\mathbf{D_1}$ - $\mathbf{D_7}$ represent the codewords. The codewords $\mathbf{D_1}$ - $\mathbf{D_5}$ are matched to the feature. **a)** The ISM assigns equal weights (**m**) to all the matched codewords. **b)** The proposed method employs LLC to assign the weights based on the degree of match between the feature and the codewords.

where $\mathbf{b}_i(y) \in \mathbb{R}^k$ is a vector with the spatial probability entries for the codewords that are matched to the feature,

$$b_{ij}(y) = \begin{cases} \frac{1}{|\mathbf{D}_j|} & \text{if } d(\mathbf{D}_j, \mathbf{x}_i) < \kappa \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

Here $\mathbf{b}_i(y)$ is the vote map for the i^{th} feature and $|\mathbf{D}_j|$ is the number of features that are matched matched to j^{th} codeword during training.

4.4.2. Learning Global Weights

As discussed in the previous section, the LLC leads to a local weighting for votes from the feature as the weights are feature specific, *i.e.*, the weights computed for a feature only depend on the degree of match between that feature and the codebook and independent of the other features. In this section, we discuss an approach to assign global weights for the votes from each codeword. We use a k-means codebook learned using the appearance of the training features. The construction of the codebook does not take the location

information of the features into consideration and hence may not be optimal for the task of localization. Also, as discussed in Section 4.4, k-means leads to more codewords at high density regions and lacks codewords at low density regions. Thus the contribution of votes from the codewords to the object center will not be uniform. This calls for a weighting scheme for the votes from the codewords such that the learned weights are optimized for localization. Intuitively, if the local part of an object is repeatative (occurs at most instances of the object) and occurs at a consistent location with respect to the object center, then the codeword corresponding to such part contributes more votes to the center of the object. Instead of assigning uniform weights for votes from the codewords as in ISM, we assign weights for the votes from the codewords based on their contribution to the object center, *i.e.*, the codewords that contribute more votes to the object center is assigned high votes as compared to others.

The term $p(O|\mathbf{D}_j, l_i)$ in Eq. 4.8 is the probability that the codeword votes for the class O as opposed to some other classes. If we assume this independent of the location of the patch, *i.e.*, $p(O|\mathbf{D}_j, l_i) = p(O|\mathbf{D}_j)$, then $p(O|\mathbf{D}_j)$ can be computed as,

$$p(O|\mathbf{D}_j) = \frac{\frac{|n_O|}{|n_{Otr}|}}{\sum_{c \in \mathbf{C}} \frac{|n_c|}{|n_{ctr}|}} \tag{4.19}$$

where $|n_O|$ is the number of votes to the object O from the codeword $\mathbf{D}_j$, $|n_{Otr}|$ is the total number of features used to learn the class O. This is referred as Naive Bayes weights in [79] as the probability $p(O|\mathbf{D}_j)$ only considers the appearance of the feature and ignores the spatial distribution of the features. It was also shown in [79] that such assumption leads to poor localization performance. Hence it is necessary to jointly consider the appearance and spatial location $p(O|\mathbf{D}, l)$ of the codeword to compute the weights. From Eq. 4.16, we have

$$\begin{aligned} S(O, y|\mathbf{x}_i, l_i) &= \sum_j p(O|\mathbf{D}_j, l_i) p(y|O, \mathbf{D}_j, l_i) h_{ij} \\ &= \sum_j w_j p(y|O, \mathbf{D}_j, l_i) h_{ij} \\ &= \big(\mathbf{w} \odot \mathbf{b}_i(y)\big)^T \mathbf{h}_i \end{aligned} \tag{4.20}$$

where $\mathbf{w} = [w_1, w_2, \cdots, w_k]$ is the global weight vector, $w_j = p(O|\mathbf{D}_j, l_i)$. The Eq. 4.20 can also be written as

$$S(O, y|\mathbf{x}_i, l_i) = \mathbf{w}^T A(y) \tag{4.21}$$

where $A(y) = (\mathbf{h}_i \odot \mathbf{b}_i(y))$ are the activations. Since the activation term $A(y)$ is a function of the location y, we can discriminatively learn the weights by considering the localization information of the training sequences. Eq. 4.20 shows the final expression for the weighted votes for an object O at a location y where $\mathbf{w}$ is global weight and $\mathbf{h}_i$ is the local weight. The overall procedure for obtaining the global weights is summarized in Fig. 4.9.

4.4.3. Discriminative Votemaps

In this section, we discuss the generation of discriminative vote maps for each feature. The local features use the matched codewords to cast weighted votes for the probable Action center. These votes are accumulated in an output space called Hough space. In order to have a good localization, the votes at the spatio-temporal center of the action should be high compared to the other locations (background). Since the groundtruth annotations are available for the training set, we can generate output Hough spaces of the training sequences and use this to compute discriminative weights that maximize the response at the location of the center of action compared to other locations.

Let $S(y)$ denote the probabilistic Hough votes at a location y in the Hough space of the action. Let Y_c denote an area around the center of the action as shown in Fig. 4.8. The objective is to maximize the votes at the centre of the action compared to other locations, *i.e.*, to maximize the sum of votes inside Y_c as compared to sum of the votes at other locations. This can be expressed as

$$\left(\sum_{y \in Y_c} S(y) - \sum_{y \notin Y_c} S(y) \right) \tag{4.22}$$

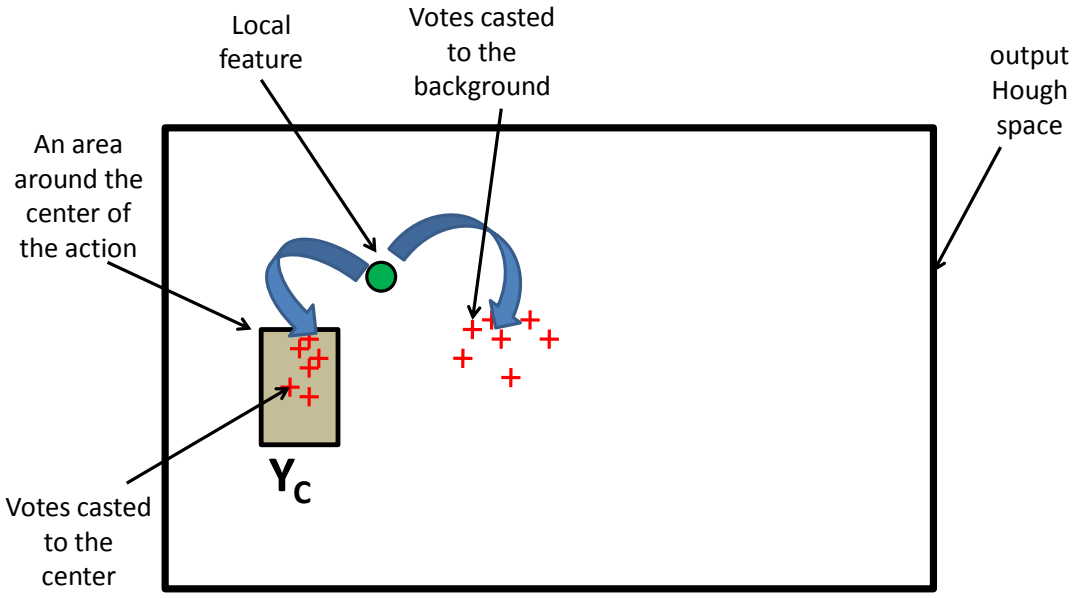


Figure 4.8.: Discriminative votemaps: The Hough space of an image. The gray area represents a bin or an area around the center of the object. The discriminative votemaps are computed as the difference of the sum of votes inside Y_c and sum the votes in the rest of the Hough space excluding the region Y_c.

This is equivalent to minimizing the quantity

$$\begin{aligned} & \Big(\sum_{y \notin Y_c} S(y) - \sum_{y \in Y_c} S(y) \Big) \\ = & \Big(\sum_{y \notin Y_c} \big(\mathbf{w} \odot \mathbf{b}_i(y)\big)^T \mathbf{h}_i - \sum_{y \in Y_c} \big(\mathbf{w} \odot \mathbf{b}_i(y)\big)^T \mathbf{h}_i \Big) \\ = & \ (\mathbf{w} \odot \mathbf{a}_i)^T \mathbf{h}_i \end{aligned} \tag{4.23}$$

where

$$\mathbf{a}_i = \big(\sum_{y \notin Y_c} \mathbf{b}_i(y) - \sum_{y \in Y_c} \mathbf{b}_i(y) \big) \tag{4.24}$$

denote the sum of the Hough votes at the non-object location from i^{th} feature $\mathbf{x}_i$. For each feature in the training set, we accumulate the votes at the Hough space of the sequence to which it belongs and compute discriminative vote maps as described in Eq. 4.24. Note that the vote maps $\mathbf{b}_i(y)$ in Eq. 4.18 computes the un-weighted votes from the matched codewords at any location y of the sequence whereas the discriminative votes maps in Eq. 4.24 computes the difference of un-weighted votes from matched

codewords at the center and non-center locations. Also unlike the vote maps $\mathbf{b}_i(y)$, the discriminative votemaps $\mathbf{a}_i$ is independent of the term y and contains the discriminative information in it. We can also introduce a weighting for the aggregation of votes in Eq. 4.24. For *e.g.* we have used a binned aggregation in Eq. 4.24 which is equivalent to uniform weighting for all the votes in the region Y_c. Instead we can use a soft weighting scheme in which a vote near the center is given high weight and the weight decays as the votes are farther from center.

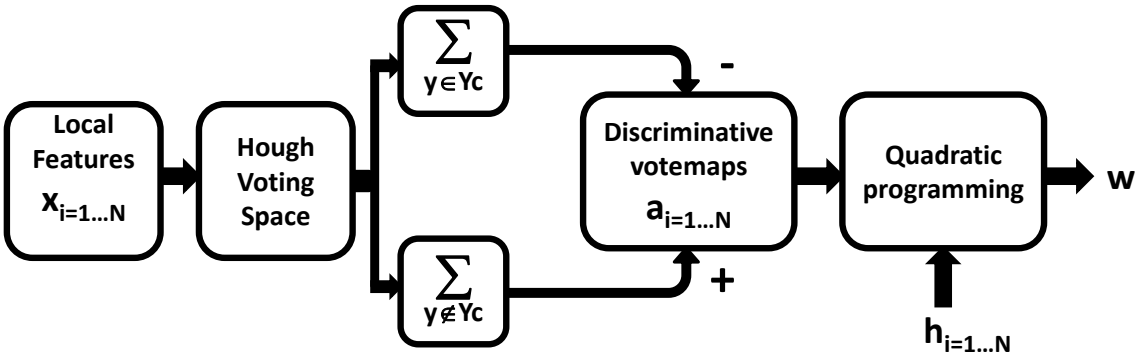


Figure 4.9.: Computation of global weights: A local feature is compared with the codebook to select the nearest codewords and the local weights for theses codewords are computed using LLC. The feature then votes for the Hough space of the sequence to which it belongs and discriminative votemaps are obtained by computing the difference of sum of votes at the center and non-center locations. The discriminative votesmaps along with the local weights are used by a quadratic programming module to compute the global weights $\mathbf{w}$.

4.5. Objective Function

In this section, we propose to incorporate Hough voting scheme into the LLC framework. The LLC described in Section 4.4.1 considers only the appearance of the feature in the objective function Eq. 4.12. We add discriminative information derived from the localization information of the training sequences into the objective function. The combined objective function is given by

$$f(\mathbf{h}_1, \cdots, \mathbf{h}_n, \mathbf{w}) = \lambda \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}\mathbf{h}_i\|^2 + \eta \|\mathbf{g}_i \odot \mathbf{h}_i\|^2 + \sum_{i=1}^{n} (\mathbf{w} \odot \mathbf{a}_i)^T \mathbf{h}_i + \gamma \mathbf{w}^T \mathbf{w} \quad (4.25)$$

$$\text{s.t} \quad \mathbf{1}^T \mathbf{w} \leq \mu, \quad \mathbf{c}^T \mathbf{h}_i = 1, \quad \mathbf{h}_i \geqslant 0, \quad \mathbf{w} \geqslant 0, \quad \forall i$$

where $\mathbf{h}_i \in \mathbb{R}^k$ are the coefficients that indicate the local weight of the training samples. The term $\|\mathbf{x}_i - \mathbf{D}\mathbf{h}_i\|^2$ and the term $\|\mathbf{g}_i \odot \mathbf{h}_i\|^2$ corresponds to the reconstruction error and the locality constraint as described in Eq. 4.12 and the term $(\mathbf{w} \odot \mathbf{a}_i)^T\mathbf{h}_i$ is a linear function of the probabilistic Hough votes. The discriminative votemaps $\mathbf{a}_i$ is computed from the Hough voting spaces of the training sequence as described in Eq. 4.23 and Eq. 4.24. The last term is a regularization term used to compute weights. Although l_1 norm is popularly used to impose sparseness constraints, we obtain sparse weight vector using l_2 norm similar to the approaches in [21,79] that have an objective function similar to SVM [19]. The LLC objective function in Eq. 4.12 imposes the constraint $\mathbf{1}^T\mathbf{h}_i = 1$ which forces all k codewords to contribute for the reconstruction of $\mathbf{x}_i$ whereas the objective function in Eq. 4.25 uses only those nearest codewords that are within a certain distance from the feature to reconstruct $\mathbf{x}_i$. The value of $\mathbf{c}$ is given by Eq. 4.15. The constraints $\mathbf{c}^T\mathbf{h}_i = 1$ and $\mathbf{h}_i \geqslant 0$ ensures that $\mathbf{c} \odot \mathbf{h}_i$ is a valid probability distribution, $\mathbf{w} \geqslant 0$ imposes positivity constraints on the weights. In addition we also add a constraint on the global weights $\mathbf{1}^T\mathbf{w} \leq \mu$ to handle scaling. Solving for $\mathbf{h}_i$ in Eq. 4.25 yields a sparse coefficient vector with significant weights for the local codewords. Hence similar to the approach in [118], we select q nearest codewords for the data vector $\mathbf{x}_i$ and solve a smaller system of equations. Then objective function can be rewritten as

$$\begin{aligned} f(\mathbf{h}_1, \cdots, \mathbf{h}_n, \mathbf{w}) = \quad & \lambda \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}^i\mathbf{h}_i\|^2 + \sum_{i=1}^{n} (\mathbf{w} \odot \mathbf{a}_i)^T\mathbf{h}_i + \gamma \mathbf{w}^T\mathbf{w} \\ & \text{s.t} \quad \mathbf{1}^T\mathbf{w} \leq \mu, \quad \mathbf{c}^T\mathbf{h}_i = 1, \quad \mathbf{h}_i \geqslant 0, \quad \mathbf{w} \geqslant 0, \quad \forall i \end{aligned} \tag{4.26}$$

where $\mathbf{D}^i \in \mathbb{R}^{m \times q}$ is a subset of the whole dictionary $\mathbf{D}$ containing q nearest neighbors to the data vector $\mathbf{x}_i$ and $\mathbf{h}_i \in \mathbb{R}^q$. This modification also results in significant reduction in the computational complexity since $q \ll k$. The constants λ and γ are the weights for the reconstruction error term and the regularization term. The values these constants can be chosen using the cross-validation. Note that the cost function in Eq. 4.26 is non-convex w.r.to the variables $\mathbf{w}$ and $\mathbf{h}_i$ and hence cannot be solved together. In order to optimize the cost function in Eq. 4.26, we follow an alternating optimization procedure. More precisely, we solve for one of the unknown parameter $\mathbf{w}$ and $\mathbf{h}_i$ by keeping the other parameter fixed. The training procedure of the proposed method is as shown in Fig. 4.10.

4.5.1. Solving for $\mathbf{h}_i$ and $\mathbf{w}$

We first solve for the coefficients $\mathbf{h}_i$ by keeping the weights $\mathbf{w}$ fixed. The cost function in Eq. 4.26 reduces to

$$\widehat{\mathbf{h}}_i = \arg\min_{\mathbf{h}_i} \quad \lambda\|\mathbf{x}_i - \mathbf{D}^i\mathbf{h}_i\|^2 + (\mathbf{w} \odot \mathbf{a}_i)^T\mathbf{h}_i \qquad (4.27)$$
$$\text{s.t} \quad \mathbf{c}^T\mathbf{h}_i = 1, \quad \mathbf{h}_i \geqslant 0, \quad \forall i$$

Since the global weights $\mathbf{w}$ are fixed, the third term vanishes. As the local features are independent of each other, we solve for each coefficients $\mathbf{h}_i$ separately with the constraints that the coefficients should be positive and should sum to one. These constraints are imposed to ensure that the coefficients $\mathbf{h}_i$ is equivalent to $p(\mathbf{D}_j|\mathbf{x}_i)$. The global weights $\mathbf{w}$ and the discriminative votemaps $\mathbf{a}_i$ can be assumed to be independent of the coefficients $\mathbf{h}_i$. Thus Eq. 4.27 is quadratic with respect to $\mathbf{h}_i$ and hence can be solved using a conventional quadratic solvers [123]. After obtaining the local weights $\mathbf{h}_i$ from Eq. 4.27,

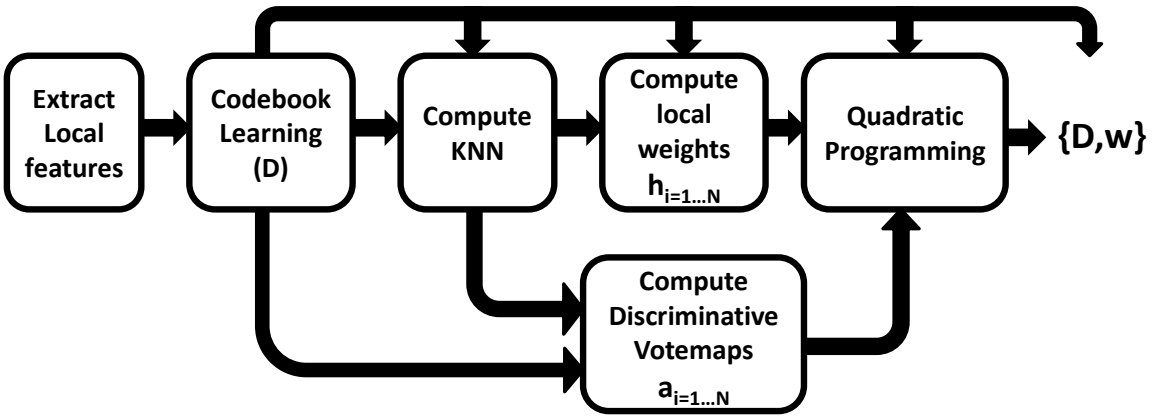


Figure 4.10.: Pipeline for training: During training, a codebook and the weights for the votes from the codeword are learned. Initially a codebook is learned from the local features using an unsupervised clustering algorithm. This codebook is used to compute local weights for each feature. The feature then votes for the Hough space of the sequence using the local codewords and this Hough space is used to compute discriminative votemaps for each feature. Finally the local weights and the discriminative votemaps are employed to compute global weights.

we compute the discriminative votemaps for the training features by carrying out the voting procedure and aggregating the votes from the Hough spaces of the training sequences. we then solve for the global weights $\mathbf{w}$ by keeping $\mathbf{h}_i$ fixed. The cost function

in Eq. 4.26 reduces to

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (\mathbf{h}_i \odot \mathbf{a}_i)^T \mathbf{w} + \gamma \mathbf{w}^T \mathbf{w} \qquad (4.28)$$
$$\text{s.t} \quad \mathbf{1}^T \mathbf{w} \leq \mu, \quad \mathbf{w} \geqslant 0$$

The Eq. 4.28 is quadratic with respect $\mathbf{w}$ and a conventional quadratic solvers [123] can be used to compute $\mathbf{w}$.

4.5.2. Action Localization

The aim of this work is to spatio-temporally localize instances of actions in video sequences, *i.e.*, to predict the location of the object and also the start and end of the action. The Fig. 4.11 shows the overview of the system. To achieve this, we follow the spatio-

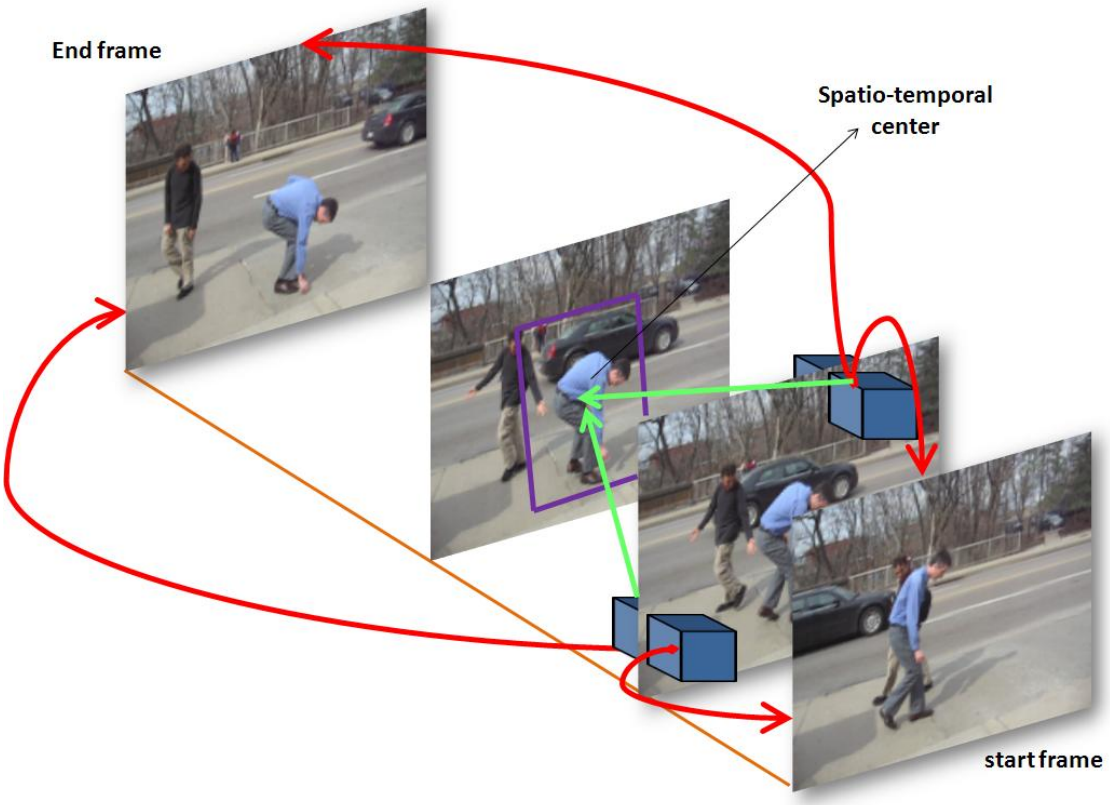


Figure 4.11.: Overview of the system: The spatio-temporal descriptors extracted from the video sequence use the codebook to vote for the spatio-temporal center (green arrows) and start and end (red arrows) of the action

temporal Hough voting framework similar to [87]. During training, we detect salient points in video sequences using spatio-temporal interest point detectors and extract descriptors at these interest point locations. In order to capture the actions at different sizes and speed, descriptors are extracted at different spatial and temporal scales at the interest point locations. A codebook is created from these descriptors using k-means algorithm. The codewords are then compared against each of the training descriptors. If the codeword is matched to a descriptor, then the codeword records the information about the descriptors such as the relative location and scale. Let (l, f) denote the spatial location and frame number (temporal location) at which a descriptor was detected, σ, τ denote the detected spatial and temporal scales respectively. If a feature is matched to a codeword, then the codeword records the relative spatial location $l_m = l - l_c$, relative temporal locations, $f_{sm} = f - f_s$, $f_{em} = f - f_e$ and the spatial, temporal scales σ_m, τ_m respectively where l_c is the spatial centre of the object and f_s, f_e denote the start and end frame of the action respectively.

