AN ABSTRACT OF THE THESIS OF

Anirban Roy for the degree of Master of Science in Computer Science presented on August 30, 2013.

Title: Learning Affinities of Exemplar Videos for Multi-View Action Recognition

Abstract approved: _____

Sinisa Todorovic

This thesis addresses a fundamental computer vision problem, that of action recognition. The goal of action recognition is to recognize a class of human actions in a given video. Action recognition has a wide range of applications, including automated surveillance, sports video analysis, internet-based searches etc. The main challenge is that actors move and change their pose while performing the action, and/or that the actors may be captured from different viewing angles. For example, certain viewpoints of a distinct action class may produce very similar appearance and motion features in the videos. This may cause confusion in recognizing the two action classes. Therefore, action recognition should be invariant to changes in camera viewpoints. While action recognition in videos captured from a fixed view has received significant interest in the past, multi-view action recognition is still an under-explored field.

We have specified an exemplar based approach for multi-view action recognition. We propose a novel method for training the K - Nearest Neighbor (K-NN) classifier.

Specifically, the K-NN is learned within a large-margin framework with a set of suitable constraints specified over camera viewpoints and action classes. Our constraints enforce that affinity between videos belonging to the same class should be larger than that of the videos belonging to distinct classes. Moreover, within the same action class, affinity between videos captured from the same camera viewpoint should be larger than that of distinct viewpoints. We efficiently compute the affinity between any two videos based on many-to-many matching of their supervoxels. The correspondences of the supervoxels of two videos are treated as latent random variables. Thus, we formulate a novel latent large-margin learning of the K-NN, subject to a set of viewpoint and class constraints. Given a new video, we use K-NN classifier to identify K closest training exemplars to the video, and transfer their majority action class label as the class label of the new video. Our approach outperforms the state of art on benchmark datasets: INRIA IXMAS, a newer version of IXMAS (NIXMAS) and i3DPost.

Learning Affinities of Exemplar Videos for Multi-View Action Recognition

by

Anirban Roy

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented August 30, 2013
Commencement June 2014

Master of Science thesis of Anirban Roy presented on August 30, 2013.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Anirban Roy, Author

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Sinisa Todorovic for the continuous support and valuable advices which helped to accomplish the project goal. I would like to thank Prof. Eugene Zhang, Prof. Ronald Metoyer and Prof. Len Coop for agreeing to be in my committee. I would also like to thank my lab mates for their suggestions.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

To my Parents, Family and Friends.

Chapter 1: Introduction

A human action is defined as a spatiotemporal configuration of body movements of a single person. Typical human actions considered in computer vision, include walking, running, dancing, kicking, punching etc. The goal of action recognition is to identify the class of action performed in a video. Some applications of action recognition include, automated surveillance in public places (e.g. airports and stations) and monitoring of patients in nursing home [10]. Recently action recognition has been applied in sports videos as well, to analyze the performance of players [27].

Typically, prior research focuses on a sanitized setting where actors are captured from a single, pre-specified camera viewpoint (e.g. frontal view). In this thesis, we focus on developing a multi-view approach for action recognition. Our goal is to recognize different classes of actions from the videos, which were recorded from multiple predefined viewpoints.

Recognition invariance to viewpoint changes is critical for a number of applications, because human actions may unfold in arbitrary directions relative to the camera viewing axis, or actions may be captured with arbitrarily pointed cameras. This problem is challenging, because human movements and poses that are characteristic of an action class may appear foreshortened in 2D space and time under perspective projection, and thus falsely characteristic of another action class. In addition, movements of the actor's arms and legs characteristic of an action class (e.g., kicking) may not be visible from all

viewpoints. So there is a need for developing multi-view action recognition methods.

We propose a novel method for training K-NN classifier. Specifically, the K-NN is learned within a large-margin framework with a set of suitable constraints specified over camera viewpoints and action classes. The constraints enforce that affinity between videos belonging to the same class should be larger than that of the videos belonging to other classes. Moreover, within the same action class, affinity between videos captured from the same camera viewpoint should be larger than that of distinct viewpoints.

Our approach consists of following steps. First, a video is oversegmented in space-time into small spatiotemporal segments, called supervoxels. Second, we efficiently compute the affinity between any two videos based on many-to-many matching of their supervoxels. The correspondences of the supervoxels of two videos are treated as latent random variables. Then, we formulate a novel latent large-margin learning of the K-NN, subject to a set of viewpoint and class constraints. Finally, the learned K-NN is used for classification of test videos. In the following section we motivate our approach.

1.1 Motivation of our Approach

Motivated by the recent proliferation of large video datasets, we formulate an exemplar-based approach to multi-view action recognition. Its appeal is conceptual simplicity, while addressing the aforementioned challenges of multi-view action recognition. Given a test video, we identify its K nearest neighbors from a large annotated dataset, and transfer their majority action class to the video. Our approach has a number of advantages over existing approaches. It relaxes the stringent requirements for 3D scene/human re-

construction. It increases robustness, especially in terms of handling (self-)occlusions and background clutter, by accounting for the spatiotemporal structure of actions, and by partitioning foreground from background video parts. At the same time, it is computationally efficient, does not require retraining with the appearance of new data, and is scalable to an increasing number of action classes and number of viewpoints. Finally, with the rapidly growing abundance of video data, exemplar-based approaches to multi-view action recognition are becoming a viable solution in many applications. This is because, the K-NN classifier can be viewed as a non-parametric probability density function (pdf) estimator, which counts data samples in a neighborhood of our query video, where robustness of this estimation grows with an increasing number of videos collected every day.

1.2 Overview of our Approach

In this section, we first give the intuition and then provide the basic details of our approach. Here we formally define the affinity between two videos and two types of constraints. Then we explain how constraints are imposed in learning, based on action classes and viewpoints of the videos.