Algorithm 4: Algorithm for the proposed framework

input : $\mathbf{X}$, λ, l, γ
output: $\mathbf{D}, \mathbf{w}$
begin
  $S1$: Compute Codebook ($\mathbf{D}$) using k-means algorithm
  $S2$: Compare the codewords against each feature and store the relative location of the feature with respect to spatio-temporal centre
  $S3$: Compute $\mathbf{a}_u$ for all the training features using Eq. 4.29
  $S4$: Compute local weights $\mathbf{h}_i$ for all the training features using Eq. 4.27
  $S5$: Compute the global weights $\mathbf{w}$ using Eq. 4.28
end

The codebook is used to compute the discriminative votemaps for a feature in the training set using Eq. 4.24 , *i.e.*, the discriminative votemaps are computed by accumulating the votes casted by the feature in the Hough space and computing the difference of votes at the foreground and background. Since we are interested in predicting the spatial center, start and end (frame) of the action, we need three Hough spaces, *i.e.*, a 2D Hough space to accumulate the votes for spatial center, two 1D Hough spaces to accumulate the votes for start frame and end frame. In order to give equal weights to the Hough spaces, we compute the discriminative votemaps for each of the Hough spaces and add them together. Let l_c, f_s, f_e denote the spatial centre which corresponds to the center of the human (or bounding box containing the human), start and end frame of a sequence in the training set respectively. let L_c, F_s, F_e denote bins around the centres of

the Hough spaces. For u^{th} feature from the sequence, we have,

$$\mathbf{a}_u^c = \left(\sum_{y \notin L_c} \mathbf{b}_u(y) - \sum_{y \in L_c} \mathbf{b}_u(y) \right)$$

$$\mathbf{a}_u^s = \left(\sum_{y \notin F_s} \mathbf{b}_u(y) - \sum_{y \in F_s} \mathbf{b}_u(y) \right)$$

$$\mathbf{a}_u^e = \left(\sum_{y \notin L_e} \mathbf{b}_u(y) - \sum_{y \in L_e} \mathbf{b}_u(y) \right)$$

and the discriminative votemap for the u^{th} feature, $\mathbf{a}_u$ is computed as

$$\mathbf{a}_u = \mathbf{a}_u^c + \mathbf{a}_u^s + \mathbf{a}_u^e \tag{4.29}$$

During training, a feature detected at a location (l_t, f_t) votes to the spatial location $l = l_t - \frac{\sigma_t}{\sigma_m} l_m$ and the start frame $f_{st} = f_t - \frac{\tau_t}{\tau_m} f_{sm}$, end frame $f_{et} = f_t - \frac{\tau_t}{\tau_m} f_{em}$ where σ_t, τ_t denote the spatial and temporal scales of the detected descriptor. Then the local weights $\mathbf{h}_i$ for each of the training features are computed using Eq. 4.27. The computed local weights are then used to compute the global weights $\mathbf{w}$ using Eq. 4.28. The training algorithm is summarized Algorithm 4.

During testing, a test feature $\mathbf{x}_t$ is compared against the codebook and q nearest codewords are selected. The local weights for these codewords are computed by solving for $\mathbf{h}_t$,

$$\begin{aligned} \widehat{\mathbf{h}}_t = \arg\min_{\mathbf{h}_t} \quad & \|\mathbf{x}_t - \mathbf{D}^t \mathbf{h}_t\|^2 \\ \text{s.t} \quad & \mathbf{c}^T \mathbf{h}_t = 1, \quad \mathbf{h}_t \geqslant 0 \end{aligned} \tag{4.30}$$

where $\mathbf{D}^t \in \mathbb{R}^{m \times q}$ contains q nearest codewords of $\mathbf{x}_t$. Only those codewords for which $d(\mathbf{D}^t, \mathbf{x}_t) < \kappa$ cast weighted votes (weighted by the global weights) to the spatio-temporal location of the action in the Hough space and also to the start and end frame of the action. The maxima of the Hough space is computed using Meanshift algorithm [24]. The overall procedure for testing is shown in Fig. 4.12.

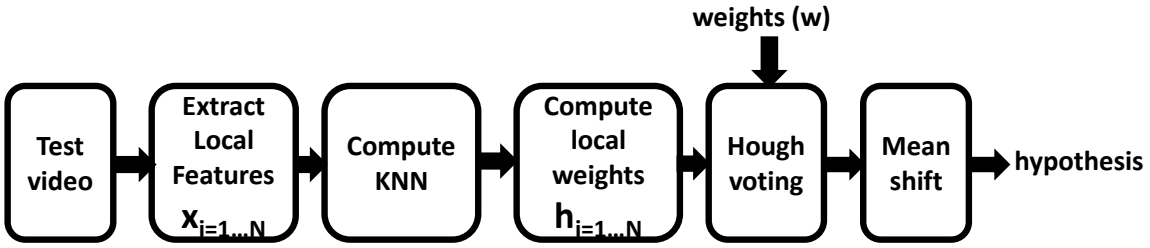
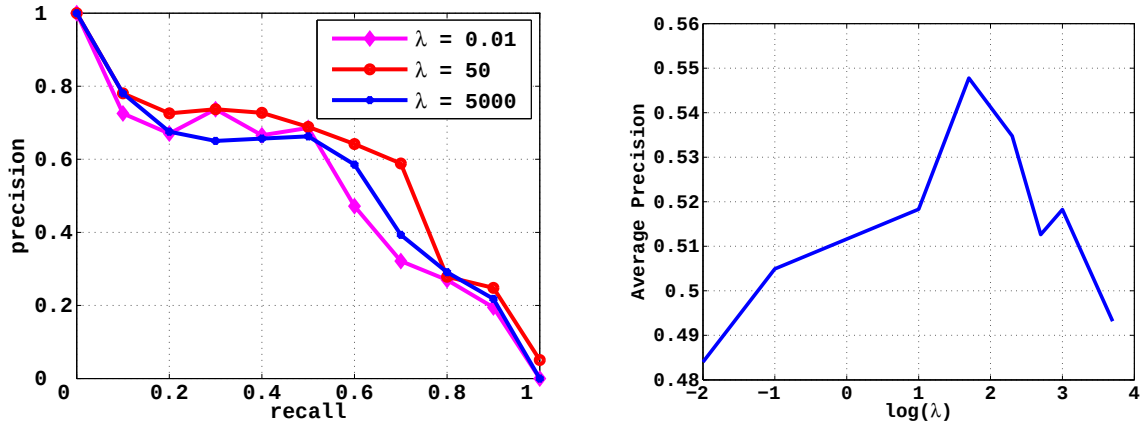


Figure 4.12.: Pipeline for testing: The local features extracted using interest point detectors are compared with the codebook. The nearest matching codewords and the corresponding local weights are computed for each feature. The nearest codewords are used by the feature to cast votes that are weighted by the local weights. These probabilistic votes are further weighted by the global weights computed during training. Finally a mean-shift algorithm is employed to detect the peaks in the Hough space which corresponds to the hypothesis

4.6. Experimental Results

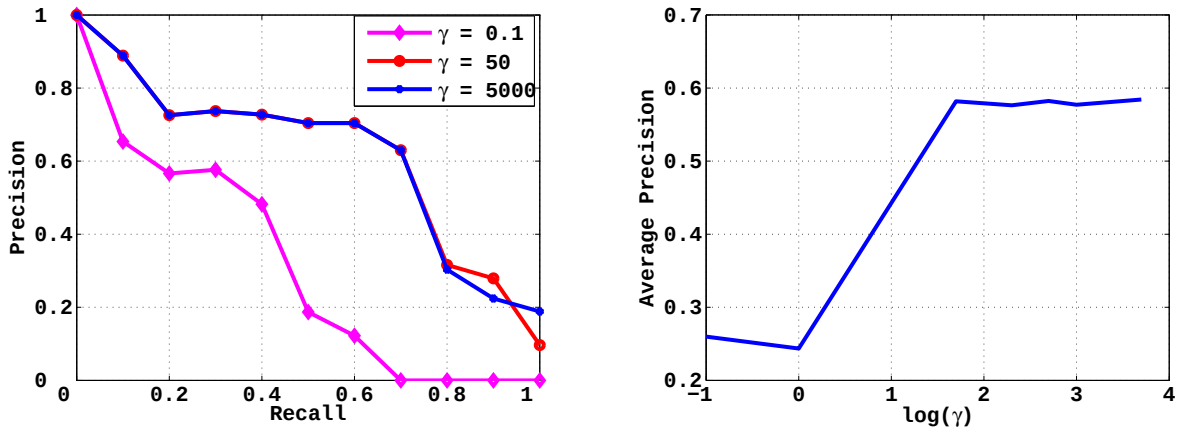
In this section, we demonstrate the performance of the proposed dictionary weighting approach on a subset of KTH action dataset [101] and CMU action dataset [49]. The term action refers to a simple dynamic pattern by a human over a short duration of time. For our experiments, we consider a single repetition of an activity as an action instance *.e.g.* pickup, handwave. In this work, we only consider still actions for the evaluation of the algorithm.

We set the value of nearest neighbors, $l = 20$ for our experiments. The weighting for the reconstruction error λ and the regularization term γ are chosen in the range of $10 \leq \lambda \leq 300$ and $10 \leq \gamma \leq 300$ respectively for our experiments. During training, we generate a dictionary for each action separately by using the features extracted from the training sequences. The number of codewords (size of the dictionary) is chosen empirically according to the rule $\approx n/4$ where n is the number of features used to train the dictionary. We can also use cross-validation to compute the size of the dictionary. During testing, the test video is matched against all the dictionaries and the output Hough spaces are generated for all the categories. Subsequently the category of the hypothesis is chosen as the the Hough space containing maximum Hough response. For validation, a detection is considered correct if the detected action label is same as the ground truth action label and the volumetric overlap is greater than 50%, *i.e.*, the intersection to union ratio of the hypothesis and the ground truth is greater than 0.5, $\frac{(V \cap G)}{(V \cup G)} > 0.5$ where V is the hypothesis and G is the groundtruth.



(a) (b)

Figure 4.13.: Effect of λ on the detection performance (a) the precision recall curves for various values of λ. A good performance is observed for $\lambda = 50$. (b) The average precision values for various values of λ. The detector performance degrades for very high and low values of λ.



(a) (b)

Figure 4.14.: Effect of the parameter γ on the detection performance (a) the precision recall curves for various values of γ. (b) The average precision values vs γ. The detector performance degrades for low values of γ.

4.6.1. Effect of the Parameters λ and γ

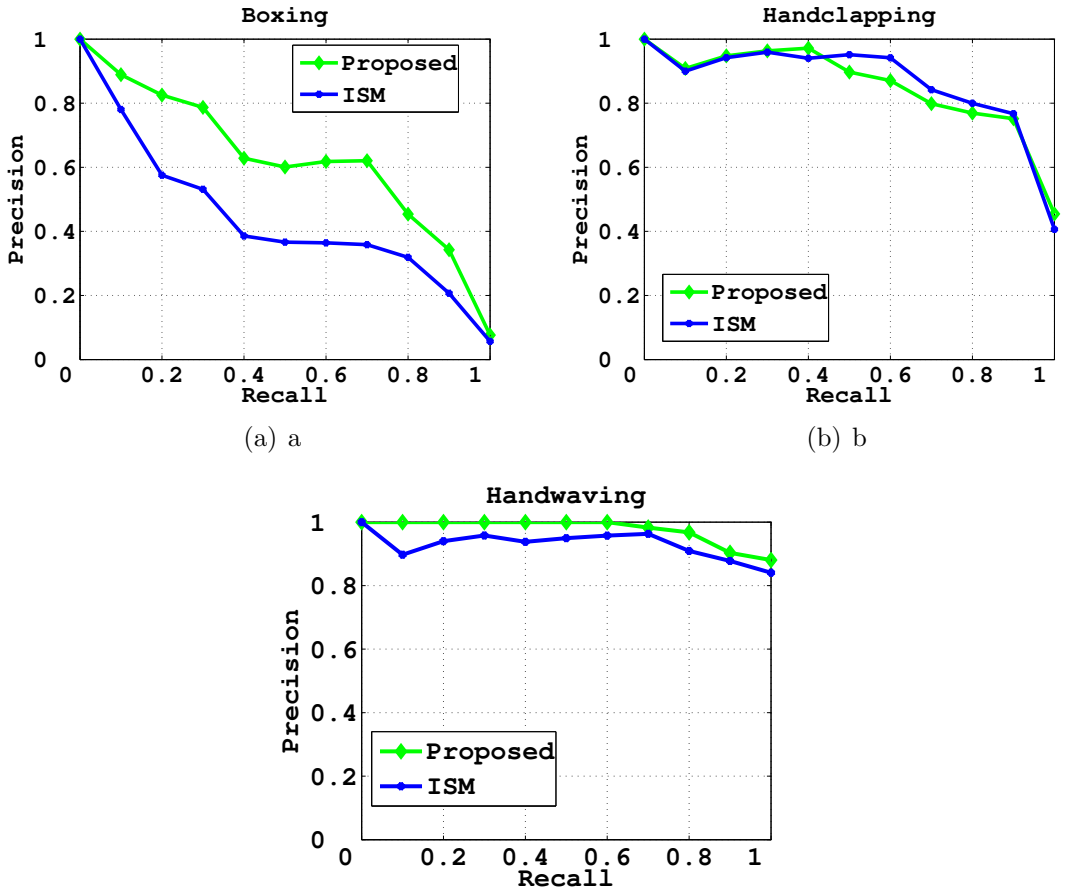
The purpose of the experiment is to demonstrate the effect of the parameters λ and γ on the detection result. We employ the videos from the Boxing category of KTH dataset for this experiment. We divide the videos into three non-overlapping sets and use two sets for training and the one set for testing. We repeat the test three times

such that each video set is tested once and report the average performance. We first demonstrate the effect of the parameter λ on the performance of the algorithm. It can be observed from Eq. 4.26 that the parameter λ acts as a weight to the reconstruction error term. Fig. 4.13.a shows the precision-recall curves for three different values of λ. From the figure we observe that the detector performs well for $\lambda = 50$. Fig. 4.13.b shows the average precision values for various values of λ. For low values of λ the weight on the reconstruction error term is low. This leads to non-optimal reconstruction of the input data and hence degrades the performance of the detector. For high values of λ, the first term in Eq. 4.26 becomes dominant and the discriminative term (second term) computed from the localization information of the training images does not contribute to the objective function. Thus the performance of the detector is similar to the ISM which uses a generative model for the detection task. From the graph, it is also evident that the values of λ in the range $10 \leq \lambda \leq 300$ are optimal for detection.

We next demonstrate the influence of the parameter γ on the detection performance. The parameter γ is the weighting for the term $\mathbf{w}^T\mathbf{w}$ in Eq. 4.26. It can be seen from Fig. 4.14.a that the detector gives a similar performance for high values of γ but poor performance for low value ($\gamma = 0.01$). Fig. 4.14 shows the graph for average precision vs γ. The significant decrease in the detector performance at low values of γ is due to the constraint $\mathbf{1}^T\mathbf{w} \leq \mu$ because the low values of γ causes $\mathbf{w}$ to take higher values in Eq. 4.28 but the above constraint restricts the values of $\mathbf{w}$. For higher values γ, the $\mathbf{w}$ scales by a constant factor and hence the performance of the detector remains same.

4.6.2. Comparison to the Baseline and State-of-the-art

KTH dataset: In this section, we compare our method with the baseline line method, ISM [60] on the KTH dataset. We use a subset of the KTH dataset (actions where the human subject is stationary) for the evaluation. The KTH subset for this experiment consists of 25 subjects in four scenarios performing three actions (box, handclap, handwave). Each video consists of a single action. Among 25 videos, we use 16 for training and remaining 9 for testing. This procedure is repeated three times such that all the videos are tested atleast once and we report the averaged values. For training, we use a subset of the action instances from each class of the training video sequences and extract HOG and HOF features [54] (dimension=162) at the interest point detected using Harris3D detector [54]. We use k-means algorithm to generate the dictionary. The precision and recall are computed using the relation $Precision = TP/(TP + FP)$,



(a) a

(b) b

(c) c

Figure 4.15.: Precision-Recall curves for three actions of the KTH dataset: a) Boxing, b) Handclapping and c) Handwaving. The blue curve shows the recall for the baseline method [60] and the green curve shows the performance of the proposed algorithm.

$Recall = TP/nT$ where TP is the the number of true positives, FP is the number of false positives and nT is the total number of positive instances in the test dataset [27]. Please note that FP also includes the undetected actions. The groundtruth data was obtained using hand annotated bounding boxes for selected frames of the video and the annotations for the intermediate frames were obtained using linear interpolation. We compare our results with the Implicit Shape Model [60] that uses the codebook constructed using the k-means algorithm.

Fig. 4.15 shows the Precision-Recall curves for boxing, handclapping and handwaving actions of the KTH dataset. The methods are trained on the training part of the dataset and validated on the testing part of the each fold. Fig. 4.15(a) shows the precision-recall curves for the Boxing action. It is evident from the figure that our

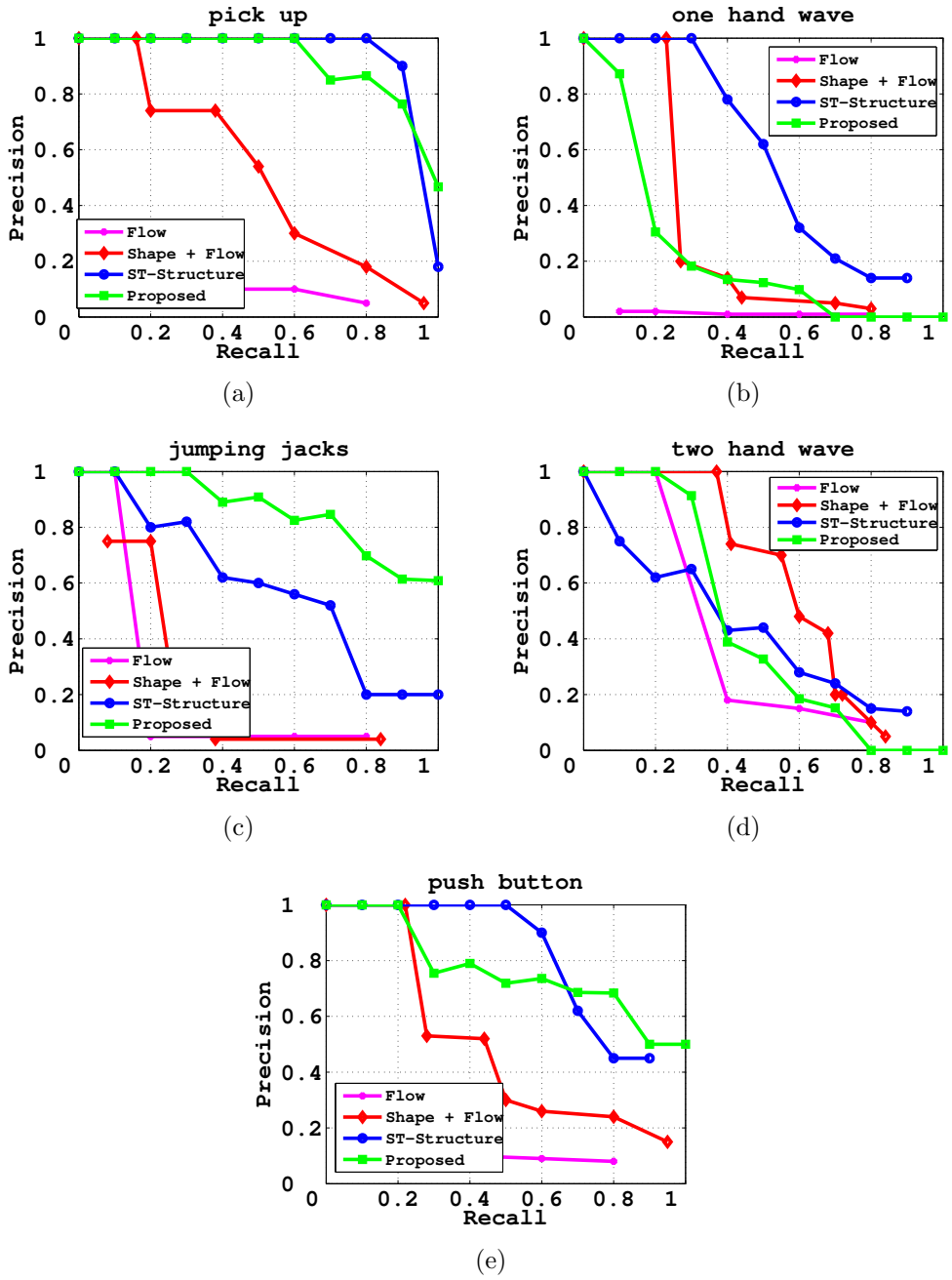


Figure 4.16.: Precision-Recall curves for five actions of the CMU dataset: a) pickup, b) one hand wave, c) jumping jacks d) two hand wave and e) push button. The magenta, red and blue curves correspond to the algorithms in [15], [49] and [27] (as published in [49] and [27]). The green curve shows the performance of the proposed approach

method significantly outperform the baseline method at all precision rates. For action handclapping Fig. 4.15(b), the performance of algorithm is similar to the baseline method and for handwaving Fig. 4.15(c), the our method marginally outperforms the the ISM.

Table 4.1.: Average Precision values for Three actions of KTH dataset

Action/Method	ISM [60]	Proposed
Boxing	0.40	**0.59**
Handclap	**0.85**	0.84
Handwave	0.94	**0.97**
Avg	0.73	**0.80**

The average precision for the actions are show in Table 4.1. From the table, we see that our method (59%) significantly outperform the baseline algorithm (40%) for the action boxing. For handclapping, the average precision for the proposed method (84%) is slightly lower than the baseline (85%) and for handwaving, the proposed method obtains 3% improvement in the average precision compared to [60]. Overall, the mean average precision values for our method is 7% better than the baseline method.

CMU dataset: We now compare the performance of the proposed method with the state-of-the-art methods [49], [15], [27] on the CMU dataset [49]. We divide each action category into two non-overlapping sets, train the model on one of the sets and test the model on the other. This is repeated such that all videos are tested once. For training, we construct a k-means codebook by the features extracted from the training set using the HOG and HOF features.

Table 4.2.: Average Precision values for CMU dataset

Action/Method	[15]	[49]	[27]	Proposed
pick up	0.09	0.46	**0.9**	**0.89**
one hand wave	0.01	0.28	**0.52**	0.17
jumping jacks	0.14	0.21	0.55	**0.84**
two hand wave	0.29	**0.57**	0.37	0.40
push button	0.1	0.45	**0.74**	**0.74**
Average	0.13	0.4	**0.61**	0.60

Fig. 4.16 shows the precision-recall curves for five actions of the CMU dataset. The green curve shows the recall for our method whereas the magenta, red and blue curves show the recall for [15], [49] and [27] respectively. For the actions pick up [Fig. 4.16(a)] and push button [Fig. 4.16(e)] our method is similar to the state-of-the-art at high precision rates. Our method outperforms all the others algorithms for the action jumping

jacks [Fig. 4.16(c)]. The reduction in the performance of the proposed method for the action two hand wave [Fig. 4.16(d)] is due to the confusion with the action jumping jacks. We also see a drop in the performance of the action one hand wave [Fig. 4.16(b)] as the action contains significantly less number of features at the location of the action as compared to the background. Table 4.2 compares the average precision values for the proposed method with the other algorithms. From the values it is evident that the proposed method has a similar average precision values ($\approx$ 90% and 74%) for the actions pick up and push button and also we can note that the proposed method has significantly higher average precision (84%) compared to the state-of-the-art (55%). Overall the proposed method (60%) performs better than the algorithms in [15] (13%) and [49] (40%) and on par with the algorithm in [27] (61%).

4.7. Conclusions

In this chapter, we presented an action detection algorithm based on a codebook generated using k-means algorithm. In order to overcome the limitations of the generative model of the ISM, we introduced a discriminative approach for the voting in the ISM where the discriminate information is obtained by the localization information of the training sequences, *i.e.*, we proposed a weighting approach which assigns weights for the votes generated by the codebook. We proposed two kinds of weighting for the codebooks. The local weights quantify the degree of matching between the codeword and the feature. In order to compute the local weights, we employed LLC where the codes obtained using LLC are used as the local weights. Further, we proposed a discriminative global weighting scheme for the codebook that maximize the response at the spatio-temporal location of the activity compared to background of the training set. The proposed method is tested on a subset of the KTH dataset where it is shown that the proposed method significantly outperforms the codebook generated using k-means algorithm. Also the experiments on the CMU dataset shows that our algorithm performs on par with the state-of-the-art.

Chapter 5.

Supervised Dictionary Learning for Action Localization

5.1. Introduction

In the previous Chapter 4, we described the ISM [60] that consists of a probabilistic generative model where the parts of the object provide evidence about the center and scale of the object. However, it is reported in [37] that the powerful generative capabilities of the ISM results in high false positives. To cope with the false positives, a few methods have been proposed where a discriminative classifier is used along with the generative model [37, 51, 79]. In the previous chapter 4, we proposed a discriminative weighting for the codebook that maximizes votes at the center of the object compared to the background. However, regardless of these efforts the codebook learning stage is unsupervised in all of the above mentioned methods. In this chapter, we learn a task-dependent dictionary that is adapted for the task of action localization.

The generative models have widely been used in the computer vision problems due to its advantages such as robustness to partial occlusions, variations in viewing conditions, intra-class variations etc. ISM is one such generative model that can provide a greater flexibility in representing a target category. ISM employs a codebook (or dictionary) to map a descriptor from the continuous descriptor space to a set of representative codewords which are in-turn mapped to a set of votes in the output Hough space. Since the local descriptors independently vote for the center of the object, the model can interpolate between the local parts seen on different training images and hence it can achieve a good detection results on the target category. However, this greater flexibility

comes at the price of high false positive rates. This is due to the unsupervised approach employed in learning the codebook for ISM. In order to overcome these drawbacks, discriminative models are used in conjunction with the generative models where the generative model generates the hypothesis which are verified by their discriminative counterpart. But recently there has been an increasing interest in developing algorithms that combine the advantages of the generative and discriminative models, *i.e.*, to learn a task dependent dictionary. In this chapter, in contrast to the ISM [60,61] which learns the dictionary in an unsupervised manner by considering only the appearance of the local descriptors, we incorporate the localization information into the learning of the dictionary.

Recently many methods have been proposed [17,45,55,73,75,76,92,126,135] to learn the in a supervised manner. The methods in [17,45,76] couple the the dictionary learning and the classifier stage by incorporating the classifier information into the dictionary learning stage. The codes in all these methods were obtained using sparse coding algorithm. But the promising results in [118] where codes are learned from LLC by imposing the locality constraint (as opposed to sparsity constraint) suggest that locality is more important than sparsity. Inspired by this, we employ LLC to learn a supervised dictionary for action detection. We employ the ISM in which the descriptors extracted at the interest points of the video cast probabilistic votes to the spatial location and the temporal extent of the action using an unsupervised codebook. While this unsupervised codebook might be statistically adapted to the data but it may not be optimal for localization. Since the ground truth localization information is available for the training set, we can use this information to learn a codebook optimized for localization. We compute the Hough voting maps for the training sequences and incorporate this information to update the dictionary. This results in a discriminative dictionary that maximizes the Hough voting response at the spatio-temporal location of the activity as compared to the background. Summarizing, the main contributions of this chapter:

- We develop a framework that enables supervised learning of the dictionary for ISM, *i.e.*, the proposed algorithm computes the localization information from the output space of the training sequences and this discriminative information is incorporated into dictionary learning. This is in contrast to the ISM [60] that only uses the appearance information of the training descriptors to learn the dictionary. To the best of our knowledge, this is the first attempt to use the output Hough space information for dictionary learning.

- We employ the above framework to learn a task dependent dictionary optimized for the localization task. More precisely, we use the localization information of the training sequences to learn a discriminative dictionary that maximize the response at the spatio-temporal center of the activity compared to background.
- We also extend the above approach to include the background information into the dictionary learning. This results in a dictionary that can discriminate the descriptors extracted at the foreground and background.

The rest of the chapter is organized as follows. In Section 5.2, we briefly describe the Hough voting procedure. In Section 5.3, we explain the procedure for obtaining the local, global weights for the dictionary. Section 5.4.2 describes the dictionary updation process and we extend this model to incorporate the background into dictionary training in Section 5.5. We discuss the experimental results in Section 5.6 and finally we conclude in Section 5.7.

5.2. Hough Voting

As described in Section 4.2 of chapter 4 the Implicit Shape Model (ISM) proposed by Leibe *et al*. [61] learns a set of codewords from the training descriptors. These codewords map the input descriptors to a set of probabilistic votes on the Hough image. During testing, the test patches are extracted at the interest point locations and compared against all the codewords. The matched codewords cast probabilistic votes that are collected in a Hough space. The mode of the Hough space corresponds to the location of the hypothesis. Let $\mathbf{x}_i$ denote a descriptor extracted at the location l_i of the input image and let $\mathbf{D}_j$ denote the j^{th} codeword. A feature $\mathbf{x}_i$ is said to be matched to a codeword $\mathbf{D}_j$ if the distance between the feature and the codeword is less than a pre-defined threshold κ, *i.e.*, $dist(\mathbf{D}_j, \mathbf{x}_i) < \kappa$. Let $S(O, y|\mathbf{x}_i, l_i)$ denote the Hough score for the object O collected at location y from the feature $(\mathbf{x}_i, l_i)$, *i.e.*,

$$S(O, y|\mathbf{x}_i, l_i) = \sum_j p(O, y|\mathbf{D}_j, l_i) p(\mathbf{D}_j|\mathbf{x}_i) \tag{5.1}$$

$$= \sum_j p(y|O, \mathbf{D}_j, l_i) p(O|\mathbf{D}_j, l_i) p(\mathbf{D}_j|\mathbf{x}_i) \tag{5.2}$$

The first term in Eq. 5.2 indicates the probabilistic Hough vote for the object location y from the codeword $\mathbf{D}_j$ and the feature $\mathbf{x}_i$. The second term specifies the confidence

that the codeword $\mathbf{D}_j$ at location l_i is matched on the object O as opposed to other objects. And the third term indicates the quality of the match between the codeword and the patch. We proposed a dictionary weighting approach in Chapter 4 to learn the third term using LLC and second term by considering the Hough voting space of the training images. In this work, we present a method that employs the localization information from the Hough voting space of the training sequences to learn a supervised dictionary. In the first part of this chapter, we build a task-dependent dictionary by only considering the descriptors extracted at the object location (foreground) and ignore the descriptors extracted at the background, *i.e.*, we update the first term in Eq. 5.1 to include the discriminative information obtained by the localization of the training sequences. In second part of this chapter we incorporate the class information of the training descriptors into the model which allows us to discriminate the foreground from background and hence we update the first and the second term in Eq. 5.2.

5.3. Discriminative Voting for Localization

Generally the algorithms that use the implicit shape model [51, 61, 79, 137] use either a codebook learned using an unsupervised algorithm or by incorporating Hough voting scheme into the random forest framework [127]. Though the Hough forest algorithm [127] incorporates the voting information for codebook generation, it does not consider the resulting Hough voting space of the training sequences (at the sequence level) during training. In this work we propose to learn an adaptive codebook for action localization that maximizes the response of the Hough votes at the spatio-temporal location of the activity compared to background. , *i.e.*, we compute the contribution of votes to the spatio-temporal location of the activity and background for each local descriptor of the training sequences and use this information to update the dictionary as illustrated in Fig. 5.1(b). The ISM compares the descriptor with all the codewords and all matched codewords cast votes but in this work, similar to Chapter 4, we impose locality constraint on the codewords , *i.e.*, we compare the descriptor with only q nearest neighbors for matching. We employ LLC [51, 118] to compute local weights (LLC codes) for the matched codewords as opposed to [60,79,137] where uniform weights are assigned to the matched codewords. This avoids the error that could occur due to hard quantization of the codewords [17, 40]. In the following section, we briefly summarize the computation of the local weights and the generation of discriminative votemaps discussed in chapter 4.

5.3.1. Local Weights

The probabilistic Hough vote for an object O at a location y casted by a feature $\mathbf{x}_i$ is given by Eq. 5.2. The term $p(\mathbf{D}_j|\mathbf{x}_i)$ indicates the degree of match between the codeword $\mathbf{D}_j$ and the feature $\mathbf{x}_i$. It can also be considered as the local weight for the votes from the codeword $\mathbf{D}_j$. Instead of assuming a uniform distribution [60] for $p(\mathbf{D_j}|\mathbf{x}_i)$, we proposed LLC to compute $p(\mathbf{D}_j|\mathbf{x}_i)$. Let $\mathbf{h}_i \in \mathbb{R}^k$ denote the weight vector for a feature $\mathbf{x}_i$ where $\mathbf{h}_i$ is computed using LLC. Further adding the constraints [51] on the weight vector , $\mathbf{c}^T\mathbf{h}_i = 1$ and $\mathbf{h}_i \geqslant 0$ ensures that $\mathbf{c} \odot \mathbf{h}_i$ is a valid probability distribution. Thus, the probabilistic Hough vote for an object O at a location y casted by a feature $\mathbf{x}_i$ can be written as

$$S(O, y|\mathbf{x}_i, l_i) = \sum_{j=1}^{k} p(O, y|\mathbf{D}_j, l_i)h_{ij} = \mathbf{b}_i^T(y)\mathbf{h}_i \tag{5.3}$$

where $\mathbf{b}_i(y) \in \mathbb{R}^k$ is a vector with entries

$$\mathbf{b}_i^j(y) = \begin{cases} \frac{1}{|\mathbf{D}_j|} & \text{if } dist(\mathbf{D}_j, \mathbf{x}_i) < \kappa \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

here $\mathbf{b}_i(y)$ is the vote map for the i^{th} feature and $|\mathbf{D}_j|$ is the number of features that are matched to the j^{th} codeword during training.

5.3.2. Discriminative Vote Maps

In this section, we discuss the generation of discriminative vote maps [51] for each feature. In order to have a good localization, the hough space response at the spatio-temporal center of the action should be high compared to the other locations (background). Since the location of the center, spatio-temporal extent of the actions are available during training, we can incorporate these information into the learning of the dictionary. This enables us to adaptively learn a dictionary that maximize the response at the center of the action compared to background. Let $S(y)$ denote the probabilistic Hough votes at a location y of the Hough space of a sequence. Let Y_c denote a small area around the center location of the sequence as shown in Fig. 4.8. The objective is to maximize the votes at the centre compared to other locations, which is equivalent to maximizing the

quantity

$$\Big(\sum_{y \in Y_c} S(y) - \sum_{y \notin Y_c} S(y) \Big) \tag{5.5}$$

where the first term represents sum of the votes at an area around the center and the second term indicates the votes at the background. This is equivalent to minimizing the quantity

$$\Big(\sum_{y \notin Y_c} S(y) - \sum_{y \in Y_c} S(y) \Big) \tag{5.6}$$

$$= \Big(\sum_{y \notin Y_c} \mathbf{b}_i^T(y)\mathbf{h}_i - \sum_{y \in Y_c} \mathbf{b}_i^T(y)\mathbf{h}_i \Big) = \mathbf{a}_i^T \mathbf{h}_i \tag{5.7}$$

where

$$\mathbf{a}_i = \big(\sum_{y \notin Y_c} \mathbf{b}_i(y) - \sum_{y \in Y_c} \mathbf{b}_i(y) \big) \tag{5.8}$$

denote the discriminative vote map for the i^{th} feature $\mathbf{x}_i$. For each feature in the training set, we accumulate the votes at the Hough space of the sequence to which it belongs and compute discriminative vote maps as described in Eq. 5.8. Note that the vote maps $\mathbf{b}_i$ in Eq. 5.4 computes the un-weighted votes from the matched codewords at any location y of the sequence whereas the discriminative votes maps in Eq. 5.8 computes the difference of un-weighted votes from matched codewords at the object and background.

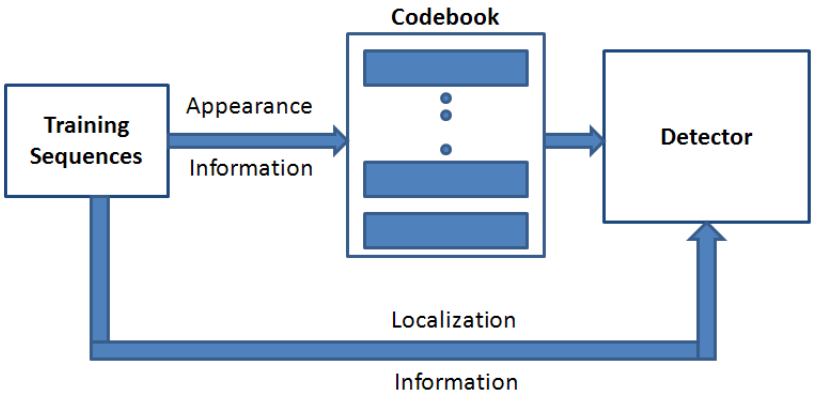
5.4. Dictionary Learning

In this section, we describe the procedure for updating the dictionary. The combined objective function is given by

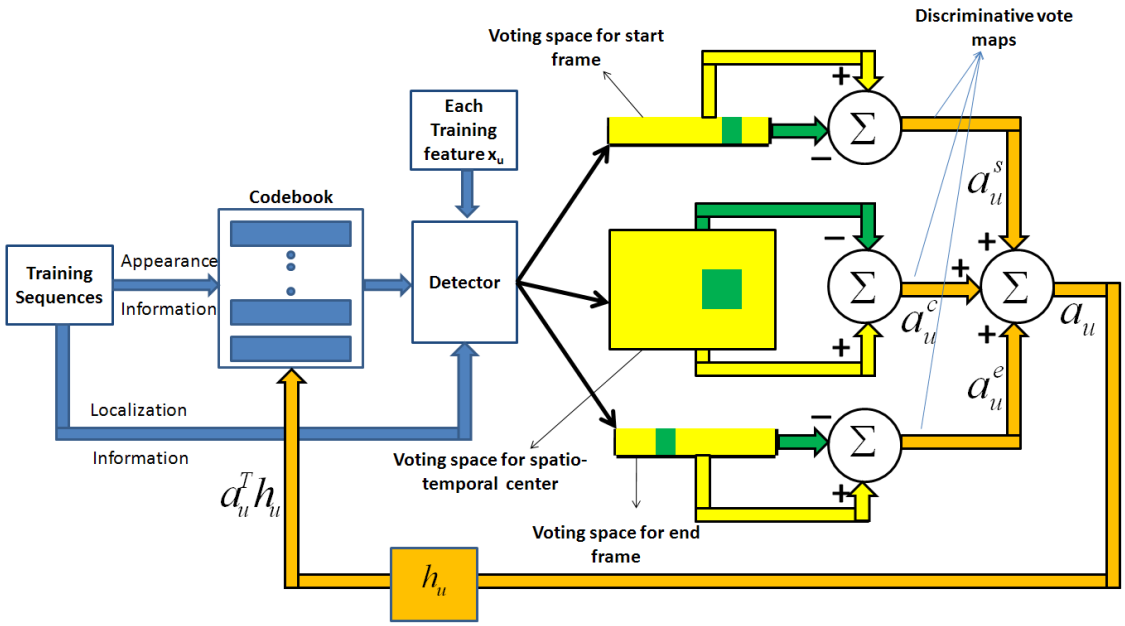
$$f(\mathbf{D}, \mathbf{h}_i) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}\mathbf{h}_i\|^2 + \eta \|\mathbf{g}_i \odot \mathbf{h}_i\|^2 + \lambda \sum_{i=1}^{n} \mathbf{a}_i^T \mathbf{h}_i \tag{5.9}$$

$$\text{s.t} \quad \mathbf{c}^T \mathbf{h}_i = 1, \quad \mathbf{h}_i \geqslant 0 \quad \forall i$$

where $\mathbf{h}_i \in \mathbb{R}^k$ are the coefficients that indicate the local weight for the training samples. The term $\|\mathbf{x}_i - \mathbf{D}\mathbf{h}_i\|^2$ and $\|\mathbf{g}_i \odot \mathbf{h}_i\|^2$ corresponds to the reconstruction error and the



(a)



(b)

Figure 5.1.: a) General approach: The codebook is generated in an unsupervised way without considering the localization information of the training sequences. b) Proposed Supervised approach which considers the localization information of the training sequences in the codebook generation process. Each training feature votes for the spatial center, start and end of the action which are collected in the respective Hough voting spaces. The discriminative vote map for each feature is generated as the sum of votes at the background (as represented by yellow) - the sum of votes at (or small bin around) the center represented by green. The discriminative localization information from three different voting maps of the training sequences are combined and supplied as input to the codebook

locality constraint as described in Eq. 4.12 and the term $\sum_{i=1}^{n} \mathbf{a}_i^T \mathbf{h}_i$ is a linear function

of the probabilistic Hough votes. The discriminative vote maps $\mathbf{a}_i$ is pre-computed from the Hough voting spaces of the training sequence as described in Eq. 5.6. The parameter $\mathbf{c} \in \mathbb{R}^k$ is given by

$$\mathbf{c}_j = \begin{cases} 1 & \text{if } dist(\mathbf{D}_j, \mathbf{x}_i) < \kappa \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

The constraints $\mathbf{c}^T\mathbf{h}_i = 1$ and $\mathbf{h}_i \geqslant 0$ ensures that $\mathbf{c} \odot \mathbf{h}_i$ is a valid probability distribution. Note that the vector $\mathbf{1}$ used as the constraint in [118] is replaced by $\mathbf{c}$ in this work. This is because the constraint $\mathbf{1}^T\mathbf{h}_i$ forces $\mathbf{h}_i$ to have non zero weights for the un-matched codewords which is inconsistent with the Hough voting. Also note that the cost function in Eq. 4.25 is a function of the local weight $\mathbf{h}_i$ and the global weight $\mathbf{w}$ since the algorithm does not update the dictionary whereas the objective function in this work (Eq. 5.9) is a function of the dictionary $\mathbf{D}$ and $\mathbf{h}_i$.

Solving for $\mathbf{h}_i$ in Eq. 5.9 yields a coefficient vector with significant weights for the local codewords. Hence similar to the approach in [118], we can select q nearest codewords for the data vector $\mathbf{x}_i$ and solve a smaller system of equations. Then the objective function can be rewritten as

$$\begin{aligned} f(\mathbf{D}, \mathbf{h}_i) = \quad & \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}^i\mathbf{h}_i\|^2 + \lambda \sum_{i=1}^{n} \mathbf{a}_i^T\mathbf{h}_i \\ & \text{s.t} \quad \mathbf{c}^T\mathbf{h}_i = 1, \quad \mathbf{h}_i \geqslant 0 \quad \forall i \end{aligned} \tag{5.11}$$

where $\mathbf{D}^i \in \mathbb{R}^{m \times q}$ is a subset of the complete dictionary $\mathbf{D}$ that contains q nearest codewords to the data vector $\mathbf{x}_i$, $\mathbf{h}_i \in \mathbb{R}^q$ and $\mathbf{a}_i \in \mathbb{R}^q$ are the coefficient vector and the discriminative vote maps for the feature $\mathbf{x}_i$ respectively. This modification also results in significant reduction in the computational complexity since $q \ll k$. Note that the cost function in Eq. 5.11 is non-convex w.r.t the $\mathbf{D}$ and $\mathbf{h}_i$. In order to optimize it, we follow an alternating optimization procedure. More precisely, we solve for one of the unknown parameter $\mathbf{D}$ and $\mathbf{h}_i$ while keeping the other parameter fixed.

5.4.1. Solving for $\mathbf{h}_i$

We first solve for the coefficients $\mathbf{h}_i$ by keeping the dictionary fixed [51]. When $\mathbf{D}^i$ is fixed the cost function in Eq. 5.11 can be solved as

$$\widehat{\mathbf{h}}_i = \arg\min_{\mathbf{h}_i} \|\mathbf{x}_i \quad - \mathbf{D}^i\mathbf{h}_i\|^2 + \lambda \mathbf{a}_i^T \mathbf{h}_i \qquad (5.12)$$
$$\text{s.t} \quad \mathbf{c}^T\mathbf{h}_i = 1, \quad \mathbf{h}_i \geqslant \mathbf{0}, \quad \forall i$$

, *i.e.*, we solve for each $\mathbf{h}_i$ separately with the constraints that the coefficients should be positive and should sum to one. These constraints are imposed to ensure that the coefficients $\mathbf{h}_i$ model $p(\mathbf{D}_j^i|\mathbf{x}_i)$. Eq. 5.12 is quadratic with respect to $\mathbf{h}_i$ and hence can be solved using a conventional quadratic solver.

5.4.2. Dictionary Update

The approaches in [17, 76] incorporate the discriminative information into dictionary learning by introducing classification cost into the objective function. In contrast, we use the localization information of the training sequences computed using discriminative vote maps to update the dictionary. We solve for $\mathbf{D}$ by keeping the coefficients computed in Eq. 5.12 fixed. Note that $\mathbf{a}_i \in \mathbb{R}^l$ is a function of $\mathbf{b}_i$ which in turn is a function of the dictionary $\mathbf{D}$ as given in Eq. 5.4. Thus the j^{th} element of the vector denoted by $\mathbf{a}_i^j$ is a function of the term $|\mathbf{D}_j|$, which indicates the total number of descriptors matched to the j^{th} column of $\mathbf{D}$ during training. We model the count of offsets for each codeword using the sum of sigmoid functions. From Eq. 5.4,

$$\mathbf{b}_i^j = \frac{L(\mathbf{x}_{ij})}{Q(\mathbf{D}_j)}, \text{ where } \; Q(\mathbf{D}_j) = \sum_{i=1}^{n} L(\mathbf{x}_{ij}) \qquad (5.13)$$

the term $Q(\mathbf{D}_j)$ approximates the count of offsets $|\mathbf{D}_j|$ in Eq. 5.4 and

$$L(\mathbf{x}_{ij}) = \left[1 + \exp\left(\frac{dist(\mathbf{D}_j, \mathbf{x}_i) - \kappa}{\sigma}\right)\right]^{-1} \qquad (5.14)$$

is the sigmoid function which assigns a value $\approx \frac{1}{|D_j|}$ to $\mathbf{b}_i^j$ only if the feature $\mathbf{x}_i$ matches with the j^{th} codeword $\mathbf{D}_j$. $dist(\mathbf{D}_j, \mathbf{x}_i)$ computes the distance between the input data vector $\mathbf{x}_i$ and the j^{th} codeword $\mathbf{D}_j$. For each input descriptor we only consider a subset of the nearest codewords $\mathbf{D}^i$ as the dictionary, and update the subset $\mathbf{D}^i$ instead of the

whole dictionary $\mathbf{D}$. The cost function in Eq. 5.11 is non-convex function w.r.t $\mathbf{D}^i$. We can use either batch methods such as batch gradient descent or the on-line methods such as stochastic gradient descent [16]. From the experiments we noticed that the batch methods converges slowly as compared to the on-line methods. Hence we use the stochastic gradient descent algorithm which takes a sample at random, computes the coefficients and updates the dictionary at each iteration. The derivative of Eq. 5.11 for a single sample w.r.t $\mathbf{D}^i$ is given by

$$\frac{\partial f}{\partial \mathbf{D}^i} = -2(\mathbf{x}_i - \mathbf{D}^i\mathbf{h}_i)\mathbf{h}_i^T + \lambda \frac{\partial}{\partial \mathbf{D}^i}(\mathbf{a}_i^T\mathbf{h}_i) \tag{5.15}$$

where the first term is the derivative of the reconstruction error and the second term is the derivative of the discriminative votemaps.