Consider the triplet of videos from the NIXMAS dataset [48], shown in the top row of Figure 1.1. Suppose we want to estimate the action class and viewpoint of a query video, v_i. We have access to exemplar videos v_j and v_l, annotated with their respective action-class and viewpoint labels. Let $\mathcal{A}_{ji}$ and $\mathcal{A}_{li}$ be the asymmetric affinities from v_j to v_i, and from v_l to v_i. Since v_i and v_j show instances of the same action class,

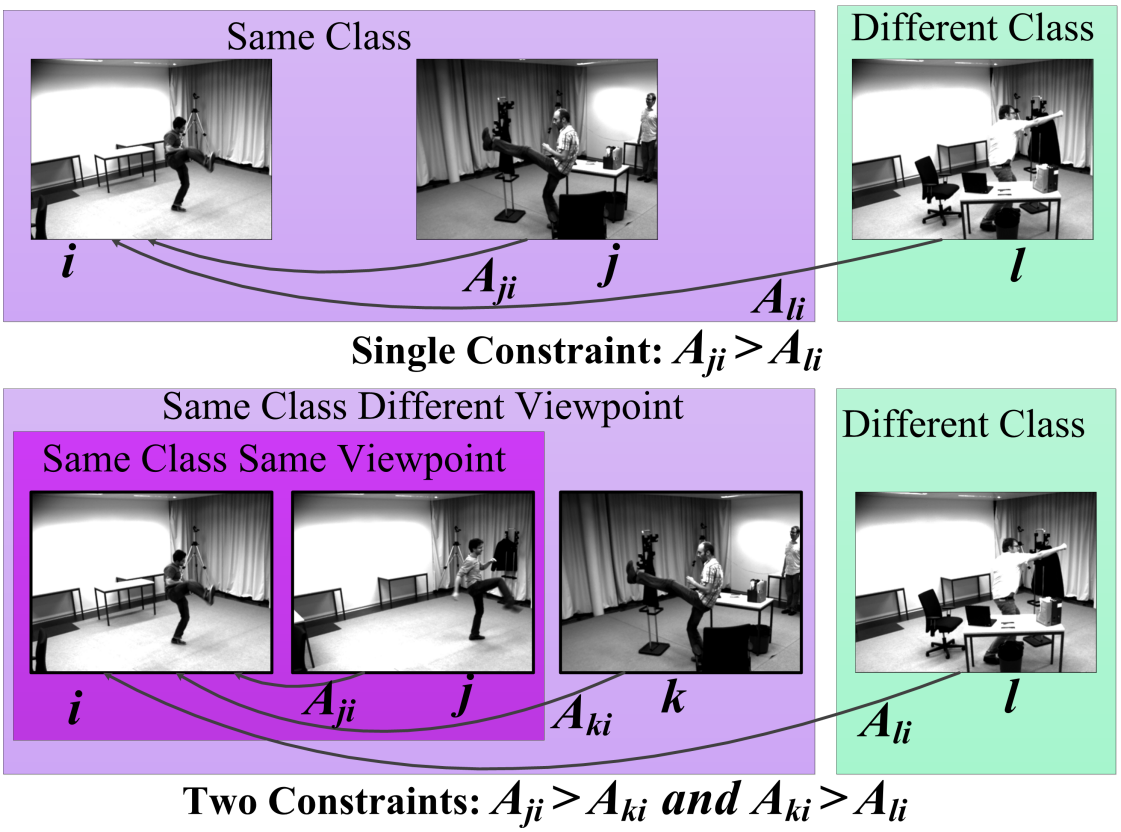

Figure 1.1: An overview on sample videos from the benchmark NIXMAS dataset [48]. (**top**) The video triplets – v_i and v_j are of the same class, and v_l is from a different class – can be used to learn the K-NN classifier under the constraint that for all triplets $\mathcal{A}_{ji} > \mathcal{A}_{li}$. But this does not ensure accuracy in viewpoint estimation. (**bottom**) It takes quadruples – v_i and v_j are of the same class and same view, and v_k is from the same class but different view, and v_l is from a different class – to learn the K-NN classifier under the two constraints that for all quadruples $\mathcal{A}_{ji} > \mathcal{A}_{ki}$ and $\mathcal{A}_{ki} > \mathcal{A}_{li}$ for ensuring accuracy in both action-class and viewpoint estimation.(Best viewed in color)

we expect that $\mathcal{A}_{ji} > \mathcal{A}_{li}$. If so, the 1-NN classifier based on these affinities would correctly transfer the action class of v_j to v_i. But, what about the viewpoint? Clearly, we would like that the relation $\mathcal{A}_{ji} > \mathcal{A}_{ki}$ simultaneously informs the 1-NN classifier that both action class and viewpoint of v_j can be transferred to v_i. Therefore, our goal is to learn these video affinities so as to ensure high accuracy of the K-NN classifier in both action-class and viewpoint estimation. This additional viewpoint information helps in action recognition.

This can be efficiently achieved by learning local, asymmetric affinity functions characterizing each exemplar video in the dataset. The key advantage of tying the affinity function locally with a particular video is that it allows K-NN to account only for the discriminative foreground action parts of that video. This is particularly important for multi-view action recognition, because the action-characteristic movements of the actor seen in one video, may not be visible in other videos taken from different viewpoints, due to self-occlusion. In addition, any changes of the dataset would require retraining of only a relatively small number of the local affinity functions, which would not be the case with a global unique affinity defined over the entire feature space.

For computing the affinity among the videos, we consider quadruples (v_i, v_j, v_k, v_l), where v_j comes from the same class and same view as v_i; v_k is from the same class as v_i; and v_l is from a different class, as illustrated in the bottom row of Figure 1.1. Given a set of these quadruples, we would like the following two constraints hold with respect to v_i:

- Constraint 1: $\mathcal{A}_{ji} > \mathcal{A}_{ki}$, for all videos v_j and v_k,
- Constraint 2: $\mathcal{A}_{ki} > \mathcal{A}_{li}$, for all videos v_k and v_l.