Computation of $\frac{\partial}{\partial \mathbf{D}}(\mathbf{a}_i^T\mathbf{h}_i)$: Similar to the approach in chapter 4, we use a hough voting framework for action detection where the descriptor extracted at the interest point detectors vote for the center of the action as well as the start and end frame of the action as shown in Fig. 4.11. This results in three Hough spaces as described in Section 4.5.2, one for accumulating the spatial votes and two for accumulating temporal votes. We employ discriminative information extracted from all these spaces and to learn the dictionary. Let l_c, f_s, f_e denote the spatial centre, start and end frame of a sequence in the training set respectively. let L_c, F_s, F_e denote bins around these centres as shown in Fig. 4.8. If $\mathbf{x}_i$ is a feature from the sequence, then the discriminative voting maps are given by

$$\mathbf{a}_{ci} = \Big(\sum_{y \notin L_c} \mathbf{b}_i(y) - \sum_{y \in L_c} \mathbf{b}_i(y) \Big) \tag{5.16}$$

$$\mathbf{a}_{si} = \Big(\sum_{y \notin F_s} \mathbf{b}_i(y) - \sum_{y \in F_s} \mathbf{b}_i(y) \Big) \tag{5.17}$$

$$\mathbf{a}_{ei} = \Big(\sum_{y \notin L_e} \mathbf{b}_i(y) - \sum_{y \in L_e} \mathbf{b}_i(y) \Big). \tag{5.18}$$

where $\mathbf{a}_{ci}$, $\mathbf{a}_{si}$ and $\mathbf{a}_{ei}$ represent the discriminative votemaps for the spatial, start and end Hough spaces and the combined vote map is computed as

$$\mathbf{a}_i = \mathbf{a}_{ci} + \mathbf{a}_{si} + \mathbf{a}_{ei} \tag{5.19}$$

If the feature $\mathbf{x}_i$ is matched to the codeword $\mathbf{D}_j$, then the feature uses the offsets recorded at $\mathbf{D}_j$ to cast votes. The discriminative votemaps in Eq. 5.8 represent the difference

between the fraction of votes at the background and the foreground. Let $|\mathbf{D}_j^i|$ represent the number of offsets at $\mathbf{D}_j$ and let n_c, n_s and n_e represent the number of votes inside the bins L_c, F_s and F_e respectively. Then the fraction of votes inside the bins is given by $\frac{n_c}{|\mathbf{D}_j^i|}$, $\frac{n_s}{|\mathbf{D}_j^i|}$ and $\frac{n_e}{|\mathbf{D}_j^i|}$ respectively. From Eq. 5.8 and Eq. 5.13, the j^{th} term of $\mathbf{a}_{ci}$ can be computed as

$$a_{ci}^j = \left[\left(1 - \frac{n_c}{|\mathbf{D}_j^i|}\right) - \left(\frac{n_c}{|\mathbf{D}_j^i|}\right)\right] L(\mathbf{x}_{ij}) \tag{5.20}$$

where the term $1 - \frac{n_c}{|\mathbf{D}_j^i|}$ indicates the fraction of votes at the background. Substituting Eq. 5.20 in Eq. 5.19, we get,

$$a_i^j = \left[\left(1 - \frac{n_c}{|\mathbf{D}_j^i|} - \frac{n_c}{|\mathbf{D}_j^i|}\right) + \left(1 - \frac{n_s}{|\mathbf{D}_j^i|} - \frac{n_s}{|\mathbf{D}_j^i|}\right) + \left(1 - \frac{n_e}{|\mathbf{D}_j^i|} - \frac{n_e}{|\mathbf{D}_j^i|}\right)\right] L(\mathbf{x}_{ij}) \tag{5.21}$$

$$= \left[3 - 2\left(\frac{n_c + n_s + n_e}{|\mathbf{D}_j^i|}\right)\right] L(\mathbf{x}_{ij}) \tag{5.22}$$

During training, $|\mathbf{D}_j^i|$ is approximated using $Q(\mathbf{D}_j^i)$. Hence replacing $|\mathbf{D}_j^i|$ by $Q(\mathbf{D}_j^i)$ in Eq. 5.21, we see that a_i^j is a product of functions of $\mathbf{D}_j^i$ and this derivative can be solved using the product rule which also requires the derivative of $\frac{1}{Q(\mathbf{D}_j^i)}$. The codewords are computed from the input descriptors and hence can be assumed to be independent of each other. This assumption enables us to treat the columns of $\mathbf{D}^i$ independent and compute derivative w.r.t each column of $\mathbf{D}^i$ separately, *i.e.*,

$$\frac{\partial(\mathbf{a}_i^T \mathbf{h}_i)}{\partial \mathbf{D}^i} = \left[\frac{\partial(a_i^1 h_i^1)}{\partial \mathbf{D}_1^i} \quad \cdots \quad \frac{\partial(a_i^q h_i^q)}{\partial \mathbf{D}_q^i}\right] \tag{5.23}$$

where a_i^z, h_i^z represent the z^{th} terms of the vectors $\mathbf{a}_i$ and $\mathbf{h}_i$ respectively and $\mathbf{D}_z^i$ represents the z^{th} column of the dictionary $\mathbf{D}^i$. We can substitute the value of a_i^j in Eq. 5.21 and use the derivatives

$$\frac{\partial}{\partial \mathbf{D}_j^i}\left(\frac{1}{Q(\mathbf{D}_j^i)}\right) = \frac{2}{\sigma[Q(\mathbf{D}_j^i)]^2} \sum_{i=1}^{n} L(\mathbf{x}_i)[1 - L(\mathbf{x}_i)](\mathbf{D}_j^i - \mathbf{x}_i) \tag{5.24}$$

$$\frac{\partial}{\partial \mathbf{D}_j^i}\Big(L(\mathbf{x}_{ij})\Big) = \frac{2}{\sigma} L(\mathbf{x}_{ij})\big[L(\mathbf{x}_{ij}) - 1\big](\mathbf{D}_j^i - \mathbf{x}_i) \tag{5.25}$$

to compute the derivative of the second term in Eq. 5.15. We use the Euclidean distance for the function *dist* in Eq. 5.14. The dictionary is finally updated using stochastic

gradient descent,

$$\mathbf{D}^{t+1} \leftarrow \mathbf{D}^{t} - \frac{\gamma}{\sqrt{t}} \frac{\partial f}{\partial \mathbf{D}} \tag{5.26}$$

where the derivative is computed using Eq. 5.15. The parameters t and γ represent the iteration index and the learning rate respectively. The $\mathbf{D}$ thus obtained is projected onto the unit circle. In our implementation, we select q nearest codewords for a feature $\mathbf{x}_i$ at each iteration, compute the coefficients $\mathbf{h}_i$, and then update the corresponding columns of $\mathbf{D}$ using $\mathbf{h}_i$.

Algorithm 5: Algorithm for the proposed framework

input : $\mathbf{X}$, $\mathbf{D}init$, $MAXITER$, λ, σ, l, γ
output: $\mathbf{D}$
begin
  $\mathbf{D} = \mathbf{D}init$;
  repeat
    $S1$: Randomly select a sample $\mathbf{x}_i$ from $\mathbf{X}$
    $S2$: Compute q nearest codewords (nn) to the sample $\mathbf{x}_i$ and form a local dictionary $\mathbf{D}^i$ with these codewords , *i.e.*, $\mathbf{D}^i = \mathbf{D}(:, nn)$
    $S3$: Solve for the coefficients $\mathbf{h}_i$ using Eq. 5.12
    $S4$: Compute the derivative of $\mathbf{D}^i$ using Eq. 5.15 and Eq. 5.24.
    $S5$: Update the dictionary using Eq. 5.26.
    $S6$: Project $\mathbf{D}^i$ onto the unit circle, $\mathbf{D}^i \leftarrow \mathbf{D}^i / \|\mathbf{D}^i\|^2$
    $S7$: Update corresponding columns of the dictionary, $\mathbf{D}(:, nn) = \mathbf{D}^i$
    $S8$: Compare each sample with the updated codewords and recompute the offsets for each updated codewords.
    $S9$: Compute the discriminative vote maps $\mathbf{a}_i$ using updated dictionary
  until $iter \leq MAXITER$ ***or*** *convergence*;
end

5.4.3. Implementation

In this section, we briefly describe the detection process employed in our work. The goal of our work is to spatio-temporally localize instances of actions in video sequences, *i.e.*, to predict the center and temporal extent of the action. To achieve this we follow, the spatio-temporal Hough voting framework described in the Section 4.5.2 of chapter 4. In order to extract descriptors at salient points from the training video sequences, spatio-temporal interest point detectors are applied on the training sequences and the

descriptors are extracted at the detected salient points. To handle the actions at different sizes and speed, descriptors are extracted at different spatial and temporal scales at the interest point locations. An initial codebook is created from these descriptors using k-means algorithm. The codewords are then compared against each of the training descriptors and the relative location of matched descriptors are stored. We next incorporate the discriminative localization information into the dictionary by randomly selecting the descriptors from the training set and updating the dictionary, *i.e.*, for the randomly chosen descriptor of the training sequence we compute the local weights using Eq. 5.12. we then compute the discriminative votemaps using Eq. 5.21 and update the dictionary using Eq. 5.26. This procedure is repeated until convergence. The steps involved in training the algorithm is summarized in Algorithm 5.

During testing, a test feature $\mathbf{x}_t$ is compared against the codebook and q nearest codewords are selected. The local weights for these codewords are computed by solving for $\mathbf{h}_t$,

$$\widehat{\mathbf{h}}_t = \arg\min_{\mathbf{h}_t} \ \|\mathbf{x}_t - \mathbf{D}^t\mathbf{h}_t\|^2 \qquad \text{s.t} \quad \mathbf{c}^T\mathbf{h}_t = 1, \quad \mathbf{h}_t \geqslant 0 \tag{5.27}$$

where $\mathbf{D}^t \in \mathbb{R}^{m\times l}$ contains q nearest codewords of $\mathbf{x}_t$. Only those codewords for which $dist(\mathbf{D}^t, \mathbf{x}_t) < \kappa$ cast weighted votes to the spatio-temporal location of the action in the Hough space and also to the start and end frame of the action. The maxima of the Hough space is computed using Meanshift algorithm [24].

5.5. Background Modelling

In the previous sections, we used the ISM without considering the class information of the descriptors or we have trained the model by only considering the descriptors extracted from the foreground. Thus we have ignored the second term $p(O|\mathbf{D}_j, l_i)$ in Eq. 5.2 that specifies the confidence with which the codeword $\mathbf{D}_j$ votes for class O as opposed to other classes. It can also be viewed as a weight for the votes from the codeword $\mathbf{D}_j$. The objective is to assign high weights for the votes from the foreground descriptors and suppress the votes from the background descriptors. Learning the weights $p(O|\mathbf{D}_j, l_i)$ allows us to model the background and discriminate the descriptors extracted at the background from the foreground. It is shown in [79] that the performance of the ISM

can be improved by learning the weights $p(O|\mathbf{D}_j, l_i)$. In this section, we extend the ISM described in previous sections to learn the weights for the codewords.

5.5.1. Computation of Discriminative Votemaps

The Hough score for an object O collected at location y from the feature $(\mathbf{x}_i, l_i)$ is given by Eq. 5.2. The weight $p(O|\mathbf{D}_j, l_i)$ can be computed as

$$p(O|\mathbf{D}_j) = \frac{\frac{|n_O|}{|n_{Otr}|}}{\sum_{\forall c \in \mathbf{C}} \frac{|n_c|}{|n_{ctr}|}} \tag{5.28}$$

where $|n_c|$ is the number of votes to the object c from the codeword $\mathbf{D}_j$, $|n_{ctr}|$ is the total number of features used to learn the class c. Incorporating the weights into the computation of discriminative vote maps in Eq. 5.21, the quantity we need to minimize is,

$$\begin{aligned} &\Big(\sum_{y \notin Y_c} \big(\mathbf{w} \odot \mathbf{b}_i(y)\big)^T \mathbf{h}_i - \sum_{y \in Y_c} \big(\mathbf{w} \odot \mathbf{b}_i(y)\big)^T \mathbf{h}_i \Big) \\ = \; &(\mathbf{w} \odot \mathbf{a}_i)^T \mathbf{h}_i \end{aligned} \tag{5.29}$$

where each entry of the weight vector $\mathbf{w}$ is given by $w_j = p(O|\mathbf{D}_j)$ can be computed using Eq. 5.28 and $\mathbf{a}_i$ is computed as in Eq. 5.8. The coefficients for the input descriptors are obtained by fixing the dictionary $\mathbf{D}$ and incorporating the weighted vote maps,

$$\begin{aligned} \widehat{\mathbf{h}}_i = \arg\min_{\mathbf{h}_i} \| \mathbf{x}_i \quad &- \mathbf{D}^i \mathbf{h}_i \|^2 + \lambda (\mathbf{w}_i \odot \mathbf{a}_i)^T \mathbf{h}_i \\ &\text{s.t} \quad \mathbf{c}^T \mathbf{h}_i = 1, \quad \mathbf{h}_i \geqslant \mathbf{0}, \quad \forall i \end{aligned} \tag{5.30}$$

where $\mathbf{w}_i \in \mathbb{R}^l$ are the weights corresponding to the nearest codewords of the the descriptor $\mathbf{x}_i$.

5.5.2. Updating Dictionary

The inclusion of the weights for the discriminative vote maps in Eq. 5.30 changes the dictionary update process described in Section 5.4.2 as the weights given by $p(O|\mathbf{D}_j)$ is

a function of the dictionary. Hence Eq. 5.15 can be re-written as

$$\frac{\partial f}{\partial \mathbf{D}^i} = -2(\mathbf{x}_i - \mathbf{D}^i\mathbf{h}_i)\mathbf{h}_i^T + \lambda \frac{\partial}{\partial \mathbf{D}^i}\big((\mathbf{w}_i \odot \mathbf{a}_i)^T\mathbf{h}_i\big) \tag{5.31}$$

Following the assumption made in Section 5.4.2 that the dictionaries are independent of each other, the second term in Eq. 5.31 can be written as

$$\frac{\partial}{\partial \mathbf{D}^i}\Big((\mathbf{w}_i \odot \mathbf{a}_i)^T\mathbf{h}_i\Big) = \begin{bmatrix} \frac{\partial(a_i^1 w_i^1 h_i^1)}{\partial \mathbf{D}_1^i} & \cdots & \frac{\partial(a_i^q w_i^q h_i^q)}{\partial \mathbf{D}_q^i} \end{bmatrix} \tag{5.32}$$

w_i^z, a_i^z are the z^{th} elements of $\mathbf{w}_i$, $\mathbf{a}_i$ and computed using $w_i^z = p(O|\mathbf{D}_z^i)$ and Eq. 5.22 respectively. We compute the above derivative using product rule where the derivative of w_i is given by

$$\frac{\partial}{\partial \mathbf{D}_j^i}\big(w_i^j\big) = \frac{T_1 - T_2}{\Big(\sum_{u\in\{O,B\}} \frac{\sum_{z\in u} L(\mathbf{x}_{zj})}{|n_{utr}|}\Big)^2} \tag{5.33}$$

where

$$T_1 = \Big[\frac{1}{|n_{Otr}|}\Big(\sum_{u\in\{O,B\}} \frac{\sum_{z\in u} L(\mathbf{x}_{zj})}{|n_{utr}|}\Big)\Big(\sum_{z\in O} \nabla_{\mathbf{D}_j} L(\mathbf{x}_{zj})\Big)\Big]$$
$$T_2 = \Big[\Big(\frac{\sum_{z\in O} L(\mathbf{x}_{zj})}{|n_{Otr}|}\Big)\Big(\sum_{u\in\{O,B\}} \frac{\sum_{z\in u} \nabla_{\mathbf{D}_j} L(\mathbf{x}_{zj})}{|n_{utr}|}\Big)\Big]$$

$\nabla_{\mathbf{D}_j} L(\mathbf{x}_{zj}) = \frac{\partial(L(\mathbf{x}_{zj}))}{\partial \mathbf{D}_j^i}$ is computed using Eq. 5.25. The derivative of the term a_i^z is computed using Eq. 5.24 and Eq. 5.25. Finally we update the dictionary using Eq. 5.26.

5.6. Experimental Results

We have employed two datasets to evaluate our algorithm. We first evaluate the algorithm on a subset of KTH action dataset [101] with three actions and secondly more challenging CMU dataset [49] with five actions. We consider the single repetition of an activity as an action instance. During training we build a dictionary for each action category separately. In particular, we consider two cases for the experiments. In the first case, we only consider the descriptors extracted at the foreground (*i.e.*, the descriptors inside the action volume) for dictionary construction and hence the background is not modeled. We refer to this case as *Proposed-A* while reporting the results . In the second

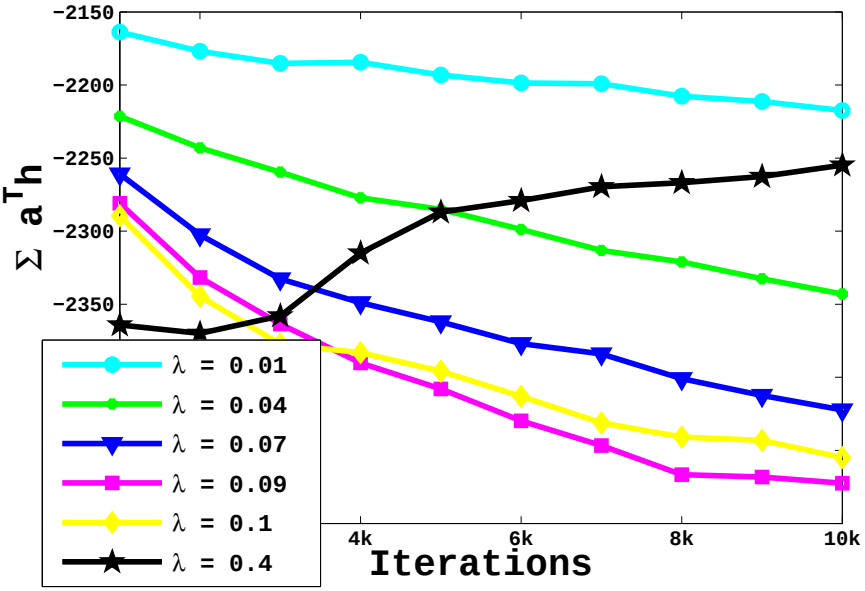
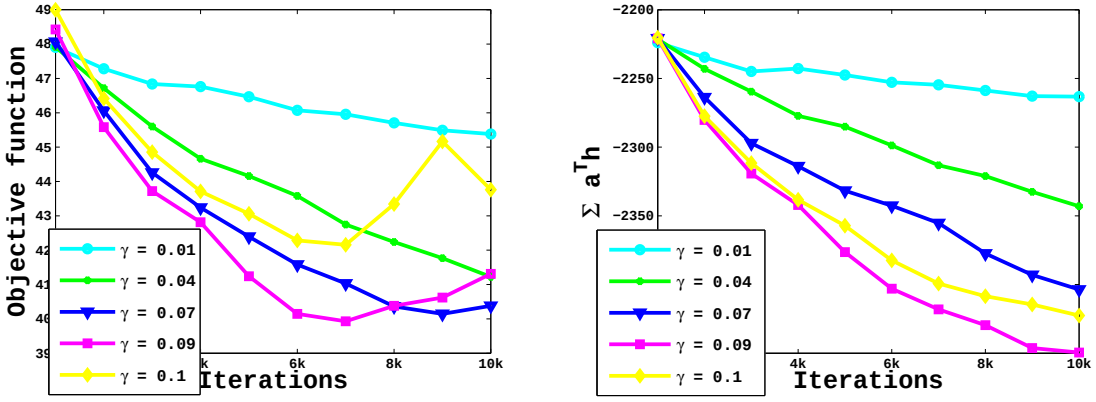


Figure 5.2.: Sum of weighted Discriminative votemaps ($\sum \mathbf{a}^T\mathbf{h}$) vs Iterations for various values of λ on action category *boxing* of the KTH dataset.

case, we learn a dictionary by extracting the descriptors both at the foreground and the background. Hence the model can discriminate the votes from the foreground and background. We refer to this case as *Proposed-B* in our experiments. For training, we apply Harris-3d interest point detector to detect salient points in the video sequence and extract HOG and HOF features [54] (dimension=162). We set the number of codewords to approximately $n/4$ where n is the total number of training descriptors. We set nearest neighbors $q = 20$ for our experiments. For a test video we extract the descriptors at the interest points and we construct a Hough space for each action category by voting with the corresponding dictionaries. The category that produces the highest response in the output Hough space is considered as the action category. The localization of the action is considered correct if the detected action label is same as the ground truth action label and the intersection to union ratio of the hypothesis and the ground truth is greater than 0.5, *i.e.*, $\frac{(V \cap G)}{(V \cup G)} > 0.5$ where V is the hypothesis and G is the ground truth.

5.6.1. Effect of Parameters Parameters λ and γ

In this section, we describe an experiment to show the effect of the parameters λ and γ on the objective function and discriminative vote maps. During training, we note that the objective function in Eq. 5.11 decreases and the second term which corresponds to



(a) (b)

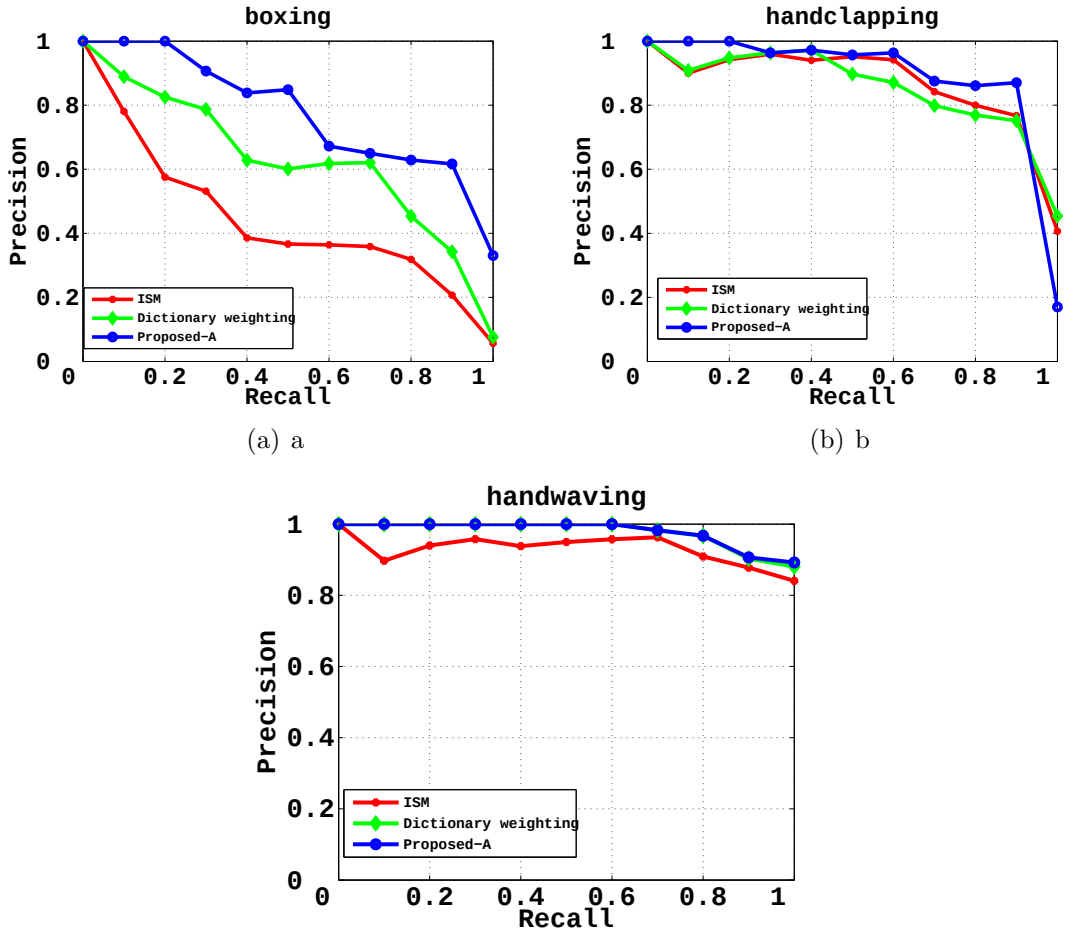
Figure 5.3.: Plot for a) Objective function vs Iterations, b) $\sum \mathbf{a}^T\mathbf{h}$ vs Iterations for various values of γ.

the sum of weighted discriminative votes (sum of the response at the background - sum of the responses at the action location) also decreases for a range of values of γ and λ.

Fig. 5.2 shows the plots for the sum of weighted discriminative votemaps ($\sum \mathbf{a}^T\mathbf{h}$) vs iterations for different values of λ for the action *boxing* of the KTH dataset. As the value of λ increases, the weight on the sum of discriminative votes ($\sum \mathbf{a}^T\mathbf{h}$) increases and results in low $\sum \mathbf{a}^T\mathbf{h}$ values. But we notice that the objective function fails to converge for high values of λ (> 0.4). We also note that the quantity $\sum \mathbf{a}^T\mathbf{h}$ decreases with the iterations for small values of λ (< 0.09) but for higher values of λ (> 0.09), $\sum \mathbf{a}^T\mathbf{h}$ increases which are shown by yellow and black curves. This is due to poor reconstruction of the descriptors (an increase in first term) which leads to poor learning of the spatial occurrence distribution and hence the sum of discriminative votemaps. We choose $0.04 \leq \lambda \leq 0.4$ for our experiments.

Table 5.1.: Average Precision values for Three actions of KTH dataset

Action/Method	ISM [60]	Dictionary weighting	Proposed-A
boxing	0.40	0.59	**0.75**
handclap	0.85	0.84	**0.87**
handwave	0.94	0.97	**0.98**
Avg	0.73	0.80	**0.87**



(a) a (b) b

(c) c

Figure 5.4.: Precision-Recall curves for three actions of the KTH dataset: a) Boxing, b) Handclapping and c) Handwaving. The red and green curves show the recall for the baseline method [60] and the dictionary weighting approach proposed in chapter 4, blue curve shows the performance of the proposed (Proposed-A) algorithm. Since the KTH dataset does not contain the descriptors extracted at the background, the results Proposed-A and Proposed-B are same

The parameter γ in Eq. 5.26 represent the learning rate for the dictionary update. We show the effect of the parameter γ on the objective function in Eq. 5.11 and the discriminative votemaps $\sum \mathbf{a}^T\mathbf{h}$. Fig. 5.3(a) shows the plots for the objective function vs iterations for various values of the learning rate. For small values of γ (< 0.09), the objective function converges slowly to a local minima. The convergence rate increases as the value of γ increases. For high values of γ (> 0.09) the algorithm fails to converge. This instability is due to large steps taken (large γ) while updating the dictionary. The variation of $\sum \mathbf{a}^T\mathbf{h}$ for different values of γ are shown in Fig. 5.3(b). From the figure, we note that the quantity $\sum \mathbf{a}^T\mathbf{h}$ shows a similar behavior as the objective function,

i.e., it decreases quickly as the γ increases but the values of $\sum \mathbf{a}^T\mathbf{h}$ starts to increase at high values of γ which is shown by the yellow curve. For our experiments, we choose the learning rate in the range $0.04 \leq \gamma \leq 0.2$.

5.6.2. Comparison with Baseline and State-of-the-art

KTH Dataset: For evaluation of the algorithm on the KTH dataset, we select a subset consisting of 25 subjects in four scenarios performing three actions (box, handclap, handwave). Each video consists of a single action. The resolution of the video is 160 × 120. We employ three-fold cross validation for evaluation, *i.e.*, among 25 videos, we use 16 for training and remaining 9 for testing. This procedure is repeated such that all videos are tested once. Since the video consists of a large number of cycles of similar repeatative activity, similar to [127], we use one or two cycles of the action for training and testing.

We learn separate dictionaries for each of the actions. The initial dictionaries are generated using k-means algorithm. The regularization parameter λ and the learning rate γ was selected empirically as $0.4 \geqslant \lambda \geqslant 0.04$ and $0.2 \geqslant \gamma \geqslant 0.04$. The parameter σ in Eq. 5.14 was set 0.001 to have a sharp cut-off between the matched and unmatched features.

The Precision-Recall curves for boxing, handclapping and handwaving actions of the KTH dataset are shown in Fig. 5.4. The methods are trained on the training part of the dataset and validated on the testing part of the each fold. The red and green curves show the recall for the baseline method [60] and the dictionary weighting approach proposed in chapter 4, blue curve shows the performance of the proposed (Proposed-A) algorithm. Since the KTH dataset does not contain the background descriptors, the results for the two proposed algorithms Proposed-A and Proposed-B are same. Fig. 5.4(a) shows the precision-recall curves for the Boxing action. It is evident from the figure that the Proposed-A method significantly outperform the baseline and the dictionary weighting approach at all precision rates. For action handclapping Fig. 5.4(b), the proposed algorithm marginally performs better than the other two algorithms at lower precision rates and for handwaving Fig. 5.4(c), the Proposed-A method performs similar to the dictionary weighting approach and marginally outperforms the ISM. The average precision for the actions are show in Table 5.1. From the table we see that the proposed method (75%) significantly outperform the baseline algorithm (40%) and the dictionary weighting approach (59%) for the boxing category. For handclapping, the average pre-

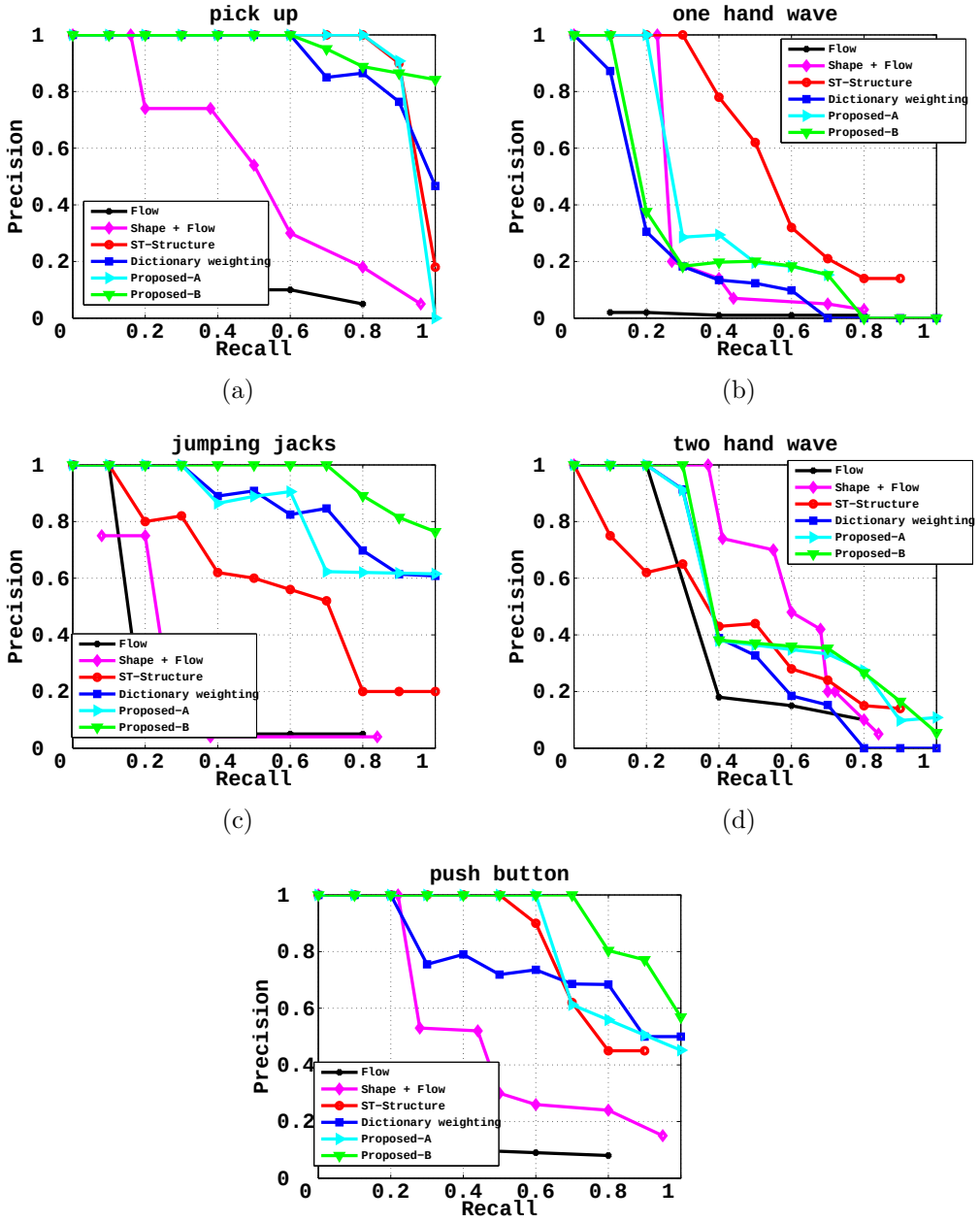


Figure 5.5.: Precision-Recall curves for five actions of the CMU dataset: a) pickup, b) one hand wave, c) jumping jacks d) two hand wave and e) push button. The magenta, black, red and blue curves correspond to the algorithms in [15], [49] , [27] (as published in [49] and [27]) and chapter 4. The cyan and green curves show the performance of the proposed approach for the dictionary learned without background (Proposed-A) and with background (Proposed-B) respectively.

cision for the proposed method (87%) is slightly better than the other two algorithms (85% and 84% for baseline and dictionary weighting approach) and for handwaving, the

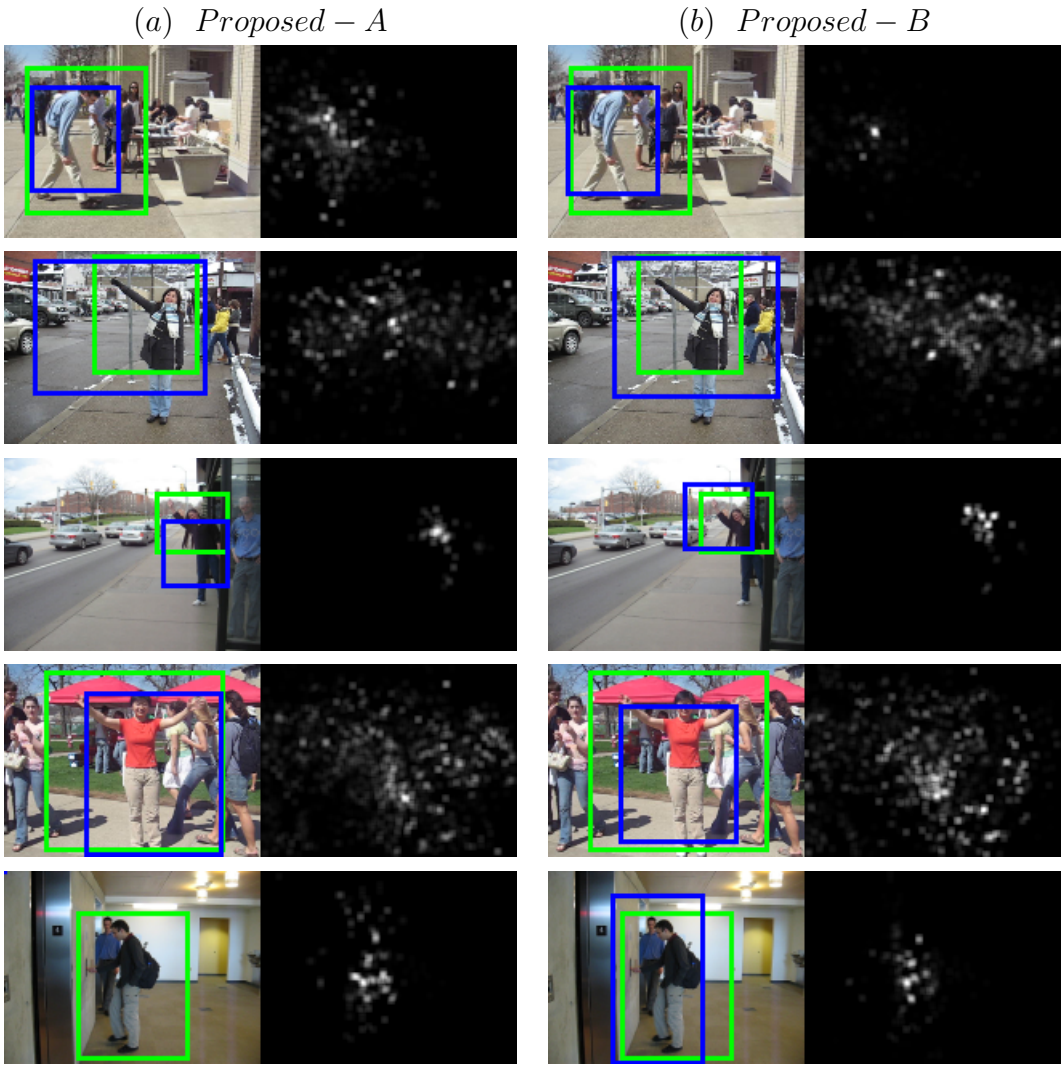


Figure 5.6.: Some false positives and misdetections for the CMU dataset: (a) First column shows the detected bounding boxes and the corresponding spatial Hough space for the method $Proposed - A$. (b) Second column shows the detected bounding boxes and the corresponding spatial Hough space for the method $Proposed - B$. The green box shows the ground truth and the blue box shows the detected bounding boxes

proposed method obtains 4% improvement in the average precision compared to [60]. Overall, the average of the average precision values for our method is 14% better than the baseline method [60] and 7% better than the dictionary weighting approach.

CMU dataset: We next compare the performance of the proposed method with the state-of-the-art methods [15], [49], [27] and the dictionary weighting approach proposed in chapter 4 on the CMU dataset [49]. We divide each action category into two non-overlapping sets, train the model on one of the sets and test the model on the other.

Figure 5.7.: Some detection results on CMU dataset: (a) First column shows the detected bounding boxes and the corresponding spatial Hough space for the method *Proposed* − *A*. (b) Second column shows the detected bounding boxes and the corresponding spatial Hough space for the method *Proposed* − *B*. We can see that modelling the background significantly reduces the background clutter and increase the votes at the center location of the object

This is repeated such that all videos are tested once. For training, we construct the initial dictionary using k-means algorithm by extracting the features from the training set using the HOG and HOF features.

Fig. 5.5 shows the precision-recall curves for five actions of the CMU dataset. The magenta, black, red and blue curves correspond to the algorithms in [15], [49] , [27] (as published in [49] and [27]) and chapter 4. The cyan and green curves show the performance of the proposed approach for the dictionary learned without background (Proposed-A) and with background (Proposed-B) respectively. For actions pick up [Fig. 5.5(a)], the proposed (Proposed-A and Proposed-B) method performs similar to the state-of-the-art at all precision except at low precision rates. The proposed-B method outperforms all the others algorithms for the action jumping jacks [Fig. 5.5(c)] and push button [Fig. 5.5(e)]. The reduction in the performance of the proposed method for the actions one hand wave [Fig. 5.5(b)] is due to significantly less number of features at the location of the action as compared to the background. The reduction in the performance of the algorithm for action two hand wave [Fig. 5.5(d)] can be attributed to the confusion with the action jumping jacks. Table 5.2 compares the average precision values for the proposed method with the other algorithms. From the table, it is evident that the proposed method (94%, 95%, 91%) has a higher average precision values for the actions pick up, jumping jacks and push button as compared to the state-of-the-art algorithms (90%, 55%, 74%). Overall the proposed method (70%) performs better than the algorithms in [15] (13%), [49] (40%), [27] (61%) and the dictionary weighting approach (60%). From Fig. 5.5 and Table 5.2 it is evident that the dictionary constructed using descriptors from both the foreground and background (Proposed-B: 70%) performs better than the dictionary constructed using only the descriptors from the foreground (Proposed-A: 66%). Fig. 5.7 shows some detection results and the corresponding spatial Hough spaces for the methods $Proposed - A$ and $Proposed - B$. From Fig. 5.7, it is evident that the background clutter is significantly reduced by modeling the background ($Proposed - B$). Fig. 5.6 shows some false and mis-detection results on the CMU dataset.

5.7. Conclusions

In this chapter, we have introduced a novel supervised dictionary learning approach for action detection. It uses the ISM in which the spatio-temporal descriptors extracted from the input sequence vote for the spatio-temporal center and the start and end of

Table 5.2.: Average Precision values for CMU dataset

Action/ Method	Flow [15]	Shap + Flow [49]	ST - Structured [27]	Dictionary weighting [51]	Proposed-A	Proposed-B
pick up	0.09	0.46	0.90	0.89	0.89	**0.94**
one hand wave	0.01	0.28	**0.52**	0.17	0.31	0.23
jumping jacks	0.14	0.21	0.55	0.84	0.82	**95**
two hand wave	0.29	**0.57**	0.37	0.40	0.48	0.49
push button	0.1	0.45	0.74	0.74	0.81	**0.91**
Average	0.13	0.4	0.61	0.60	0.66	**0.70**

the action. The ISM uses a dictionary to map the input descriptors in the continuous descriptor space to representative dictionary atoms. The descriptors use these matched dictionary atoms to vote for the hypothesis. In this work, we employ Locality Constraint linear Coding that assigns weights to votes from dictionary atoms based on the degree of match between the descriptor and the dictionary atom. This is in contrast to the ISM that assigns equal (uniform) weights. In addition, we proposed a framework that enables us to incorporate the localization information for learning the dictionary. More specifically, we measure the contribution of the votes from each dictionary atom to the hypothesis and background of the training sequences and use this discriminative information to update the dictionary. This is in contrast to the ISM that only use the appearance information of the descriptors to learn the dictionary. The resulting dictionary maximizes the response at the spatio-temporal center and temporal extent of the action and hence adapted for the task of localization. In order to discriminate the background descriptors from the foreground descriptors, we extended the proposed model to include background information while learning the dictionary.

We tested the algorithm on the challenging CMU and a subset of the KTH datasets. The proposed method $Proposed-A$ outperforms the dictionary weighting method which indicates the importance of updating the dictionary according to the subsequent task. Furthermore, the method $Proposed-B$ outperforms $Proposed-A$ which shows the advantages of learning a discriminative dictionary that can distinguish the foreground and background descriptors.

Chapter 6.

Conclusions And Future Work

In this thesis, we have investigated supervised dictionary learning methods for action recognition and action localization. In chapter 3, we employed NMF as a feature extraction technique for action recognition. In contrast to the traditional approaches where the NMF stage and the classifier stage are separated, we proposed to incorporate the discriminative information from the classifier to the learning of NMF. This is achieved by imposing max-margin constraints on the formulation of NMF (in the linear case) or the KNMF (in the non linear case). We converted this non-convex optimization problem into convex subproblems by employing block coordinate descent algorithm in which we fixed a subset of the parameters and solve for the remaining parameters. The resulting bases maximize the margin of the classifier in the low dimensional subspace (for the linear case) or the high dimensional feature space (for the non-linear case).

We showed experimentally that the factorization matrices and the classifier parameters ($\mathbf{G}$, $\mathbf{H}$, $\mathbf{w}$) converge after few iterations indicating that the objective function is guaranteed to converge to a local minima. We first evaluated the method on a toy dataset which demonstrated the advantages of incorporating the supervised information (class information) during learning of the dictionary. Furthermore, we evaluated the performance of the algorithm on several computer vision applications such as facial expression recognition, object classification, pedestrian detection and action recognition. In all the cases, the proposed method outperformed its unsupervised counterpart. The KMNMF outperforms the MNMF in all the experiments, but at the cost of increased computational power.

In the second part (Chapter 4 and Chapter 5) of the thesis, we proposed two algorithms for the task of action localization. We employed a part-based object detection technique called the ISM for action localization in which the local descriptors extracted

from the video cast votes for the spatial center, start and end frames (temporal extent) of the action. In order to efficiently assign the local descriptors to the dictionary atoms, we incorporated the Hough voting scheme into LLC framework which assigns local weights for the codewords based on the degree of match between the descriptor and the codewords. Since the dictionary is learned in an unsupervised manner using the k-means algorithm, all the dictionary atoms may not contribute votes equally to the hypothesis. So we introduced a weighting scheme that assigns global weights to the dictionary atoms based on the contribution of votes to the hypothesis.

We demonstrated the effectiveness of learning weights to the dictionary through a series of experiments. The experiments in Section 4.6.1 showed that the algorithm approaches ISM for large values of λ and a good performance is achieved for certain range of λ. In the subsequent sections, we compared our dictionary weighting approach with the ISM and state-of-the-art methods. We achieved 7% improvement in the performance compared to the ISM and on par with the state-of-the-art method.

We showed in Chapter 4 that the performance of the detector is improved by learning weights for the dictionary atoms. However the underlying dictionary is learned using unsupervised k-means algorithm. This motivated us to develop task dependent dictionary for action localization (Chapter 5). This is accomplished by developing a framework that incorporates the Hough voting information into training stage of the dictionary. We computed discriminative votes from the output Hough space of the training images and employ this information to learn the ISM dictionary. In this way, we learned a task dependent (supervised) dictionary by incorporating the localization information of the training sequences. The resultant dictionary maximize the Hough voting response at the location of the center of the action as compared to the background. We also extended the above approach to include the background model into dictionary learning.

We evaluated the performance of the proposed method and compared it with state-of-the art methods in Section 5.6. The experiments showed that the objective function reduces the sum of discriminative votes indicating that the response at the center of the action is increased as compared to the background. In addition, we also showed that by learning the dictionary in a supervised manner, we achieve 7% and 6% improvement in the performance compared to the dictionary weighting approach for KTH and CMU datasets respectively. Furthermore, the proposed method outperforms the ISM by 14% and state-of-the-art method by 5%.

The work in this thesis shows that the performance of the action recognition and localization systems that use the dictionary learning algorithms can be significantly improved by incorporating the supervised information while learning the dictionary. Typically, the dictionary learning algorithms employ a generic reconstruction error criterion to learn the dictionaries from the input data. However in literature, it is shown that such representation capabilities of the dictionary may not be optimal for the discrimination between two classes. A good dictionary for the classification task should also include the supervised information while learning the dictionary. This is evident from the experiments in Chapter 3. Further, we have shown in Chapter 5 that when learning a dictionary for a specific task, it is important to include the task specific information into the dictionary learning stage so that the learned dictionary is adaptive to the task.

6.1. Future Work

We first describe the future directions for the MNMF algorithm. The non-negative constraints on the bases and the coefficients of NMF results in a part-based representation. However it was showed in [23, 64] that more meaningful parts can be obtained by explicitly imposing local constraints on the bases and the coefficients. Imposing explicit local constrains along with max-margin could result in the bases that can clearly distinguish the features belonging to different classes. But this formulation introduces two quantities that are functions of the bases and coefficients into the objective function making it tedious. Further, the MNMF framework described in this thesis is limited to binary classification. An extension to multi-class problems would be interesting. It is also interesting to develop a theoretical framework that can automatically select the parameters used in the experiment.

We now describe the future developments for the supervised dictionary learning problem described in Chapter 5. In this work, we learn a dictionary for each of the action classes. Since the dictionaries are learned independently, the dictionary learning may not capture the inter-class variations. A possible direction of research could be to learn a single dictionary for all the classes which can efficiently capture the intra and inter-class variations. This would also reduce the number of parameters required for learning the dictionary. Also it would be interesting to develop a theoretical framework that can automatically select the parameters required for dictionary learning.

Bibliography

[1] A. Pinz A. Opelt and A. Zisserman. A boundary-fragment-model for object detection. In *European Conf. on Computer Vision*, 2006.

[2] X.Munoz A.Bosch, A.Zisserman. Image classification using random forests and ferns. In *Int'l Conf. Computer Vision*, pages 1–8, 2007.

[3] A. Agarwal and B. Triggs. A local basis representation for estimating human pose from cluttered images. In *ACCV*, pages 50–59, 2006.

[4] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, November 2004.

[5] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Comput. Surv.*, 43(3):16, 2011.

[6] M. Aharon, M. Elad, and A. Bruckstein. -svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Trans. Sig. Proc.*, 54(11):4311–4322, nov 2006.

[7] M. Aharon, M. Elad, and A.M. Bruckstein. K-svd and its non-negative variant for dictionary design. In *Proceedings of the SPIE conference wavelets*, volume 5914, July 2005.

[8] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[9] D. H. Ballard. Readings in computer vision: issues, problems, principles, and paradigms. chapter Generalizing the hough transform to detect arbitrary shapes, pages 714–725. 1987.

[10] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417, 2006.

[11] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear programming: theory and algorithms*. John Wiley and Sons, 2006.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[13] Aaron F. Bobick, James W. Davis, Ieee Computer Society, and Ieee Computer Society. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:257–267, 2001.

[14] I. Bociu and I. Pitas. A new sparse image representation algorithm applied to facial expression recognition. In *Machine Learning for Signal Processing, 2004. Proceedings of the 2004 14th IEEE Signal Processing Society Workshop*, pages 539 –548, 2004.

[15] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[16] Léon Bottou. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, 1998.

[17] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *CVPR*, pages 2559–2566, 2010.

[18] C. Boutsidis and E. Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[19] V. Vapnik C. Cortes. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[20] D. Cai, X. He, X. Wu, and J. Hans. Non-negative matrix factorization on manifold. In *ICDM*, 2008.

[21] Hongping Cai, Fei Yan, and Krystian Mikolajczyk. Learning weights for codebook in image classification and retrieval. In *CVPR*, pages 2320–2327, 2010.

[22] Liangliang Cao, Zicheng Liu, and Thomas S. Huang. Cross-dataset action detection. In *CVPR*, pages 1998–2005, 2010.

[23] X. Chen, L. Gu, S. Z. Li, and H-J. Zhang. Learning representative local features for face detection. In *CVPR*, volume 1, pages 1126–1131, 2001.

[24] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):790–799, aug 1995.

[25] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[26] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[27] Konstantinos G. Derpanis, Mikhail Sizintsev, Kevin J. Cannons, and Richard P. Wildes. Efficient action spotting based on a spacetime oriented structure representation. In *CVPR'10*, pages 1990–1997, 2010.

[28] Onur Dikmen and Cdric Fvotte. Nonnegative dictionary learning in the exponential noise model for adaptive music signal representation. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2267–2275. 2011.

[29] C. H. Q. Ding, T. Li, and M. I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, 2010.

[30] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proceedings of the 14th International Conference on Computer Communications and Networks*, ICCCN '05, pages 65–72, 2005.

[31] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 726–, 2003.

[32] G. Fanelli, J. Gall, and L. Van Gool. Hough transform-based mouth localization for audio-visual speech recognition. In *British Machine Vision Conference*, September 2009.

[33] Alireza Fathi and Greg Mori. Action recognition by learning mid-level motion features. In *CVPR*, 2008.

[34] Li Fei-Fei, Rob Fergus, and Pietro Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1134–, 2003.

[35] P. Felzenszwalb, D. McAllester, and D. Ramaman. A discriminatively trained, multiscale, deformable part model. In *European Conference on Computer Vision*, 2008.

[36] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–264 – II–271 vol.2, june 2003.

[37] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminant models for object category detection. In *Proceedings of International Conference on Computer Vision*, Beijing, China, OCT 2005.

[38] Jürgen Gall and Victor Lempitsky. Class-specific Hough forests for object detection. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 1022–1029, 2009.

[39] Shenghua Gao, Ivor Wai-Hung Tsang, Liang-Tien Chia, and Peilin Zhao. Local features are not lonely - laplacian sparse coding for image classification. In *CVPR*, pages 3555–3561, 2010.

[40] Jan C. Gemert, Jan-Mark Geusebroek, Cor J. Veenman, and Arnold W. Smeulders. Kernel codebooks for scene categorization. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 696–709, 2008.

[41] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2247–2253, dec 2007.

[42] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[43] Bernd Heisele, Thomas Serre, Massimiliano Pontil, and Tomaso Poggio. Component-based face detection. In *CVPR (1)*, pages 657–662, 2001.

[44] P. O. Hoyer. Non-negative sparse coding. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002.

[45] Zhuolin Jiang, Zhe Lin, and Larry S. Davis. Learning a discriminative dictionary for sparse coding via label consistent k-svd. In *CVPR*, pages 1697–1704, 2011.

[46] Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, ICCV '05, pages 604–610, 2005.

[47] Timor Kadir and Michael Brady. Saliency, scale and image description. *Int. J. Comput. Vision*, 45(2):83–105, nov 2001.

[48] Yan Ke, Rahul Sukthankar, and Martial Hebert. Efficient visual event detection using volumetric features. In *ICCV*, pages 166–173, 2005.