By transitivity, $\mathcal{A}_{ji} \geq \mathcal{A}_{li}$ will also hold. Such affinities, under Constraint 1 and Constraint 2, will enforce K-NN to simultaneously consider both viewpoint and action-class when assigning labels to the query videos, as desired.

To approach this problem, we define the affinity as a weighted linear combination of similarities between video features. Thus, for a pair of videos, v_j and v_i, we define their similarity vector $\boldsymbol{x}_{ji} = \boldsymbol{x}(v_j, v_i)$ indexed by the features of v_j, and use it to specify the asymmetric affinity:

$$\mathcal{A}_{ji} = \boldsymbol{w}_j^T \cdot \boldsymbol{x}_{ji}. \tag{1.1}$$

The intuition is that the weights, $\boldsymbol{w}_j$, will be large for features of v_j that are relevant for multi-view action recognition, and small, otherwise. Essentially, the role of $\boldsymbol{w}_j$ associated with each video v_j in the dataset is to identify the discriminative foreground action parts of that video, and thus enable a more robust K-NN classifier. So, our goal is to learn the weights $\boldsymbol{w}_j$, $\boldsymbol{w}_k$ and $\boldsymbol{w}_l$, for 'all' v_j, v_k and v_k such that the Constraint 1 and Constraint 2 among the quadruples of videos hold for 'all' v_i .

For a target video we compute weighted affinities of all the exemplars to it. The affinities are computed based on the learned weights. Then a K - nearest neighbour (K-NN) classifier with the maximum voting scheme is used to determining the action class of the target video.

1.3 Organization of the Thesis

This paper is organized as follows. In chapter 2 we review prior work in three areas: action recognition, multi-view action recognition and large margin distance learning.

Chapter 3 provides details about our feature extraction mechanism and how we represent videos using those features. Chapter 4 describes problem formulation and Chapter 5 specifies the matching between supervoxels of two videos. Chapter 6 presents latent large-margin learning of K-NN classifier. Chapter 7 presents our results on three standard datasets and comparisons with state-of-the-art approaches.

Chapter 2: Literature Review and Our Contributions

There is a large volume of work on action recognition, and reviewing the entire literature is beyond the scope of this thesis. We only review closely related literature. We provide detailed review of prior work on multi-view action recognition. Also, we briefly review the large margin metric learning literature to provide the background because our learning mechanism is a variant of metric learning approach and it is performed in a large margin framework.

2.1 Prior Work on Action Recognition

Action recognition approaches can be divided in two broad categories: 1. Classifier based approaches and 2. Spatiotemporal model based approaches [33]. The first group of approaches do not model time explicitly. They usually summarize the entire video in a single representation. Some approaches use a specific representation for each frame. These holistic or frame based representations of the video are assumed to be distinctive across different classes, and thus used for classification. For example, K-NN and discriminative classifiers such as Support Vector Machine (SVM) are the popular choices for classification.

The second group of approaches usually have states in the model where a state represents an atomic sub-action performed in a particular moment. Spatio-temporal models

are of two types: 1) generative models and 2) discriminative models. The most popular generative model is Hidden Markov Model (HMM) [8, 20, 42, 52], where the joint probability is computed over two types of variables: hidden variables and observed variables. Hidden variables are states that correspond to atomic actions, and observed variables are features associated with states. To avoid complexity in the inference logic this type of models assume the Markov assumption: state transitions are conditioned only on the previous state and observation from a state is conditioned on that state only. Discriminative models such as Conditional Random Field (CRF) [31, 35] can account multiple observations from different time, and thus this model can overcome the limiting Markov assumption. These models are trained to discriminate between action classes based on the observation instead of learning the joint probability of action class and observation as in generative models. In [28, 38] CRFs are used for action recognition. A detailed review of action recognition is done by Aggarwal et al. [1]. Spatiotemporal model based approaches usually have models for different action classes.

However, having a specialized model for a specific action class might not be suitable for multi-view action recognition as a single model might not be sufficient to capture the large variation of the action caused by varying viewpoints. Inference mechanisms for the spatiotemporal model based approaches are more complex than that of the classifier-based approaches. So we prefer the classifier based approach over spatiotemporal model based approaches for our work.

We use K-NN classifier with maximum voting scheme for classification. To apply K-NN classifier, the entire or a part of the video is represented by a feature vector. Training is performed to determine feature weights from labelled training videos. Based

on the learned weights, k closest training examples are chosen for each of the testing video. The label of the testing video is decided by the label of maximum number of videos among k training videos. K-NN classifiers are used for action recognition in a variety of approaches [2, 3, 4, 34, 40, 45, 47]. K-NN classifier needs to compute distance among the exemplar videos for classification. The distance between two videos can be computed at frame level or over entire video sequences. In the later case whole video sequence is represented by a single global feature vector. Blank et al. [3] use global feature representation while Batra et al. [2] use Bag of Words (BOW) representation of videos. Though Euclidean distance is commonly used as the distance measure between the features, Bobick and Davis [4] propose Hu moments of different order as an alternate distance metric. Some approaches [34, 40] learn discriminative distance metric for classification. Weinland and Boyer [45] propose an exemplar based framework with K-NN classifier for action recognition.

Inspired by the success and simplicity of the K-NN classifier in action recognition, we use it for our recognition method. Similar to [47], we also use supervoxels as features for action recognition.

2.2 Prior Work on Multi-View Action Recognition

In multi-view action recognition, in addition to the action performed in the video, viewpoint information is also needed to recognize the action in the video. Various approaches are proposed to capture the viewpoint information in the multi-view action recognition literature. We categorize the multi-view action recognition approaches in three broad

categories: model based, view-invariant descriptor discovery and knowledge transfer approaches.