[49] Yan Ke, Rahul Sukthankar, and Martial Hebert. Event detection in crowded videos. In *ICCV*, pages 1–8, 2007.

[50] I. Kotsia, S. Zafeiriou, and I. Pitas. A novel discriminant non-negative matrix factorization algorithm with applications to facial image characterization problems. *IEEE Transactions on Information Forensics and Security*, 2(3):588–595, 2007.

[51] B. G. Vijay Kumar and Ioannis Patras. Learning codebook weights for action detection. In *CVPR Workshops*, pages 27–32, 2012.

[52] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Int'l Conf. Computer Vision and Pattern Recognition*, volume 0, pages 1–8, 2008.

[53] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[54] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[55] Svetlana Lazebnik and Maxim Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(7), jul 2009.

[56] L.Breiman. Random forests. In *Machine Learning*, September 2001.

[57] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[58] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[59] Bastian Leibe. Interleaved object categorization and segmentation. In *PhD Thesis No. 15752*, 2004.

[60] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *In ECCV workshop on statistical learning in computer vision*, pages 17–32, 2004.

[61] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, 2008.

[62] Bastian Leibe and Bernt Schiele. Interleaved object categorization and segmentation. In *In BMVC*, page 759–768, 2003.

[63] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 775–781, 2005.

[64] S. Z. Li, X. W. Hou, H. J. Zhang, and Q. S. Cheng. Learning spatially localized, parts-based representation. In *CVPR*, volume 1, pages 207–212, 2001.

[65] C. J. Lin. Projected gradient methods for non-negative matrix factorization, 2005. Tech. Rep., Department of Computer Science, National Taiwan University.

[66] C. J. Lin. On the convergence of multiplicative update algorithms for non-negative matrix factorization. *IEEE Transactions on Neural Networks*, pages 1589–1596, 2007.

[67] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116, 1998.

[68] Lingqiao Liu, Lei Wang, and Xinwang Liu. In defense of soft-assignment coding. In *ICCV*, pages 2486–2493, 2011.

[69] W. Liu and N. Zhen. Non-negative matrix factorization based methods for object recognition. In *Pattern Recognition Letters*, volume 25, pages 893–897, 2004.

[70] W. Liu, N. Zheng, and X. Lu. Non-negative matrix factorization for visual coding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 293–296, 2003.

[71] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, nov 2004.

[72] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205, 1998.

[73] Julien Mairal, Francis Bach, and Jean Ponce. Task-driven dictionary learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):791–804, 2012.

[74] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[75] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, 2008.

[76] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. In *NIPS*, pages 1033–1040, 2008.

[77] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008.

[78] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008.

[79] Subhransu Maji and Jitendra Malik. Object detection using a max-margin hough transform. In *Proceedings of Computer Vision and Pattern Recognition*, pages 1038–1045, 2009.

[80] Stphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd edition, 2008.

[81] Marcin Marszałek and Cordelia Schmid. Spatial weighting for bag-of-features. In *IEEE Conference on Computer Vision & Pattern Recognition*, volume 2, pages 2118–2125, jun 2006.

[82] Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *Workshop on Video-Oriented Object and Event Classification, ICCV 2009*, September 2009.

[83] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV '09: Proceedings of the Twelfth IEEE International Conference on Computer Vision*, Washington, DC, USA, 2009. IEEE Computer Society.

[84] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 128–142. Springer, 2002.

[85] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(4):349–361, 2001.

[86] Frank Moosmann, Bill Triggs, and Frederic Jurie. Fast discriminative visual codebooks using randomized clustering forests. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, pages 985–992. 2007.

[87] A. Oikonomopoulos, I.Patras, and M.Pantic. Discriminative space-time voting for joint recognition and localization of actions. In *ACM*, 2010.

[88] A. Oikonomopoulos, I. Patras, and M. Pantic. Spatiotemporal salient points for visual recognition of human actions. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 36(3):710–719, 2006.

[89] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[90] P. Paatero and U. Tapper. Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37:23–35, 1997.

[91] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmann, and R. D. Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsnmf). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):403–415, 2006.

[92] Duc-Son Pham and Svetha Venkatesh. Joint learning and dictionary construction for pattern recognition. In *CVPR*, 2008.

[93] Ronald Poppe. A survey on vision-based human action recognition. *Image Vision Comput.*, 28(6):976–990, jun 2010.

[94] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach: A spatio-temporal maximum average correlation height filter for action recognition. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2008.

[95] Rémi Ronfard, Cordelia Schmid, and Bill Triggs. Learning to parse pictures of people. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, pages 700–714, 2002.

[96] P. M. Roth, T. Mauthner, I. Khan, and H. Bischof. Efficient human action recognition by cascaded linear classification. In *IEEE Workshop on Video-Oriented Object and Event Classification*, 2009.

[97] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visiual scenes. In *NIPS*, pages 875–881, 1996.

[98] Konrad Schindler and Luc J. Van Gool. Action snippets: How many frames does human action recognition require? In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*, 2008.

[99] Henry Schneiderman and Takeo Kanade. Object detection using the statistics of parts. *Int. J. Comput. Vision*, 56(3):151–177, 2002.

[100] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *British Machine Vision Conference*, 2008.

[101] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *ICPR*, pages 32–36, 2004.

[102] Edgar Seemann, Bastian Leibe, Krystian Mikolajczyk, and Bernt Schiele. An evaluation of local shape-based features for pedestrian detection. In *British Machine Vision Conference*, 2005.

[103] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, 1987.

[104] Parthipan Siva and Tao Xiang. Action detection in crowd. In *BMVC*, pages 1–11, 2010.

[105] S. M. Smith and J. M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23:45–78, 1995.

[106] C. G. M. Snoek, M. Worring, J. C. V. Gemert, J. M. Geusebroek, and A. W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the ACM Multimedia*, pages 421–430, 2006.

[107] Richard Souvenir and Justin Babbs. Learning the viewpoint manifold for action recognition. In *CVPR*, 2008.

[108] Josephine Sullivan and Stefan Carlsson. Recognizing and tracking human action. In *ECCV (1)*, pages 629–644, 2002.

[109] Ju Sun, Xiao Wu, Shuicheng Yan, Loong Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *CVPR*, pages 2004–2011, 2009.

[110] J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[111] C. Thurau and V. Hlavac. Pose primitive based human action recognition in videos or still images. In *CVPR*, pages 1–8, 2008.

[112] Pavan K. Turaga, Rama Chellappa, V. S. Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *IEEE Trans. Circuits Syst. Video Techn.*, 18(11):1473–1488, 2008.

[113] Tinne Tuytelaars and Krystian Mikolajczyk. *Local Invariant Feature Detectors: A Survey*. Now Publishers Inc., 2008.

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

[115] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR (1)*, pages 511–518, 2001.

[116] David S. Vogel, Ognian Asparouhov, and Tobias Scheffer. Scalable look-ahead linear regression trees. In *KDD '07: Proceedings of the 13th ACM SIGKDD in-*

ternational conference on Knowledge discovery and data mining, pages 757–764, 2007.

[117] Heng Wang, A. Klaser, C. Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 3169–3176, 2011.

[118] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas S. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010.

[119] Y. Wang and Y. Jia. Fisher non-negative matrix factorization for learning local features. In *ACCV*, volume 1, pages 27–30, 2004.

[120] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 101 –108 vol.2, 2000.

[121] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Comput. Vis. Image Underst.*, 115(2):224–241, feb 2011.

[122] Geert Willems, Tinne Tuytelaars, and Luc Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 650–663, 2008.

[123] Adrian Wills. *QPC: Quadratic Programming in C*. Software available at `http://sigpromu.org/quadprog/`.

[124] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009.

[125] Jianchao Yang, Kai Yu, and Thomas S. Huang. Supervised translation-invariant sparse coding. In *CVPR*, pages 3517–3524, 2010.

[126] Liu Yang, Rong Jin, Rahul Sukthankar, and Frédéric Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *Conference on Computer Vision & Pattern Recognition*, 2008.

[127] Angela Yao, Jürgen Gall, and Luc J. Van Gool. A hough transform-based voting framework for action recognition. In *CVPR*, pages 2061–2068, 2010.

[128] Alper Yilmaz and Mubarak Shah. Actions sketch: A novel action representation. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 984–989, 2005.

[129] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato. A 3d facial expression database for facial behavior research. In *Proc. IEEE Intl Conf. Face and Gesture Recognition*, pages 211–216, 2006.

[130] Gang Yu, Junsong Yuan, and Zicheng Liu. Unsupervised random forest indexing for fast action search. In *CVPR*, pages 865–872, 2011.

[131] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2223–2231. 2009.

[132] Junsong Yuan, Zicheng Liu, and Ying Wu. Discriminative video pattern search for efficient action detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1728–1743, 2011.

[133] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas. Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification. *IEEE Transactions on Neural Networks*, 17(3):683–695, 2006.

[134] D. Zhang, Z. Zhou, and S. Chen. Non-negative matrix factorization on kernels. In *PRICAI*, pages 404–412, 2006.

[135] Qiang Zhang and Baoxin Li. Discriminative k-svd for dictionary learning in face recognition. In *CVPR*, pages 2691–2698, 2010.

[136] Wei Zhang, Akshat Surve, Xiaoli Fern, and Thomas Dietterich. Learning non-redundant codebooks for classifying complex objects. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1241–1248, 2009.

[137] Yimeng Zhang and Tsuhan Chen. Implicit shape kernel for discriminative learning of the hough transform detector. In *BMVC*, pages 105.1–105.11, 2010.

Appendix A.

A Discriminative Voting Scheme for Object Detection using Hough Forests

In this chapter, we describe our preliminary work on object detection where we proposed a supervised framework for learning the trees of a Hough Forest. Due to its simplicity, versatility and speed, Random Forests have been widely used for many computer vision tasks such as Face feature detection, object detection, pose estimation, action recognition etc. In this chapter, we first give a brief survey of the object detection techniques. We describe an adaptation of Random forests for object detection called Hough forests. Further, we propose a supervised framework for Hough forest that discriminates the object parts from the background for object detection and we finally demonstrate the performance of the proposed algorithm using publicly available datasets.

A.1. Related Work

In this section, we discuss an overview of features and approaches employed for the task of object detection. A local feature is an image pattern which differs from its immediate neighborhood [113]. A local features or key points are necessary for the visual tasks because a) they might have specific semantic interpretation pertaining to a certain application. b) they provide a limited set of localized and individually identifiable anchor points. c) they can be used as a set of robust image representation. The overall performance of a detector also depends on the reliability, accuracy, invariance to certain

image attributes and repeatability with which key points can be detected for a certain class. The most commonly used key point detectors are Harris edge detector [42], Laplacian [67], Difference-of-Gaussians(DoGs) [71], Laplacian of Gaussian (LoG) [84], Histogram of Oriented Gradients (HoG) [26], SUSAN [105], SIFT [71], SURF [10] etc.

Most of the object detection techniques follow either the sliding window approach [26, 35, 78, 97, 99] or the Hough transform [9, 60, 62, 79] framework. Kanade *et al.* [97] used sliding windows for the extraction of facial images in images. Small windows of size 20×20 are extracted at different scales from the image and pre-processed before passing through a neural-network classifier which detects whether the input window has the face or not. In [99], the object detector consists of multiple classifiers, each spanning a different range of orientation that predict the object in a given window of the image. Each classifier computes the part (a transform from a subset of wavelet coefficients to a discrete set of values) values within the local window and looks up their associated class conditional probabilities. The classifier then uses a likelihood ratio test to make a decision.

Dalal *et al.* [26] used a collection of Histogram of Oriented Gradients (HoGs) computed at small spatial regions called "cells" within a detector window. A linear SVM classifier is employed to classify the window. Ramanan *et al.* [35] follows a similar approach where each window consists of a root filter and several part models. Each part model specifies a spatial model and a part filter. The spatial model defines a set of allowed placements for a part relative to a detection window, and a deformation cost for each placement. The score for a detection window is the score of the root filter on the window plus the sum over parts, of the maximum over placements of that part, of the part filter score on the resulting subwindow minus the deformation cost. HoG features are used as descriptors for both root and part filters. In [78], histogram intersection kernel SVM's (IKSVM's) are applied on multilevel features extracted from a window to classify the window. To obtain multilevel features, a grayscale window is convolved with eight filters oriented in different direction to obtain oriented energy responses. These responses are then L_1 normalized over all directions in each 16×16 blocks independently to obtain normalized responses. Multilevel features are then extracted by construction histograms of oriented gradients by summing up the normalized response in each cell.

Mohan *et al.* [85] used a part-based object model where a set of hand-picked parts of the image (arms,face,legs) are learned during training. During testing, component detectors are applied to a window of the image. The response of these component detectors are applied to a combined classifier which predicts the window as "person"

or "non-person". Heisele *et al.* [43] followed a similar approach as [85] but instead of hand-picked components as in [85], the algorithm learns the components automatically.

As opposed to sliding window approach which classifies the windows extracted at all locations at different scales of the image, a part-based approach [1,60,62,79] divides the object into patches and use the information from the parts of the object to detect the hypothesis. Leibe *et al.* [60,62] introduced Implicit Shape Model (ISM) where the parts of the object are used for detection as opposed to the sliding window approaches. During learning, a vocabulary of visual patches called codebooks are learned using training patches in an unsupervised manner. During testing a generalized Hough transform [9] is used by the patches of the images to cast votes to the center of the object. In [1], a similar ISM is used where boundary fragments (from the boundaries of the training objects) are used instead of appearance of the patches. It also constructs a strong detector (rather than a classifier) by boosting over a set of weak detectors built on boundary fragments. Malik *et al.* [79] employed a discriminative approach by computing weighs for the votes from the patches. The learning framework takes into account both the appearance of the part and the spatial distribution of its position with respect to the object center and parts which are both repeatable and occur at a consistent location are assigned higher weights. This is accomplished using a max-margin formulation.

In this work, we use Hough forest technique [38] that uses random forests for learning the visual codebooks for the ISM. Random forests have been used in many computer vision problems such as classification [2], segmentation [100], object detection [38] and so on. Most of the recent image classification techniques treat images as a collection of patches characterized by local visual descriptors. Patches can be obtained by sampling the images densely or at selected salient points. These patches are encoded as vectors by extracting various features. The features can be as simple as gray level values or can be complex features like SIFT, PHOG, SURF etc. These feature vectors are vector quantized or clustered to form visual codebooks. Many of the methods use unsupervised k-means for the construction of visual codebooks. For good classification performance, a large codebook is required [38], this makes learning computationally expensive. Similarly during testing, each feature vector is compared with all the codebooks which also is time consuming. Further, a single data structure cannot capture the richness and diversity of the high dimensional feature vectors [86]. All these reasons led to the use of ensemble trees [56] for clustering which is known for its simplicity, speed and performance.

Moosmann et. al. [86] used extremely randomized trees for building fast discriminative visual codebooks. Each leaf node of each tree is assigned a distinct region label

(visual word). A histogram of labels is obtained for an image and this histogram is classified using a global SVM classifier. Lepetit et al. [63] introduced randomized trees for fast keypoint recognition. In this method, trees are used not only for keypoint matching but also to select two pixels in the neighborhood of keypoints which improves the matching performance. Gall *et al.* [38] used Random forests for object detection where the class as well as the spatial information of the patches are learned in a supervised manner. The leaves of the trees form a discriminative codebook and store the spatial information along with the class information. This spatial information encodes the location of the center of the object and is used to cast probabilistic Hough votes to the center of the object. Fanelli *et al.* [32] followed a similar Hough forest approach for the localization of the mouth in facial images.

A.2. Hough Forests for Object Detection

Numerous approaches have been used for object detection in recent years. Among these, the Implicit Shape Model (ISM) introduced by Leibe [60, 62] forms the basis for many object detection algorithms that use a part-based approach. As the name suggests, the ISM does not define an explicit model for all possible shapes of an object class but instead define allowed shapes implicitly in terms of which local appearances are consistent with each other. This makes the model more flexible and also the shapes can be learned with fewer examples [60]. In this section, we describe the Hough forest technique for object detection [38]. The overall steps involved in [38] can be summarized as shown in Fig. A.1. . During training, the ISM learns a model of the spatial occurrence distributions of local patches with respect to anchor points, such as the object center. During testing, this learned model is used to cast probabilistic votes to the location of the center of the object based on the generalized Hough transform technique [9].

A.2.1. Codebook Generation

The codebook generation is a process in the training where the patches extracted from training samples are learned to build a vocabulary of local appearances. In [60], an interest point detector is applied on the training images to extract features of the image. This results in reduced amount of data to be processed as compared to uniform sampling of the image. A patch of size 25×25 pixels is extracted around each interest point. These

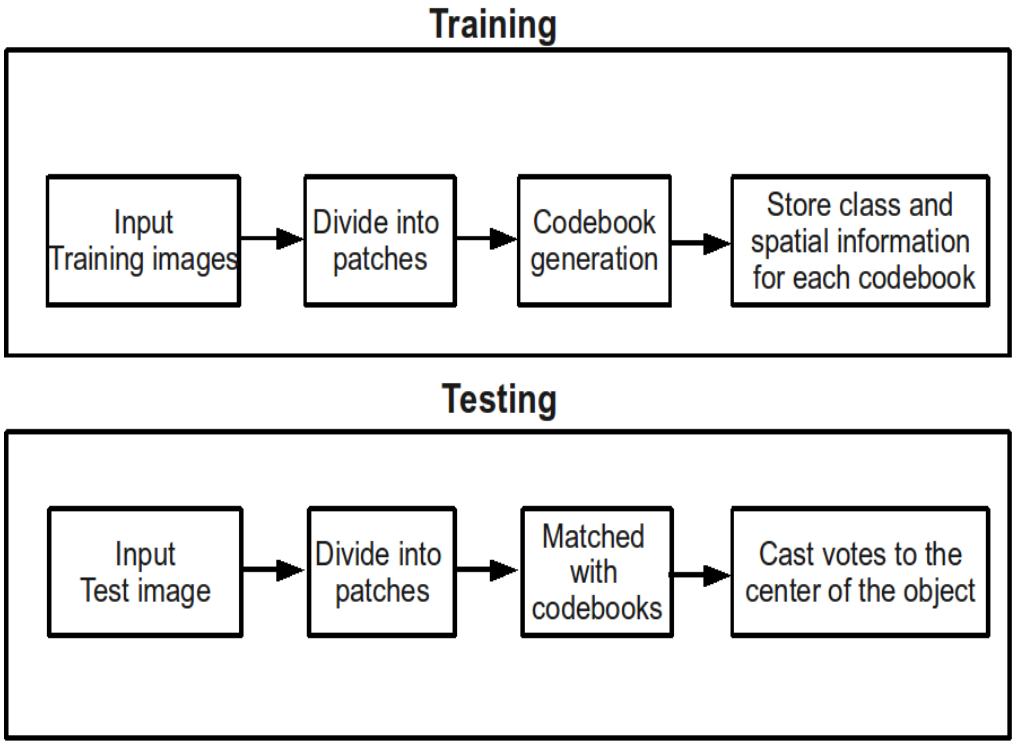


Figure A.1.: Block diagram for Training and Testing [38]

patches are grouped based on the visual appearance using a clustering algorithm. The similarity between the patches is measured using Normalized Grayscale Correlation and the clusters are represented by the cluster mean.

In order to reduce uncertainty due to nearest-neighbor assignment, a patch is compared against all codebooks and assigned to all the "activated" codebooks. A codebook is "activated" if the similarity between the patch and the cluster center is above a threshold t. This also results in increased robustness of the matching process. Traditional clustering algorithms such as k-means or Agglomerative clustering can be used to build these codebooks. The codebooks thus constructed can efficiently represent unseen parts of the object and also more robust to partial occlusion. However the procedures involved are highly computationally expensive. In order to efficiently model the object, a large number of generative codebooks are required. These codebooks are generated by applying computationally expensive clustering algorithms on a large number of training patches. Further, a patch is compared against a large number of codebooks during testing which is time consuming. Secondly, the codebook generation process follows a generative approach and does not utilize the class labels of the patches while training.

Discriminative codebooks can be constructed by making use of the class labels of the training patches.

A.2.2. Hough Forests

Gall *et al.* [38] introduced "Hough Forests" for object detection that learns a direct mapping between the appearance of the patch and its Hough vote. The mapping is accomplished within random forest framework and hence computationally efficient. Unlike [60], the codebooks are generated in a supervised manner by considering the label of the training patches. While Hough forests inherit the properties of general random forests, they also posses a few additional object specific properties [38]. They are:

- The set of leaf nodes of each tree can be regarded as a discriminative codebook. The information stored in each leaf node during training can be used to classify whether the patch belongs to the object or not and to predict the location of the center of the object with respect to the patch location.
- The trees are optimally constructed to reduce uncertainty in probabilistic votes towards leaves.
- Trees are constructed in a fully supervised manner using all the supervision available about the patch namely 1) the class label which specifies whether the patch is a positive or negative example 2) the offset that specifies the position of the centroid with reference to the patch.
- Similar to random forests, Hough forests can be used to train a large, high dimensional features without significant overfitting. For *e.g.*, the dimension of the features and the size of the training set is 8192 and 50000 respectively in [38]
- Matching a patch against a tree is logarithmic in the number of leaves. Consequently, the number of comparisons during testing is reduced making Hough forests efficient at runtime. This enables a dense sampling of the input image rather than only considering interest points.
- Hough forests can handle significant amount of noise and errors in the training data. Thus bounding box annotated data can be used as opposed to pixel-accurate segmentations.

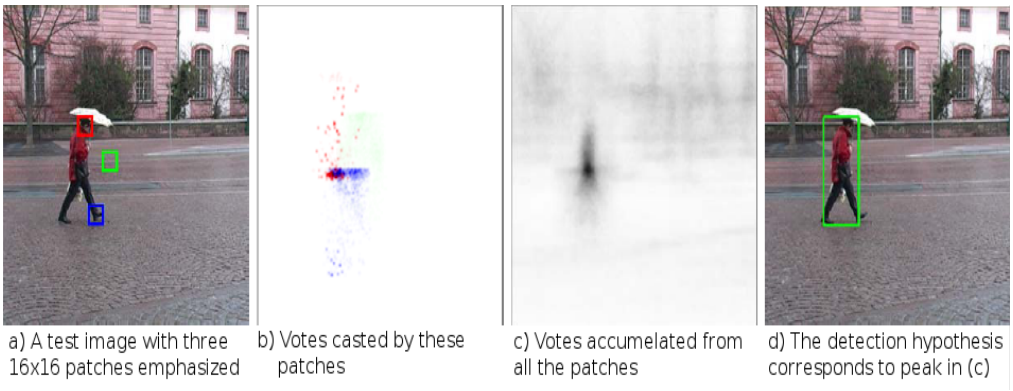


Figure A.2.: a) Three patches emphasized on a test image [38]: the patches with red and blue are sampled from object and green from background. b) Votes casted by these patches to the centroid of the object: votes from the object patches (red and blue) are prominent while votes from the background patch (green) are less prominent and scattered. c) Probabilistic Hough votes accumulated on Hough image. d) The hypothesis is detected by seeking the local maxima in the Hough image

Hough forests is an ensemble of random binary trees used for combined classification and regression of the patches. Each non-leaf node of the tree consist of a binary test that decides whether the entered patch should go to the left or right child node. This binary test is chosen during training stage based on the training patches arrived at that node. Based on the set of patches that ends up in a leaf node during training, a prior knowledge about the center of the object is computed and stored. During testing, a test image patch is passed through the trees until a leaf node is reached. A generalized Hough transform technique uses the information gathered at the leaf node during training along with the patch information to predict probable location of the object center. This is demonstrated in Fig. A.2. In what follows, we describe the training and testing procedures involved in the Hough forests technique.

Tree Construction

The computational complexity of clustering in [60] increases as the number of patches increases whereas Random forest can handle large amount of data with less computational complexity. Hence we uniformly sample the positive and negative training images to obtain the training patches. During training, a set of random trees are constructed using these training patches. The Hough forests in [38] is not invariant to scale. Hence the patches are extracted from the training images having the same object size. To handle scale variations, the test image is applied to the algorithm at different scales

to obtain Hough images. Finally, a 3D meanshift algorithm is applied on these set of Hough images to obtain local maxima which corresponds to the object location.