2.2.1 Model Based Approaches

In model based multi-view action recognition approach, human body is represented by 2D or 3D models. Then, view-invariant features are extracted from the models for action recognition. These models are built using very fine details such as silhouettes of the human body, point trajectories of the body junctions, poses of the body parts etc. For example, [44] presents an exemplar based approach where projected silhouettes of the human body are stored as action models and HMM is used to find correspondence between template model and target video models. In [54] a spatiotemporal body contour is generated as an action model while Yan et al. [53] develop a 4D action model (4D-AFM) from the spatiotemporal volume information. High curvature points are extracted from the 4D-AFM as view-invariant features.

Trajectory based modelling of body motion is proposed in [36] and [55] where junction points of human body are tracked to obtain the trajectories. Properties of these trajectories such as alignment and how they evolve in time are used as discriminative action features. Parameswaran et al. [32] try to find projective invariants of coplanar landmarks for action representation.

Pose informations of an action provide informative queues for action recognition. For example, a pose of ‘stretching hand’ hints that the corresponding action can be of ‘waving hand’ or ‘punching’ but its less likely to be a action of ‘sit down’ or ‘get up’.

Mikic et al. [29] and Cheng et al. [6] use the 3D human pose as queues for multi-view action recognition. First approach [29] uses the video volume data to locate body parts (head, hand, leg etc) for pose estimation while the second one [6] relies on Gaussian Mixture Model to track body parts and hand models. Additional kinematic constraints can be imposed on the body model for more accurate representation [6].

Model based approaches capture detailed information about the action, so performance of these approaches is usually satisfactory. But these modelling techniques are computationally expensive and occlusion can easily introduce errors in them. So we avoid these type of approaches.

2.2.2 View-Invariant Descriptor Discovery

Another line of work focuses on finding rich descriptors that have view-invariance property. Junejo et al. [23] compute symmetric self-similarity matrix for frame-to-frame transition as a discriminative property of a particular action. In [46] they use 3D HOG as a view-invariant descriptor which can also handle occlusion. Cohen et al. [7] use the cylindrical histogram of voxels as a view-invariant descriptor and [21] relies on 3D cylindrical shape-context to capture human body configuration. Motion based 3D descriptors are shown to have view-invariance property by Holte et al. [18]. They use 3D Motion Context (3D-MC) and Harmonic Motion Context (3D-HMC) as view-invariant descriptors. 3D-MC can be viewed as a 3D version of shape context [18] descriptor which implicitly incorporates motion information by embedding 3D histogram of optical flow (3D-HOF) in the descriptor. 3D-HMC is an advanced version of 3D-MC which

decomposes the feature representation into a set of spherical basis functions.

These types of approaches are less complex as the descriptor computation is easier than modelling the human bodies. Also linear classifiers such as SVM can be used for classification, once descriptors are computed. We use similar representation of videos, where supervoxels are extracted from videos and each supervoxel is represented by the HOG and HOF features.

2.2.3 Knowledge Transfer Based Approaches

The knowledge transfer based approaches gain knowledge from the videos of certain viewpoints, and then apply this knowledge to recognize action in the videos from other viewpoints. For example, if we see a video of 'walking' recorded from the side view, that still helps to recognize the action from a video recorded from frontal view. The more viewpoints we see in training the better the recognition accuracy in testing. Farhadi $et.al.$ [11] obtain features from multiple viewpoints of an action and let the classifier learn the transition of features from training views to target views. In [26], actions are represented as a bag of 'bilingual words' where each of the word is computed from two different views. Li et al. [25] propose an approach where descriptors from source view to target view can be bridged by a series of linear transformation in an augmented feature space.

A review of recent progress in multi-view action recognition can be found in [19].

2.3 Prior Work on Large Margin Distance Metric Learning

Accuracy of the K-NN classifier that we use for our approach, significantly depends on the selected features and their weights while computing distance between exemplars. Finding appropriate features and their weights is itself a separate line of research. This problem is known as *distance metric learning* (DML) and usually this learning is done in a large margin framework. Approaches for solving the DML problem fall into three broad categories: 1. Eigenvector method, 2. Convex optimization and 3. Fully supervised method to optimize KNN classifier error [43].

Eigenvector methods usually focus on finding linear transformations which map input features to a target feature space, so data analysis becomes efficient. Popular eigenvector methods are principal component analysis (PCA), linear discriminant analysis (LDA), relevant component analysis (RCA) etc. PCA is widely used as a dimensionality deduction tool and LDA is used for liner preprocessing of data in a supervised setting. A 'kernelized' version of these type of transformations can be used to project the features in a non-linear feature space [30, 41].

In Convex optimization based approaches, DML is formulated as a convex problem and usually solved using semi-definite programming (SDP). This type of approach is first introduced by Xing et al. [50], where the goal is to learn a distance metric such that distances between exemplars from different classes are high while distances between same class exemplars are kept low.

In a fully supervised settings, we have labeled data and the aim to find a distance metric that minimizes the empirical misclassification error. In neighborhood component

analysis (NCA) approach, a distance metric is learned especially for K-NN classifier. This method is proposed by Goldberger et al. [16], where expected leave-one-out class-satisfaction error is computed by stochastic K-NN method. Then a linear transformation is learnt which minimizes the expected misclassification error. Globerson et al. propose another method known as Metric Learning by Collapsing Classes (MLCC) [15]. This method finds a distance metric which keeps intra-class variance low, keeping inter-class variance high.

Weinberger et al. [43] aim to learn the Mahalanabis distance metric to compute distance between two feature vectors. Assume two feature vectors: x_i and x_j, then the distance is computed as $\mathcal{D}_Z(x_i, x_j) = (x_i - x_j)^T \boldsymbol{Z}(x_i - x_j)$, where $\boldsymbol{Z}$ is the distance metric to be learnt. These types of approaches usually aim to learn feature weights for every dimension. Another type of distance learning method, introduced by Schutlz et al. [37], which goes to an extreme and aims to find feature weighs for each of the training exemplars. This method learns global distance metric from local information by comparing pair of exemplars. The constraint of learning is that distances of inter-class exemplars have to be higher that intra-class exemplars and this constraint should hold for any pairs of exemplars. Similar idea of using local distance function for learning feature weights of the exemplars, is successfully applied for object retrieval and classification by Frome et al. in [12] and [13].

We advance the K-NN based learning approach by considering triplets of exemplars for computing local distance function instead of pairs, so that distance learning become suitable for multi-view action recognition. Learning of weights is done in a more constrained settings, where apart from class based constraints, viewpoint based constraints

are added to include the information about viewpoints in recognition.

2.4 Our Contributions

As our key contribution, we formalize the learning problem within the *latent* large-margin framework [56], subject to constraint 1 and constraint 2 as mentioned in Section 1.2. The latent variables, H_{ji}, are defined as unknown correspondences between features of two videos, v_j and v_i, that are used for computing their similarity vector $\boldsymbol{x}_{ji} = \boldsymbol{x}(v_j, v_i)$. As our results demonstrate, estimation of these correspondences is crucial for multi-view action recognition, since we cannot expect that the actor would be prominently featured, in the center of the video frames, while performing an action. Rather, in real videos, the actor's trajectory may take them to various locations within the space-time volume of the video. Thus, for robust multi-view action recognition, it is critical to estimate which video parts correspond to the activity of interest, and use these foreground visual cues for computing the affinity between the videos. Since discriminative video parts are not known a priori, our contribution is to incorporate estimation of the latent variables $H_{j\cdot}$ together with learning the affinity weights $\boldsymbol{w}_j$. $H_{j\cdot}$ is formalized as many-to-many matching of video features, and optimized so as to increase the classifier margin.

Similar large-margin learning from relative comparisons of images has been used for image classification [12]. The approach of [12] is concerned only with the image class, and thus uses only constraint 2, whereas we introduce in the large-margin learning both constraint 1 and constraint 2. Also, [12] does not have latent variables, whereas we

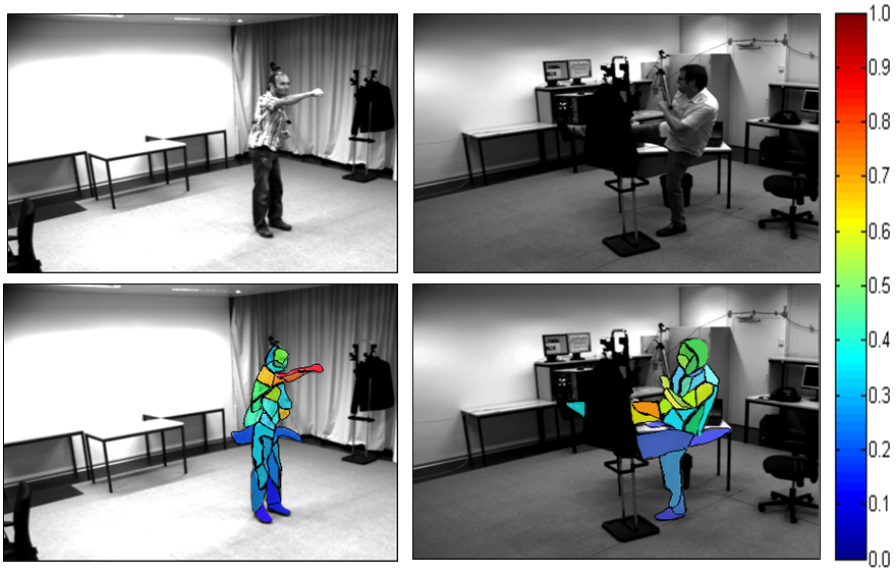

Figure 2.1: Boundaries of supervoxels are usually aligned with object boundaries. (left) The actor's arm is partially occluded, but we still identify the supervoxel corresponding to the actor's other arm, and use it to recognize "punching". (right) The kicking leg is partially occluded, and yet its supervoxel enables localizing the informative visual cue. Shown weights are normalized between zero and one.

specify the latent large-margin learning.

As another contribution, we argue that mid-level video features, i.e., supervoxels, are more appropriate for multi-view action recognition than common spatiotemporal points [36]. Our choice of supervoxels as basic video features is motivated by the observation that partial self-occlusions in different views seem to present the major obstacle to capturing visual cues characteristic of the action class. This is well illustrated in Figure 2.1 that shows two videos of the "punching" and "kicking" classes from the IXMAS dataset [48], and their supervoxels (i.e., intersections of supervoxels with the selected video frame). As can be seen, boundaries of supervoxels are nearly aligned with object

boundaries. Therefore, recognizing the right combination of supervoxels immediately delineates the actor and parts of their body. Access to parts is important under occlusion, as shown in both frames in Figure 2.1. Although, we cannot see the actor's arm, because it is behind the actor's body, we can still identify the supervoxel corresponding to the actor's other arm, and use this visual cue to recognize "punching". In the other example, the kicking leg is partially occluded, and yet its supervoxel enables localizing the informative visual cue. Prior works [7, 32] use relatively larger space-time tubes as video features for multi-view action recognition. However, the tubes are less stable than supervoxels, and often merge the actor with the low-contrasting background.

We formulate a novel approach to multi-view action recognition based on latent large-margin learning of K-NN, where:

- Videos are represented by supervoxels as its features,
- Supervoxel correspondences are latent variables, and
- Local, asymmetric video affinities are subject to Constraint 1 and Constraint 2

Our approach outperforms the state of art by 9.04% on the IXMAS [48], and by 7.4% on the NIXMAS [46], and by 3.5% on the i3DPost [14].

Chapter 3: Feature Extraction

We use supervoxels as features for our approach. Supervoxels are described by histogram of oriented gradients (HOG) and histogram of optical flow (HOF) descriptors. Steps for feature extraction are explained below.