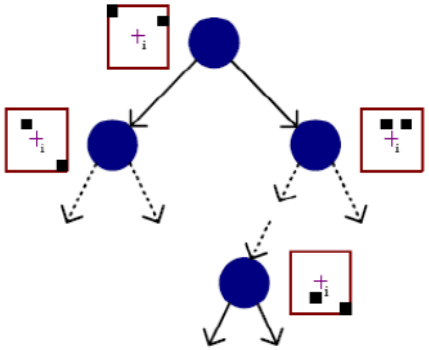


Figure A.3.: Tree construction [38]: At each node, two pixels of the patches are compared and the patch is passed to one of the child nodes based on the result of the comparison

Let $F = \{f^j\}_{j=1}^{M}$ denotes M feature channels of the patch P and $f^j \in \mathbb{R}^M$ is the feature vector of the j^{th} feature channel (e.g. for color images, j = 1 .. 3 are indexes to the color components). Let us denote with $c \in \{0, 1\}$ the class label, that is background or foreground and $\mathbf{d} \in \mathbb{R}^2$, the offset between the center of the bounding box and the center of the patch. Once a tree is constructed, any patch P can be propagated through it and end upto one of its leaves, following the path from the root to the leaves according to a test that takes place at each node. The tree expands by performing several random tests at each node and passing the patches to the child nodes based on the result of the tests. The test t for a patch P is given by

$$t(P) = \begin{cases} 0 & if \quad f^j(\mathbf{p}) < f^j(\mathbf{q}) + \tau, \\ \\ 1 & otherwise \end{cases} \tag{A.1}$$

where $\mathbf{p}$, $\mathbf{q}$ are the coordinates of the randomly chosen indexes in the feature channel f^j, $1 \leq j \leq M$ and τ is a randomly chosen threshold.

The training is performed in a supervised manner by utilizing the class label and offset information of the patches. The leaves of the trees make following predictions about the patches that end up in it. Firstly, the probability of the patch being an object patch and secondly the probable locations of the center of the object in the Hough image. In order to have a good decision about the object center with less ambiguity, the main

focus of the tree construction is to have leaves with minimum uncertainties about the class and offset information of the patch. Once constructed, each leaf node L stores the class information C_L and the offset information $D_L = \{\mathbf{d}\}$ of the patches that end upto it. The class information C_L is the proportion of the object patches (i.e patches with label 1) and D_L is the set of offsets for the patches that end up to the leaf node. Each non-leaf node is assigned a test by choosing the best test from a pool of tests.

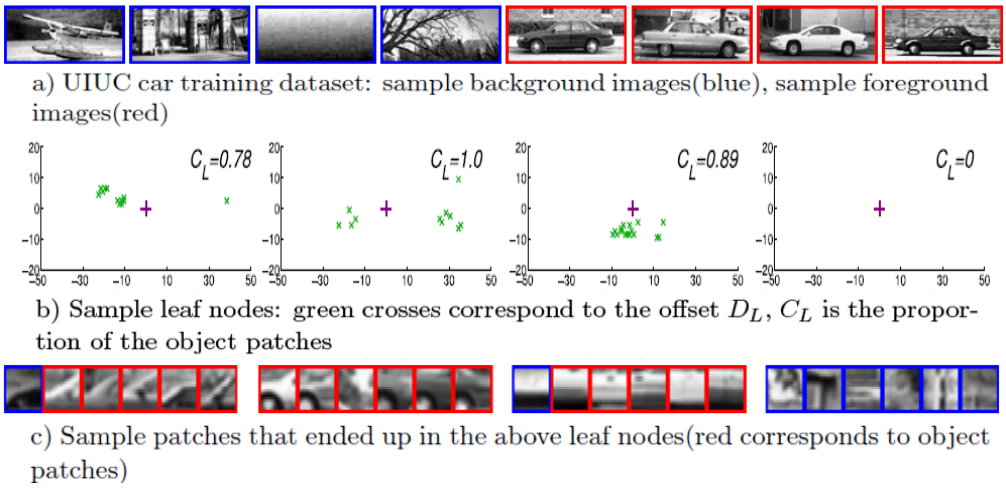


Figure A.4.: a) Bounding box of the positive and negative training images [38]. b) The offsets are clustered at the leaves thereby reducing the uncertainties in the casted votes, the leaf node in the last column has no positive patches and hence $C_L = 0$. c) Patches collected at each leaf node: patches having same appearance are grouped together

The tests are chosen such that the uncertainties in class labels and offsets are reduced towards leaves. This is accomplished by using two measures namely the class uncertainty E_c and the offset uncertainty E_o. The class uncertainty measure at a node N is given by

$$E_c(N) = -|N|\big(C_N \log(C_N) + (1 - C_N) \log(1 - C_N)\big) \tag{A.2}$$

where C_N is the proportion of object patches at the node N. The offset uncertainty measure is given by

$$E_o(N) = \sum_{\mathbf{d}_i \in D_N} (\mathbf{d}_i - \overline{\mathbf{d}})^2, \tag{A.3}$$

where $\overline{\mathbf{d}}$ is the mean of the offset vectors in D_N. The negative patches corresponding to the background images are ignored here. At each node, the values of $\mathbf{p}$, $\mathbf{q}$ and τ are

randomly chosen for each test and the patches are passed to child nodes according to the test in Eq. A.1. The best test among the K tests is chosen as

$$k = \arg\min_{k} \left[E_*(N_l^k) + E_*(N_r^k) \right] \tag{A.4}$$

where N_l^k and N_r^k are the child nodes for the k^{th} test and $*$ is chosen randomly between c or o at each node. In general c or o is chosen with equal probability unless the number of negative patches is less than 5% in which case node is chosen to minimize offset uncertainty. The construction of a tree stops when either the depth of the tree reaches D_{max} or the number of patches in a node drops below a certain threshold. By interleaving between the class uncertainty and offset uncertainty, the tree construction procedure ensures that the patches in a leaf node have less variations in both class labels and offsets.

A.2.3. Object Detection using Hough Forests

In this section, we describe the application of Hough forests for localization of an object. Let $W \times H$ be the size of the bounding box which is assumed to be fixed during training as well as testing. Hence the parameter that describes the bounding box is the centroid of the bounding box.

Let $P(\mathbf{y})$ be a patch from the test image with center $\mathbf{y}$ and feature vector $F_\mathbf{y}$. Let $c_\mathbf{y}$(unknown variable) be the class label of the patch which specifies whether the patch lies inside the object or not. Let $E(\mathbf{x})$ denote a random event corresponding to the existence of the object center at location $\mathbf{x}$ in the Hough image. The task is to determine the probabilistic evidence $p(E(\mathbf{x})|F_\mathbf{y})$ that a patch with appearance $F_\mathbf{y}$ brings about the object center at location $\mathbf{x}$. Let $B(\mathbf{x})$ denote a bounding box with center $\mathbf{x}$. Since we consider only the votes from the object patches, we are interested in $y \in B(\mathbf{x})$, *i.e.*, the patches within the bounding box. Hence

$$\begin{aligned} p\big(E(\mathbf{x})|F_\mathbf{y}\big) = p\big(E(\mathbf{x}), c_\mathbf{y} = 1|F_\mathbf{y}\big) = \\ p\big(E(\mathbf{x})|c_\mathbf{y} = 1, F_\mathbf{y}\big).p\big(c_\mathbf{y} = 1|F_\mathbf{y}\big) = \\ p\big(\mathbf{d}_\mathbf{y} = \mathbf{y} - \mathbf{x}|c_\mathbf{y} = 1, F_\mathbf{y}\big).p\big(c_\mathbf{y} = 1|F_\mathbf{y}\big) \end{aligned} \tag{A.5}$$

where $\mathbf{d}_\mathbf{y}$ is the estimated offset for the patch $P(\mathbf{y})$. The first term in Eq. A.6 corresponds to the location of the vote and the second term specifies the confidence of the vote.

Both the values for a patch can be computed by passing the patch through the tree to determine the leaf node and using the information corresponding to that leaf node computed during training.

Let us assume that the patch $P(\mathbf{y})$ ends up in a leaf node L for a tree T and C_L, D_L be the class and offset information for the leaf stored during training. The first term can be computed using Parzen-window estimate based on the offset vectors D_L. The second term can be approximated by C_L. Intuitively, if the patch reaches a leaf node with more number of object patches than background ($C_L > 0.5$), then the patch casts votes with high confidence. For a single tree T, the probability estimate $p\big(E(\mathbf{x})|F_\mathbf{y}; T\big)$ is given by,

$$p\bigg(E(\mathbf{x})|F_\mathbf{y}; T_k\bigg) = \bigg[\frac{1}{|D_L|}\sum_{\mathbf{d}\in D_L}\frac{1}{2\pi\sigma^2}exp\bigg(-\frac{\|(\mathbf{y}-\mathbf{x})-\mathbf{d}\|^2}{2\sigma^2}\bigg)\bigg]C_L \tag{A.6}$$

where $\sigma^2 I$ is the covariance of the Gaussian Parzen window.

Since the Random forests make decisions with an ensemble of trees, a similar approach is followed in Hough forests. The predictions for the location of the object centers are gathered from all K trees by passing the patch through each tree and computing the probabilistic votes $p\big(E(\mathbf{x})|F_\mathbf{y}; T_k\big)$ as given in Eq. A.6. The probabilistic vote for the entire forest $\{T_k\}_{k=1}^K$ is computed by averaging the probabilistic votes for the individual trees, *i.e.*,

$$p\big(E(\mathbf{x})|F_\mathbf{y}; \{T_k\}_{k=1}^K\big) = \frac{1}{K}\sum_{k=1}^{K} p\big(E(\mathbf{x})|F_\mathbf{y}; T_k\big) \tag{A.7}$$

The above Equation provides the probabilistic vote for a single patch. The combined vote at a location $\mathbf{x}$ of a Hough image V is computed by accumulating the votes obtained from all the patches within the bounding box $B(\mathbf{x})$,

$$V(\mathbf{x}) = \sum_{\mathbf{y}\in B(\mathbf{x})} p\big(E(\mathbf{x})|F_\mathbf{y}; \{T_k\}_{k=1}^K\big) \tag{A.8}$$

The Hough image $V(\mathbf{x})$ values serves as the confidence measure for the prediction of the hypothesis. The steps mentioned in Eq. (A.6, A.7, A.8) are computationally expensive. Instead, the Hough image can be computed by going through each pixel location $\mathbf{y}$, passing the appearance of the patch $P(\mathbf{y})$ through each tree of the forest and adding a value $\frac{C_L}{|D_L|}$ to all pixels at location $\{\mathbf{y}-\mathbf{d}|\mathbf{d}\in D_L\}$. This is followed by a Gaussian filtering to obtain the final Hough image. Then a Mean shift algorithm

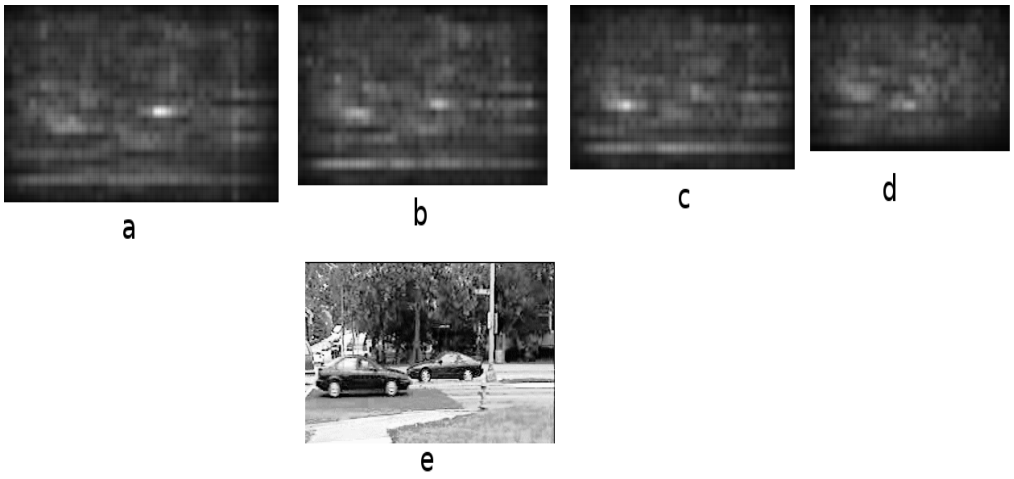


Figure A.5.: Figures a,b,c and d show the Hough images obtained at different scales a) s = 1.1, b) s = 1, c) s = 0.9, d) s = 0.8, e) Input test image

is applied on the Hough image to locate the maxima which correspond to the object hypothesis.

Since the algorithm is trained with a single scale, it can detect the objects of the same size in the test image. To handle multiple scales, the input test image is applied to the algorithm at multiple scales s. For each scale, the corresponding Hough image V^s is obtained. These Hough images are resized to the original size and a Gaussian filtering is applied in the third(scale) dimension. A 3D mean shift algorithm is then employed to find the maxima which returns the hypothesis $\big(\mathbf{x}, s, V(\mathbf{x})\big)$. The location and size of the bounding box for this hypothesis is given by $\frac{\mathbf{x}}{s}$ and $\frac{W}{s} \times \frac{H}{s}$.

A.3. A Discriminative Voting Scheme for Object Detection

In this work, we propose an offset uncertainty measure that is defined on the Hough images of the training set. This is in contrast to the Hough forest algorithms in [32, 38] that use a classical regression tree measure, namely the RSS (Residual Sum of Squares) on the leaf nodes, to compute offset uncertainty [116]. In our method, we compute Hough spaces for all the training images when evaluating how good the test split is. The Hough voting spaces are incrementally constructed when building the tree by an algorithm that has a slight overhead (10%) in comparison to the classical one. Our objective is to discriminate the location of the object centers from the background. In

order to achieve that we choose the test that has the maximum ratio of votes at objects' center to votes at other locations. The main contribution of the work is a framework for choosing an optimal test which discriminates the object from background at each node. With this, we achieve application specific Hough forests for object detection that can outperform the general Hough forest technique. The framework also allows us to incorporate different evaluation criterion for training according to the desired output.

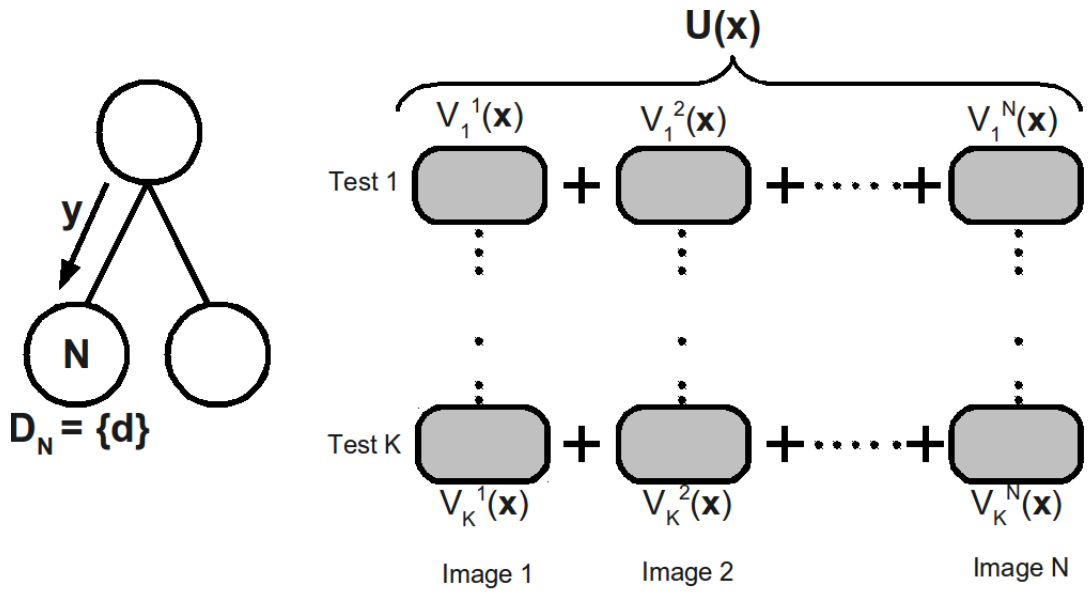


Figure A.6.: The patch $P(\mathbf{y}^j)$ with center $\mathbf{y}$ from the j^{th} training image arriving at a node N casts votes to the j^{th} Hough image $V^j(\mathbf{x})$ at locations $\mathbf{x} = \mathbf{y} - \mathbf{d}$ where $\mathbf{d} \in D_N$

The Hough forest technique for object detection [38] employs a set of trees which basically perform two tasks namely, 1) classification: where the patches are classified into object and non-object patches and 2) regression: which predicts the location of the center of the object with respect to the patch. During training, the trees are constructed to reduce the class label uncertainty (classification) and offset uncertainty (regression) towards leaves. In this section we first discuss the construction of intermediate Hough spaces and propose a new offset uncertainty criteria for choosing the test at a node based on the constructed Hough spaces. The method in [38] follows a classical tree construction approach by calculating the variance in the child nodes for the computation of the offset uncertainty measure. This approach tends to reduce the impurity of the offset vectors towards the leaf nodes but it does not consider the actual Hough space. Since the object center location is known for training set images, it can be utilized to verify the predicted object centers which in turn, can be used to evaluate the quality of the tests during training. In this work, we propose an offset uncertainty measure by determining

intermediate Hough spaces for training images at each node for each test and choose a test that maximizes the ratio of the votes at the center of the objects over the votes at other locations.

A.3.1. Computation of Intermediate Hough Spaces

The value at a pixel $\mathbf{x}$ of the Hough image is computed accumulating the votes from all the patches in the original image and averaging over all the trees. To compute the Hough image, the whole procedure needs to be repeated for all the pixels in the Hough image, a procedure that is computationally expensive. Alternatively, the Hough space can also be computed by passing a patch $P(\mathbf{y})$ through the trees to determine a leaf node L and adding $\frac{C_L}{|D_L|}$ to the location $\{\mathbf{y} - \mathbf{d} | \mathbf{d} \in D_L\}$. Then smoothing with a Gaussian kernel gives the result of Eq. A.6.

Here we compute intermediate Hough spaces at each node. More specifically, for each non-leaf node N, we calculate the parameters C_N and D_N where C_N is the proportion of the object patches in N and $D_N = \{\mathbf{d}\}$ is the set of offsets in N. The object patches arriving at node N cast votes to the Hough space using the set of offsets in N. This is demonstrated in Fig. A.6. Let $V_i^j(\mathbf{x})$ denote the intermediate Hough space for the j^{th} training image and i denote the test index. Let $P(\mathbf{y}^j)$ be a patch from the j^{th} training image arriving at node N. This patch casts votes to the j^{th} Hough image $V^j(\mathbf{x})$ at locations $\mathbf{x} = \mathbf{y} - \mathbf{d}$ where $\mathbf{d} \in D_N$. This procedure is repeated for all the patches arriving at N.

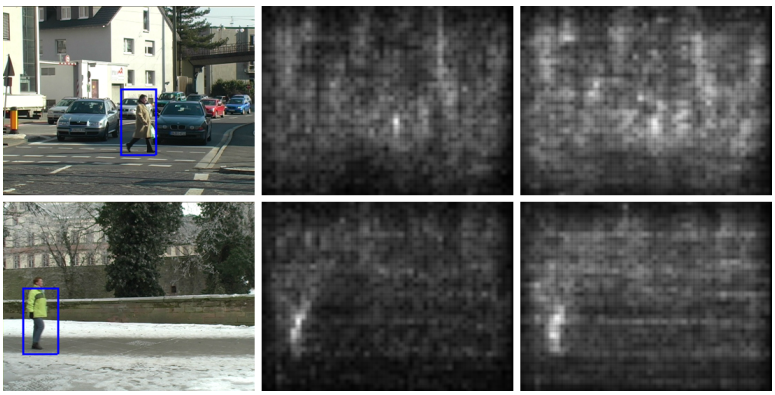


Figure A.7.: Output image (first), Hough space for the proposed method (middle)[max values: 0.289, 0.299], Hough space for [38] (last) [max values: 0.222, 0.222]

A.3.2. Proposed Offset Uncertainty Criteria

The Hough forest tree construction process is governed by Eq. A.2 and Eq. A.3. The class uncertainty criteria attempts to reduce the uncertainties in the class labels towards leaves. The offset uncertainty criteria attempts to reduce the uncertainty in the location of the casted votes. In this work, we propose an offset uncertainty criteria which attempts to maximize the votes at the center of the object locations in the combined Hough image while suppressing responses at other locations. The combined Hough votes for the k^{th} test are computed by considering the votes from the intermediate Hough spaces of all the training images, $\sum_{\mathbf{x}} V_k^j(\mathbf{x})$.

The objective is to discriminate the location of the object from the background. Let $\mathbf{x}_c^j$ be the location of the center of the object in the j^{th} Hough image. Let $\mathbf{X}_c^j$ denote an area with few pixels around the center $\mathbf{x}_c^j$ of the object. The proposed offset uncertainty criteria E_{ho} is then given by

$$E_{ho} = \frac{\sum_{j=1}^{T} \sum_{\mathbf{x} \in \mathbf{X}_c^j} V(\mathbf{x})}{\sum_{j=1}^{T} \sum_{\mathbf{x} \neq \mathbf{X}_c^j} V(\mathbf{x})} \tag{A.9}$$

where at the denominator is the accumulated votes at other parts of all the training images. The above uncertainty criteria is computed for the left and right child nodes of a node and the test that maximizes E_{ho} is chosen at the node under consideration. More specifically, we use the class label uncertainty (Eq. A.2) along with the proposed offset uncertainty criterion (Eq. A.9) to construct the trees. At each node, a test is chosen from a pool of tests by evaluating the criterion

$$T_1 : \qquad \arg\min_k \left[E_c(N_l^k) + E_c(N_r^k) \right] \tag{A.10}$$

or

$$T_2 : \qquad \arg\max_k \left[E_{ho}(N_l^k) + E_{ho}(N_r^k) \right] \tag{A.11}$$

where N_l^k and N_r^k are the left and right child nodes for k^{th} test and a test T_1 or T_2 is chosen randomly. Similar to the approach discussed in Section A.2.2, during testing we detect objects at local maxima of the Hough image that are above a threshold.

The proposed uncertainty criterion is based on the response in the output space. Further, additional object specific constraints can be imposed to evaluate the tests thereby

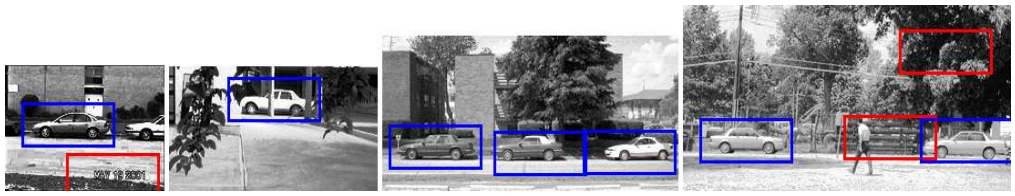
improving the performance of the forest. In contrast to the classical offset uncertainty measure (Eq. A.3) which ignores the class information, the proposed uncertainty measure utilizes the class information C_N at a node to compute the intermediate Hough space. Since the centers and offsets of the patches for the training images are known apriori, the location of the resulting votes can be pre-computed. This reduces the computational cost during training and significantly improves the tree construction speed.

The proposed method computes the Hough space for each test at each node and hence can be computationally expensive. In order to reduce the computational complexity, we observe that the training patches arriving at a node in question cast votes using the offsets of the training patches at the node in question. Since the patch centers and offsets for training patches are known apriori, the voting location for each patch-offset combination can be pre-computed. By pre-computing the voting locations, the computation time is significantly reduced. The proposed method requires 10% more computation time as compared to the classical Hough forests.

A.3.3. Experimental Results

We evaluated the performance of the proposed method on three datasets, namely the UIUC-cars, the TUD-pedestrians and the INRIA-pedestrians dataset. In all cases we used 25000 positive and 25000 negative patches for training. We set the maximum depth of the trees to 20 and the threshold for the number of examples/patches in a leaf node to 20. This means that a node is not split if either the maximum depth is reached, or if the number of examples/patches in the node in question are below the threshold. During tree construction, we evaluated 1500 tests at each node and in each case selected the one that minimizes the proposed criterion.

We constructed 15 trees for each forest in each dataset and followed a boosting approach to learn hard examples, in which the training examples/patches for a given tree are selected based on the classification result that is obtained using the previously constructed trees. More specifically, hard examples (i.e. misclassified patches) are duplicated in the initial training set. We repeat this procedure every other five trees. For all datasets, we consider the size of the bounding box fixed and deal with scale variations by resizing each test image to a number of scales (4 in our experiments), each one of which we pass through the forest in order to obtain a spatial Hough space. Then, a meanshift algorithm is applied in the 3D scalespace in order to obtain hypotheses about the location and scale of the objects.