First, given a dataset we detect Space Time Interest Point (STIP) [24] features from the training videos. Each STIP features is represented by the HOG [9] and HOF [5] descriptors. We cluster all STIPs into 1000 clusters by K-means clustering. Then we obtain 1000 codewords as the cluster centers.

Second, we extract supervoxels from each exemplar video by Graph Based Hierarchical segmentation (GBH) [17] using libsvx software [51]. In GBH method, a video is initially over-segmented into spatiotemporal supervoxels. Then, a spatiotemporal smoothing is performed over initial noisy supervoxels. This helps to obtain larger and coherent supervoxels. We only consider the supervoxels containing atleast one STIP. As videos usually have static background, STIPs usually fall in the supervoxles corresponding to moving body parts. This helps us to eliminate spurious background supervoxels.

Third, we represent each supervoxel using the BoW model based on the STIP points that fall in the supervoxel and previously obtained codewords. So each supervoxel is represented as a histogram of codewords. Similar type of feature representation is successfully used in multi-view action recognition by Li et al. [25].

Finally, we use the negative exponential of the Chi-square χ^2 distance to compute

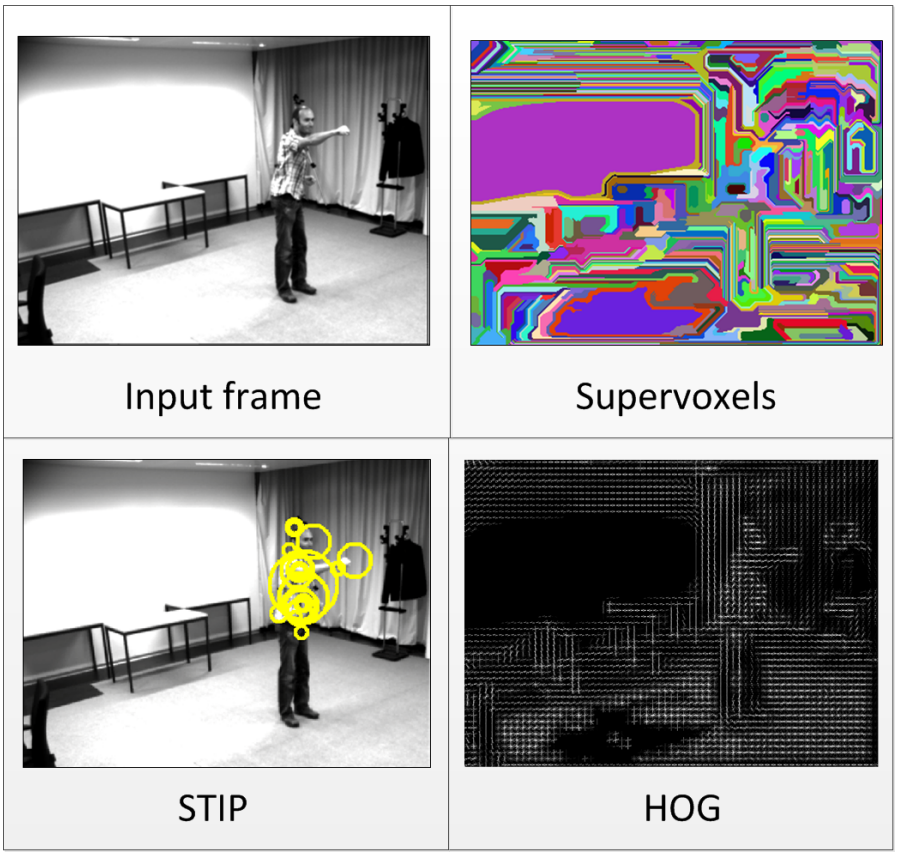

Figure 3.1: Visualization of the supervoxels, STIP locations and HOG features extracted 'punching' video from NIXMAS dataset.

the affinity between two supervoxels. We provide a visualization of the supervoxels, STIP locations and HOG features corresponding to a video of 'punching' action from the NIXMAS dataset in Figure 3.1. Top left: a frame extracted from the video of the 'punching' action. Top right: temporal cross-section of the supervoxels corresponding to the frame, Bottom left: STIP locations in the frame and Bottom right: HOG features extracted from the frame.

Chapter 4: Problem Formulation

This section introduces the latent variable and explains how to compute affinity of two videos based on the latent variable. Here, we also mention a compact feature representation, that fits in large margin learning formulation, explained in Chapter 6.

Similarity Matrix: Given two videos, v_j and v_i, with a total of M_j and M_i features, we define an $M_j \times M_i$ similarity matrix, $X_{ji} = [X_{ji}(m,n)]$, where $X_{ji}(m,n) \geq 0$ represents similarity of mth feature in v_j and nth feature in v_i.

Latent Variables: Many-to-many correspondence of features from two videos, v_j and v_i, is specified as an indicator matrix $H_{ji} \in \{0,1\}^{M_j \times M_i}$, where $H_{ji}(m,n) = 1$ denotes that mth feature of v_j is matched with nth feature of v_i, and $H_{ji}(m,n) = 0$, otherwise. Many-to-many matching is aimed at addressing errors of spatiotemporal video segmentation with noisy mergers and splits of supervoxels. It allows a set of supervoxels from v_j to be matched to a single supervoxel in v_i, and conversely a set of supervoxels from v_i to be matched to some other supervoxel in v_j.

Asymmetric Similarity Vector: Expected similarity of M_j features from video v_j to video v_i is defined as

$$x_{ji}(m) = \sum_{n=1}^{M_i} X_{ji}(m,n) \odot H_{ji}(m,n),\ m = 1, ..., M_j. \tag{4.1}$$

where '$\odot$' is element-wise multiplication operator.