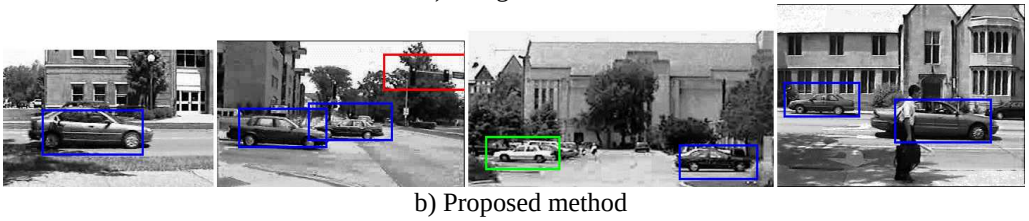
a) Hough forest

b) Proposed method

Figure A.8.: Detected objects (blue), miss detection (green), false positives (red) on *UIUC-Single* car dataset: a) Hough forest. b) Proposed method

For the TUD and INRIA datasets, we used 16 feature channels namely the RGB color channels, the magnitude of the horizontal and vertical gradients, the magnitude of second-order derivatives in both horizontal and vertical directions, and the 9 HOG channel descriptors. The HOG descriptors were obtained by accumulating the normalized magnitude in 9 orientation directions computed over a 5×5 window centered around each pixel. To handle variations in clothing, illumination, articulation of parts, the channels were filtered with a max-filter on 5×5 windows centered around a pixel.

a) Hough forest



b) Proposed method

Figure A.9.: Detected objects (blue), miss detection (green), false positives (red) on *UIUC-Multi* car dataset: a) Hough forest. b) Proposed method

We first present results on the UIUC-cars dataset. The training set for this dataset consists of 550 positive examples of images depicting side views of cars and 500 negative images each of size 100×40 pixels. To construct the trees we used 400 positive and 400 negative images from the dataset. We used the three feature channels, namely the

gray level values and the absolute values of the horizontal and the vertical gradients. The test set consists of 170 images of the objects with same scale. A location output by the algorithm is counted as a correct detection if it lies within an ellipse with center at the true location and axes 25% of the object dimensions in each direction. In addition, only one detection per object is allowed - if two or more detected windows satisfy the above criteria for the same object, only one is counted as correct; the others are counted as false detections. The proposed method achieves 98.5% Equal Error Rate (EER) for the $UIUCSingle$, and therefore performs better than our implementation of the Hough forest [38] which achieves 98% EER for $UIUCSingle$ and is in par with the state-of-the-art methods which achieve the same EER Lampert *et al.* [52]. Fig. A.8 compares the performance of the Hough forest [38] and the proposed method on $UIUCSingle$ car dataset.

We also tested the algorithm on $UIUCMulti$ that contains cars at multiple scales. To handle scale variations, we resized the original image into different scales s and applied the algorithm on these resized images. We then combine these Hough images uisng $3d$-meanshift algorithm where the third dimension is in the scale space. We have considered seven values for s (0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1). The test set $UIUCMulti$ consists of 107 cars at different scales. A location-scale pair output by the algorithm is counted as a correct detection if it lies within an ellipsoid with center at the true location-scale and axes 25% of the true object dimensions in each direction. The proposed method achieves nearly same performance as Hough forests with 98% at Equal Error Rate (EER) for the $UIUCMulti$. In Fig. A.9, we compare the performance of proposed method and the Hough forest technique [38].

For the same dataset, we demonstrate the effect of width of the area around the ground truth location of the object center of the intermediate Hough spaces $\mathbf{x}_c$ on the final Hough image. $\mathbf{X}_c$ in Eq. A.9 denotes an area of a few pixels around the actual center of the object in question. In order to do so, we constructed Hough forests by varying the size of the area X_c. In Fig. A.10 we present the final Hough images for $\|X_{c_1}\| < \|X_{c_2}\| < \|X_{c_3}\|$ for three test images. It can be seen in the figure that the Hough image that are obtained using a small area size (i.e. $\|X_{c_1}\|$) produces high responses at the object locations in the final Hough image and low responses (i.e. small ambiguity/spread) at other locations. As the area increases from $\|\mathbf{X}_{c_1}\| = 52$ to $\|\mathbf{X}_{c_3}\|$ the response at the object location decrease and the ambiguity in the location of the center increases. The average values of the maximum responses over all training set images are $0.0192, 0.0188$ and 0.0184 for $\mathbf{X}_{c_1}$, $\mathbf{X}_{c_2}$ and $\mathbf{X}_{c_3}$ respectively while the average responses at other areas

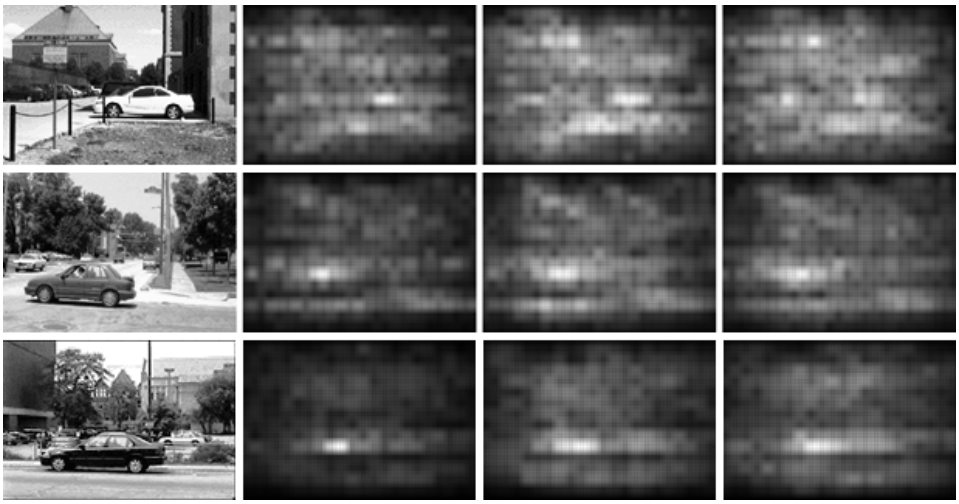
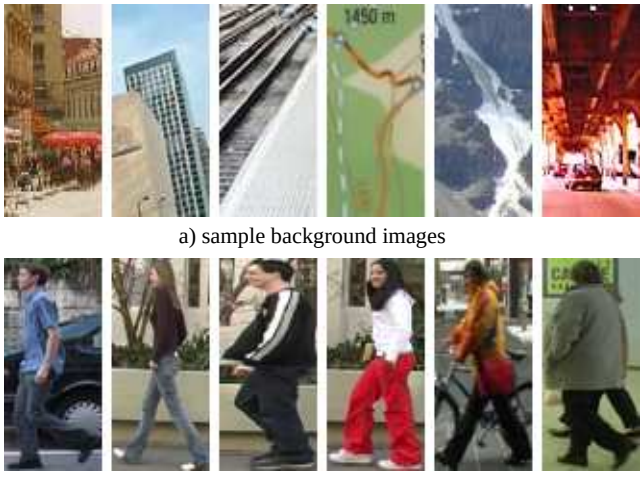


Figure A.10.: First column: Test images. Second, third and fourth columns: Hough image for $\|\mathbf{X}_{c1}\| = 52$, $\|\mathbf{X}_{c2}\| = 370$ and $\|\mathbf{X}_{c3}\| = 862$ respectively

are $0.0061, 0.0063$ and 0.0065 respectively. In all of our experiments we choose $\|\mathbf{X}_{c1}\| = 52$, a value that defines an area equal to the 13/100 of the size of the object bounding box.

Average values	X_{c1}	X_{c2}	X_{c3}
Inside X_c	0.0192	0.0188	0.0184
Other locations	0.0061	0.0063	0.0065



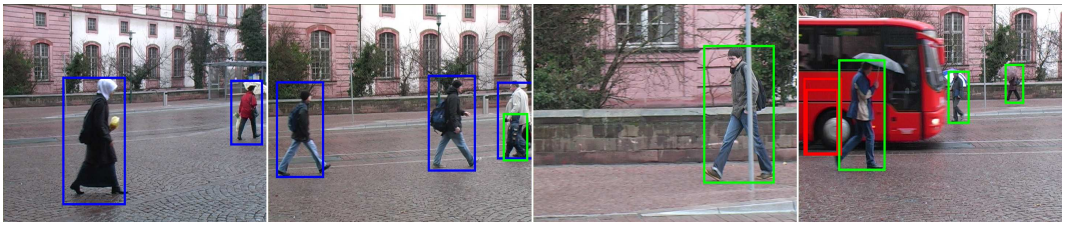
a) sample background images

b) sample object bounding boxes

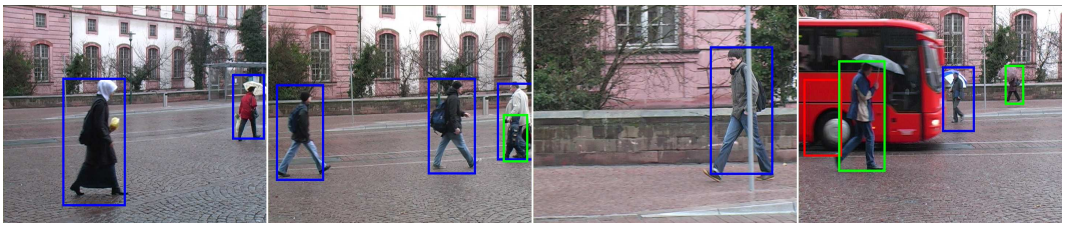
Figure A.11.: Sample training images (100×51) of TUD-pedestrian dataset: a) background bounding boxes extracted from INRIA training set, b) object bounding boxes

We then demonstrate the performance of the algorithm on the more challenging TUD-pedestrian dataset [8]. The training set for this dataset consists of 400 images with

pedestrians. We trained the Hough forest using positive examples that were constructed by extracting the object bounding boxes and resizing them to fixed dimensions of size 100×51. We constructed 400 negative examples/images by randomly sampling patches from the background areas in the images. As the diversities of the background were low, we combined it with background images of INRIA pedestrian dataset. The test set consists of 250 images containing 311 fully visible people who exhibit significant variation in clothing and articulation. The evaluation criteria used to measured the detection quality are, cover, overlap and relative distance. Cover and overlap measure how much the ground truth bounding box is covered by the detected bounding box and vice versa. Relative distance measures the distance between the center of the bounding boxes. We inscribe an ellipse in the ground truth bounding box and relate the measured distance to the radious of the ellipse at the corresponding angle [102]. In our experiments, only hypothesis that have cover and overlap more that 50% and relative distance less that 0.5 are accepted as the correct detection.



a) Hough forest



b) Proposed method

Figure A.12.: Detected objects (blue), miss detection (green), false positives (red) on TUD pedestrian dataset: a) Hough forest. b) Proposed method

Fig. A.7 depicts the Hough space obtained with our own implementation of the approach in [38] and of the proposed method. It is clear that with the proposed method the number of votes at the center are significantly higher than the number of votes at the background a fact that indicates higher discrimination capability. Fig. A.14 compares quantitatively the performance of the proposed method and the Hough forest algorithm [38]. The performance curves were generated by changing the acceptance threshold on the hypotheses vote strength $V(\mathbf{x})$. When generating recall-precision curves, we rejected the detection hypotheses with centroids inside the bounding boxes detected with higher

confidence in order to avoid multiple detections of the same instance. We notice that, the proposed method clearly outperforms [38], at high precision values and exhibits similar performance elsewhere. In particular, the proposed method shows 7% improvement over Hough forests at a precision value of 0.93. From the recall-precision curve, it is evident that the proposed method outperforms the Hough forest algorithm for the TUD -pedestrian dataset.

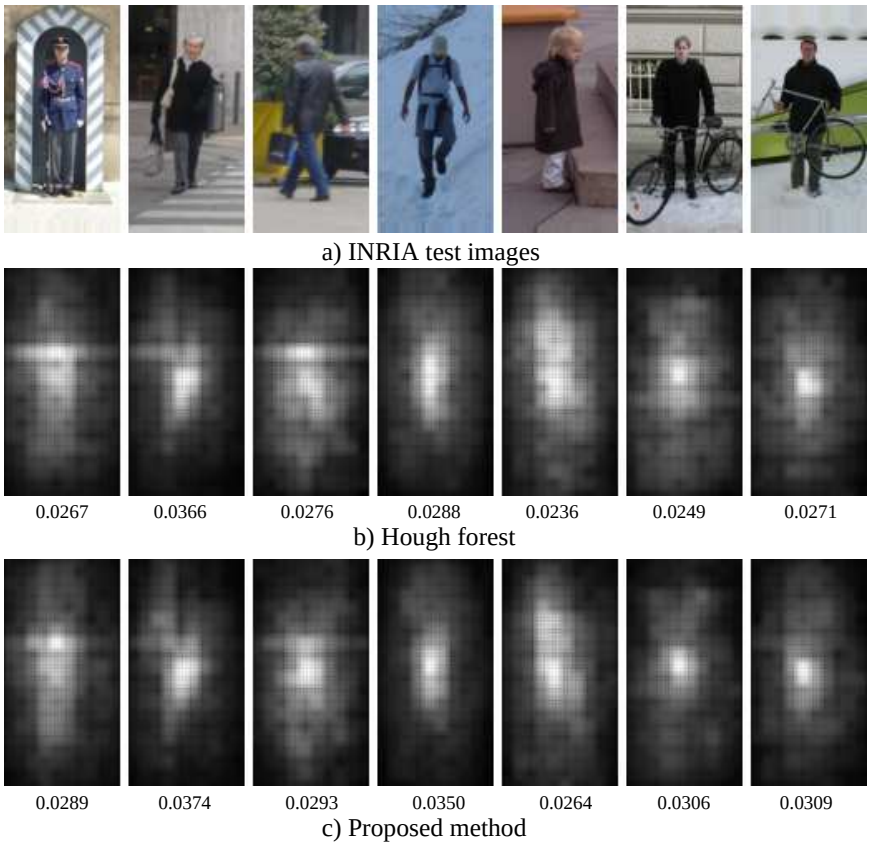


Figure A.13.: Detected objects (blue), miss detection (green), false positives (red) on INRIA pedestrian dataset. The values indicate the maximum values of the Hough image. a) Hough forest. b) Proposed method

We next demonstrate the performance of the algorithm on INRIA-pedestrian dataset [26]. We used 2416 positive and 2416 negative images for training. The negative images were obtained by sampling the person-free training images. The test set consists of 288 cropped and pre-scaled images with objects and 453 images without the object. Fig. A.13 shows the Hough images obtained for the images from INRIA dataset. We can see that the Hough images obtained using the proposed method has votes more clustered at the center compared to the Hough images with [38]. Further, the maximum values of the Hough images indicate that the proposed method produces Hough images with stronger

response at the object location as compared to the same with Hough forests. Fig. A.14 shows the recall vs False Positives per Window curves for the proposed and Hough forest algorithms. It is clear that the propose method outperforms the classical Hough forests particularly at low false positives per window.

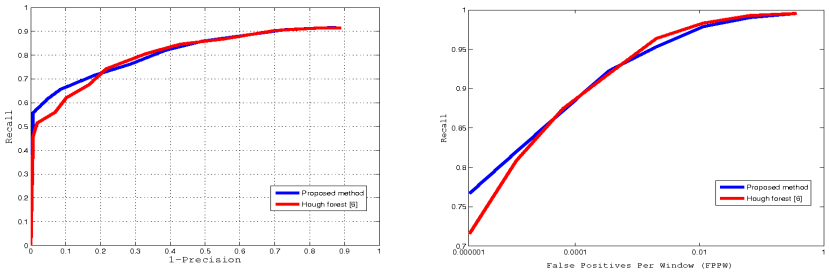


Figure A.14.: Recall-precision curves for TUD (first column) and INRIA (second column)

In order to apply the algorithm for localization of actions in video sequences, we used the video sequences from the KTH dataset. The dataset contains six categories of actions: boxing, clapping, jogging, running, walking and waving . There were 25 subjects performing each action four times in four different environments, resulting in about 600 video sequences in total. It is relatively simple to localize actions in three actions namely boxing, clapping and waving as the subject is stationary on for all the frames of the sequence. We demonstrate the performance of the algorithm on actions running and jogging. We apply the algorithm at six scales in order to handle changes in size of the subject. The model is learnt using INRIA pedestrian dataset. The algorithm is applied on each frame of the video sequence at six different scales. The final hypothesis is obtained by finding the peak response in the Hough image using meanshift algorithm. Fig. A.15, and Fig. A.16 shows the results on running and jogging actions respectively.

A.4. Conclusions

We introduced a novel framework for object detection using Hough forests that uses the knowledge of the objects center locations in training images to efficiently construct the trees. During training, we compute at each node the Hough space for each of the training images and use this Hough space to evaluate the tests at the node in question. The test with maximum ratio of votes at the center to the other parts of the combined Hough space is chosen as the candidate test for that node. In that way, we obtain Hough spaces that succesfully discriminate the object from the background. We demonstrated

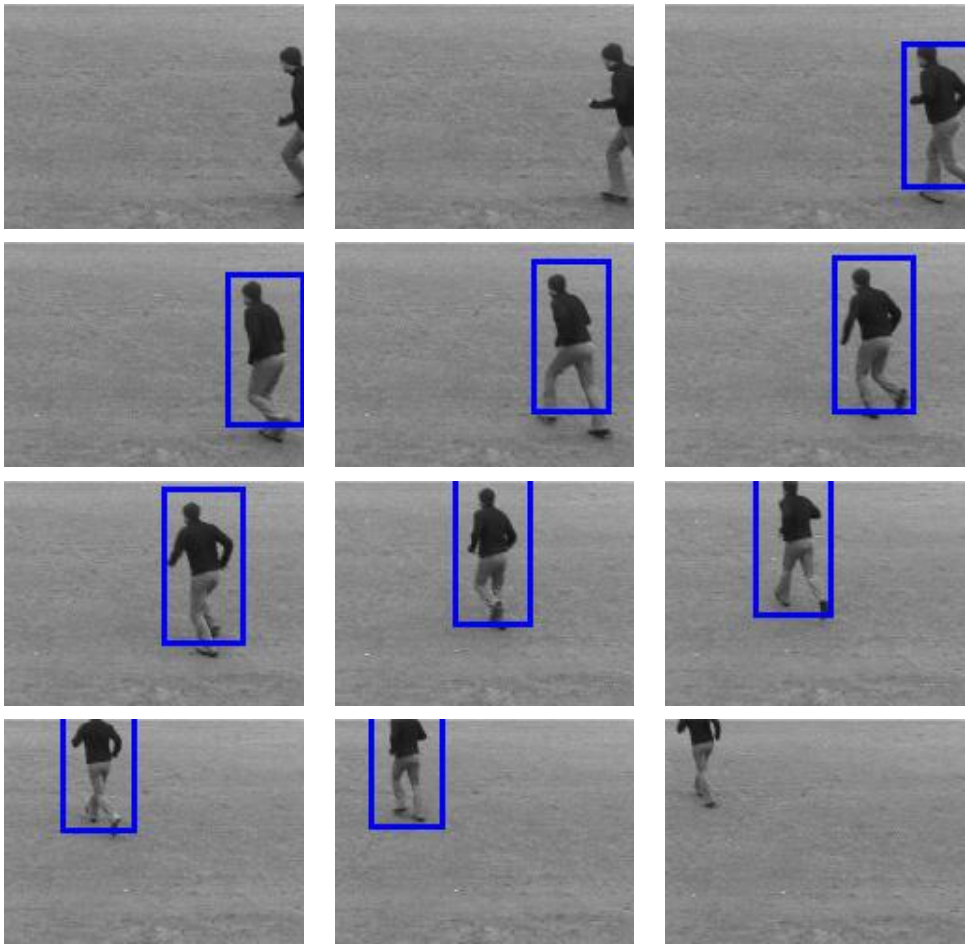


Figure A.15.: Performance of the algorithm on KTH running sequence

Figure A.16.: Performance of the algorithm on KTH jogging sequence

the efficiency of the proposed algorithm through several experiments. We tested the novel framework on standard datasets and have experimentally shown that it outperforms the classical Hough forests.

Appendix B.

Proof of Convergence of the Iterative Optimization Procedure

Here, we provide a proof of convergence for the proposed algorithms. More precisely, the iterative optimization method used, also known as alterative projections, never increases the value of Eq. 3.8 between two successive iterations, as it can be regarded to be a monotonic function (see also Fig. 1a). We define a continuous function of the form:

$$D : \{\mathbf{G}, \mathbf{H}, \mathbf{w}, \xi_i, b\} \times \mathbb{R} \rightarrow \mathbb{R} \tag{B.1}$$

where $\mathbf{G} \in \mathbb{G} \subset \mathbb{R}^{m \times k}$, $\mathbf{H} \in \mathbb{H} \subset \mathbb{R}_{+}^{k \times n}$ and $\mathbf{w} \in \mathbb{W} \subset \mathbb{R}^{m}$. We define the following three functions

$$\begin{aligned} D_1 &: \mathbb{G} \times \mathbb{R} \rightarrow \mathbb{R} \\ D_2 &: \mathbb{W} \times \mathbb{R} \rightarrow \mathbb{R} \\ D_3 &: \mathbb{H} \rightarrow \mathbb{R} \end{aligned} \tag{B.2}$$

defined as $D_1(\mathbf{G}, b) = D(\mathbf{G}, \xi_i, b; \mathbf{H}, \mathbf{w})$ (i.e, acquired fixing $\mathbf{H}$ and $\mathbf{w}$) and $D_2(\mathbf{w}, b) = D(\mathbf{w}, \xi_i, b; \mathbf{G}, \mathbf{H})$, (i.e, acquired fixing $\mathbf{G}$ and $\mathbf{H}$) and $D_3(\mathbf{H}) = D(\mathbf{H}; \mathbf{G}, \mathbf{w}, \xi_i, b)$, (i.e,

acquired fixing $\mathbf{G}$, $\mathbf{w}$, ξ_i and b) . By definition the function D has 3 mappings:

$$g_1(\mathbf{G}^*, b^*) \triangleq \arg\min_{\mathbf{G},b} D_1(\mathbf{G}, b) \tag{B.3}$$

$$g_2(\mathbf{w}^*, b^*) \triangleq \arg\min_{\mathbf{w},b} D_2(\mathbf{w}, b) \tag{B.4}$$

$$g_3(\mathbf{H}) \triangleq \arg\min_{\mathbf{H}} D_3(\mathbf{H}) \tag{B.5}$$

and $*$ denotes optimality.

The sequence of produced solutions are characterized by the following relationships:

$$\begin{aligned} D_1(\mathbf{G}^*, b^*) &\geq D_1(\mathbf{G}, b) \\ D_2(\mathbf{w}^*, b^*) &\geq D_2(\mathbf{w}, b) \\ D_3(\mathbf{H}^*) &\geq D_3(\mathbf{H}). \end{aligned} \tag{B.6}$$

Given an initial estimate $\{\mathbf{G}_0, \mathbf{H}_0, \mathbf{w}_0, b_0\}$, the proposed algorithm generates a sequence of solutions $\{\mathbf{G}_{(t)}, \mathbf{H}_{(t)}, \mathbf{w}_{(t)}, b_{(t)}\}$ via

$$g_1(\mathbf{G}^*_{(t)}, b^*_{(t)}) \triangleq \arg\min_{\mathbf{G},b} D_1(\mathbf{G}_{(t)}, b_{(t)}) \tag{B.7}$$

$$g_2(\mathbf{w}^*_{(t)}, b^*_{(t)}) \triangleq \arg\min_{\mathbf{w},b} D_2(\mathbf{w}_{(t)}, b_{(t)}) \tag{B.8}$$

$$g_3(\mathbf{H}^*_{(t)}) \triangleq \arg\min_{\mathbf{H}} D_3(\mathbf{H}_{(t)}). \tag{B.9}$$

The sequence of produced solutions are characterized by the following relationships:

$$\begin{aligned} a_1 &= g_1(\mathbf{G}^*_{(1)}, b^*_{(1,1)}) \\ &\geq g_2(\mathbf{w}^*_{(1)}, b^*_{(1,2)}) \\ &\geq g_3(\mathbf{H}^*_{(1)}) \\ &\geq \cdots \geq \\ &\geq g_1(\mathbf{G}^*_{(t)}, b^*_{(t,1)}) \\ &\geq g_2(\mathbf{w}^*_{(t)}, b^*_{(t,2)}) \\ &\geq g_3(\mathbf{H}^*_{(t)}) = a_2 \end{aligned} \tag{B.10}$$

where $t \to \infty$ and a_1, a_2 are limit values in $\mathbb{R}$ and $b^*_{(1,1)}$ is the value for b acquired when solving for $\mathbf{G}$ at time 1, while $b^*_{(1,2)}$ is the value for b acquired when solving for $\mathbf{w}$ at time 1. Therefore we can regard the alternating optimization procedure to be a composition

of 3 subalgorithms defined as:

$$\Omega^1 : (\mathbf{G}, b) \rightarrow \mathbb{R}^{m \times k} \times \mathbb{R} \tag{B.11}$$

$$\Omega^2 : (\mathbf{w}, b) \rightarrow \mathbb{R}^{m} \times \mathbb{R} \tag{B.12}$$

$$\Omega^3 : (\mathbf{H}) \rightarrow \mathbb{R}^{k \times n} \times \mathbb{R}. \tag{B.13}$$

producing $\mathbf{G}, \mathbf{H}, \mathbf{w}$ and b. Then $\Omega = \Omega_1 \circ \Omega_2 \circ \Omega_3 = \circ_{d=1}^{3} \Omega_d$ is closed when all $\mathbb{G}, \mathbb{H}, \mathbb{W}$ are compact. We should emphasize here that since all subalgorithms decrease the value of D, Ω is monotonic with respect to D. Consequently, we can say that the alternating projection method converges.