Note that the many-to-many matching formulation allows zero rows in H_{ji}. Consequently, some elements of the similarity vector, $\boldsymbol{x}_{ji} = [x_{ji}(m)]^T$ can be equal to zero. The intuition is that $x_{ji}(m) = 0$ means that mth feature in v_j is much different from all features in v_i, and thus can be regarded as irrelevant background with respect to the action occurring in v_i.

Asymmetric Affinity: Elements of the similarity vector, $\boldsymbol{x}_{ji} = [x_{ji}(m)]_{M_j \times 1}$, are weighted and linearly combined to compute the asymmetric affinity from video v_j to video v_i, $\mathcal{A}_{ji} = \boldsymbol{w}_j^T \cdot \boldsymbol{x}_{ji}$, as defined in Equation 1.1. The weight vector $\boldsymbol{w}_j$ is aimed at assigning relevance to every feature in v_j for multi-view action recognition with respect to all v_i in the training set. Note that $\boldsymbol{w}_j$ uniquely characterizes v_j, and thus makes $\mathcal{A}_{ji}$ asymmetric, i.e., in general, $\mathcal{A}_{ji} \neq \mathcal{A}_{ij}$.

Our goal is to jointly learn all weight vectors, $\boldsymbol{w} = [\boldsymbol{w}_1^T, ..., \boldsymbol{w}_i^T, ...]^T$, of exemplar videos in a given dataset, $\mathbb{V} = \{(\boldsymbol{\phi}(v_i), y_i, z_i) : i = 1, ..., N\}$, subject to Constraint 1 and Constraint 2, where $\boldsymbol{\phi}(v_i)$ is the set of extracted supervoxels, y_i denotes the action class, and z_i denotes the viewpoint of v_i. To this end, we consider quadruples of exemplar videos $\{v_i, v_j, v_k, v_l\} \subset \mathbb{V}$, where, for each $v_i \in \mathbb{V}$, v_j is a class-view hit, v_k is a class-hit, and v_l is a class-miss, as illustrated in Figure 4.2.

A class-view hit to v_i, denoted as v_j, has the same action class and viewpoint as v_i, $(y_i, z_i) = (y_j, z_j)$.

A class-hit to v_i, denoted as v_k, has the same action class but different viewpoint from v_i, $y_i = y_k$, and $z_i \neq z_k$.

A class-miss to v_i, denoted as v_l, has a different action class from v_i, $y_i \neq y_l$.

From the definitions of Constraint 1 and Constraint 2, stated in Sec. 1.2, and the

definitions presented in this section, the two constraints can be specified as:

$$\begin{bmatrix} \boldsymbol{w}_j \\ \boldsymbol{w}_k \end{bmatrix}^T \begin{bmatrix} \boldsymbol{x}_{ji} \\ -\boldsymbol{x}_{ki} \end{bmatrix} \geq 1, \quad \begin{bmatrix} \boldsymbol{w}_k \\ \boldsymbol{w}_l \end{bmatrix}^T \begin{bmatrix} \boldsymbol{x}_{ki} \\ -\boldsymbol{x}_{li} \end{bmatrix} \geq 1. \tag{4.2}$$

For considering Equation 4.2 over all videos $v_i \in \mathbb{V}$, it is convenient to define sparse vectors Φ_{ijk} and Φ_{ikl} as

$$\Phi_{ijk} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{x}_{ji} \\ -\boldsymbol{x}_{ki} \\ \mathbf{0} \end{bmatrix}, \Phi_{ikl} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{x}_{ki} \\ -\boldsymbol{x}_{li} \\ \mathbf{0} \end{bmatrix}, \tag{4.3}$$

where all elements of Φ_{ijk} and Φ_{ikl} are zero except those that correspond to $\boldsymbol{x}_{ji}$, $\boldsymbol{x}_{ki}$, and $\boldsymbol{x}_{li}$. Note that $\Phi_{ijk} = \Phi_{ijk}(X_{ji}, X_{ki}, y_i, z_i, H_{ji}, H_{ki})$, which we also write as $\Phi_{ijk} = \Phi_{ijk}(y_i, z_i, H_{ji}, H_{ki})$, for short. Similarly, we write $\Phi_{ikl} = \Phi_{ikl}(y_i, z_i, H_{ji}, H_{ki})$. To clarify the structure of $\boldsymbol{w}$ and Φ vectors, we present a visualization of their representation in Figure 4.1. Where we see Φ_{ijk} is represented as a vector of the same length as $\boldsymbol{w}$, where all the values of this vector are 'zero' except $\boldsymbol{x}_{ji}$ and $-\boldsymbol{x}_{ki}$, which are in the same range as $\boldsymbol{w}_j$ and $\boldsymbol{w}_k$ in $\boldsymbol{w}$ respectively. Moreover lengths of $\boldsymbol{x}_{ji}$ and $\boldsymbol{x}_{ki}$ are same as $\boldsymbol{w}_j$ and $\boldsymbol{w}_k$ respectively. This representation allows us to express $\boldsymbol{w}_j^T \cdot \boldsymbol{x}_{ji} - \boldsymbol{w}_k^T \cdot \boldsymbol{x}_{ki}$ in a compact way as $\boldsymbol{w}^T \cdot \Phi_{ijk}$.

From the Equations 4.2 and 4.3, it follows that all weight vectors $\boldsymbol{w}$ can be jointly

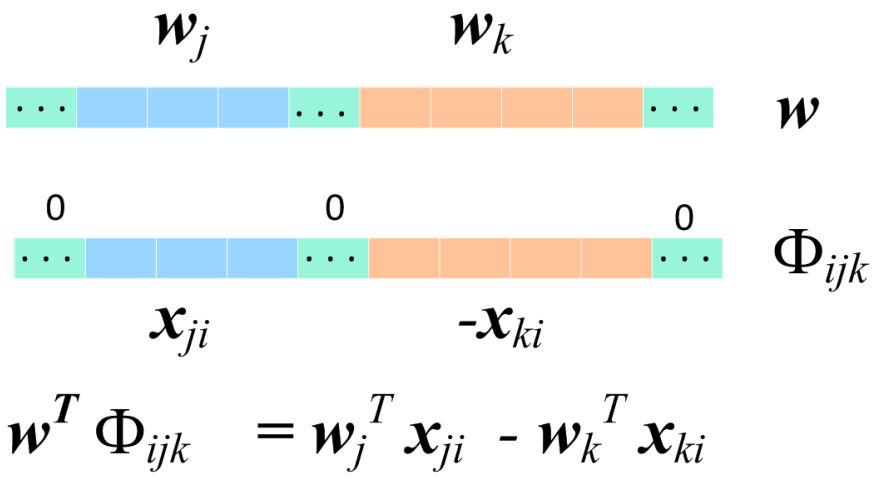

Figure 4.1: Representation of $\boldsymbol{w}$ and Φ vectors.

learned, such that for every $v_i \in \mathbb{V}$, the following two constraints are enforced:

$$\boldsymbol{w}^T \Phi_{ijk} \geq 1, \text{for all } \{v_i, v_j, v_k\} \subset \mathbb{V}, \tag{4.4}$$

$$\boldsymbol{w}^T \Phi_{ikl} \geq 1, \text{for all } \{v_i, v_k, v_l\} \subset \mathbb{V}, \tag{4.5}$$

where Φ_{ijk} represents for class-view hit and class-hit of v_i, and Φ_{ikl} is for class-hit and class-miss respect to v_i.

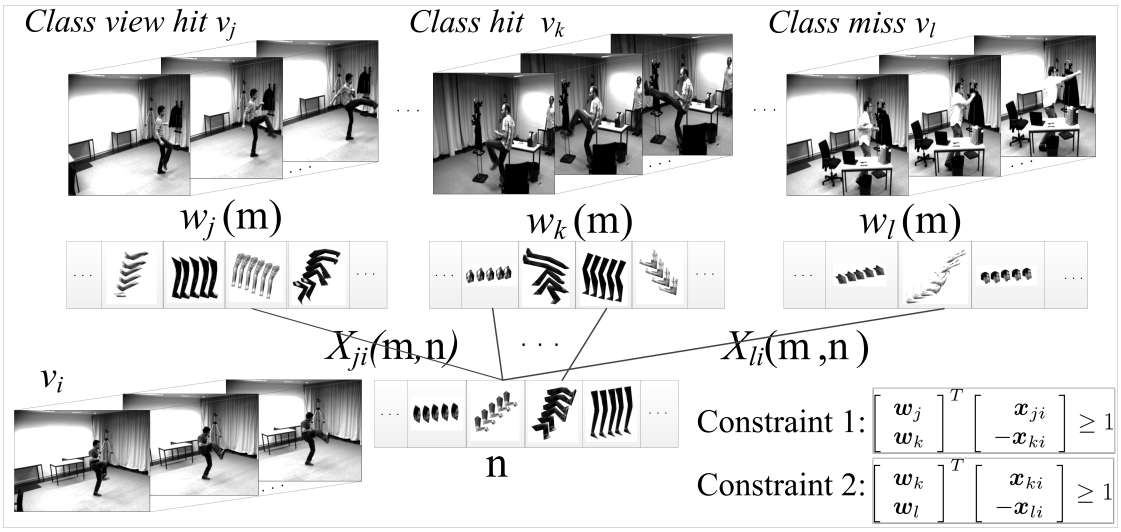

Figure 4.2: Learning by considering quadruples of videos (v_i, v_j, v_k, v_l). Supervoxels of v_i are efficiently matched to supervoxels of the class-view hit v_j, class-hit v_k, and class-miss v_l. The matching results $\boldsymbol{x}_{ji}$, $\boldsymbol{x}_{ki}$, $\boldsymbol{x}_{li}$ are used to specify Constraint 1 and Constraint 2 for latent large-margin learning of the weights $\boldsymbol{w} = [..., \boldsymbol{w}_j^T, ..., \boldsymbol{w}_i^T, ..., \boldsymbol{w}_k^T, ..., \boldsymbol{w}_l^T ...]^T$.

Chapter 5: Estimation of the Latent Variables

We define latent variables as the correspondence between the two sets of features (i.e. supervoxels) extracted from two videos. Establishing correspondences between video features is efficiently formulated as many-to-many matching with complexity $O(M^2)$ in the number of features M. This complexity comes from computing the similarity matrix X_{ji}, defined in Sec. 4. Since the number of extracted supervoxels is typically much smaller than that of other common video features ($M \approx 500$) for IXMAS and NIXMAS datasets, estimation of the latent correspondences, H_{ji}, does not represent a runtime bottleneck in our implementation.

Given two videos, v_j and v_i, and their similarity matrix $X_{ji} = [X_{ji}(m,n)]$, where $X_{ji}(m,n) \geq 0$ for all features $m = 1, ..., M_j$ and $n = 1, ..., M_i$, we want to find the feature correspondence indicator matrix, $H_{ji} = [H_{ji}(m,n)] \in \{0,1\}^{M_j \times M_i}$, so as to maximize the total similarity of features, $\sum_{m=1}^{M_j} \sum_{n=1}^{M_i} X_{ji}(m,n) H_{ji}(m,n)$. By relaxing the binary values of H_{ji} to the continuous domain, $H_{ji}(m,n) \in (0,1)$, our problem can be conveniently formulated as the following convex linear optimization:

$$\begin{aligned} &\text{maximize} \quad \text{vec}[X_{ji}]^T \text{vec}[H_{ji}] \\ &\text{subject to} \quad \|\text{vec}[H_{ji}]\|_2^2 = 1, \;\; H_{ji}(m,n) \geq 0, \end{aligned} \tag{5.1}$$

where the operator vec$[\cdot]$ takes a matrix and converts it into a vector by concatenating the matrix's columns. The convex problem of (5.1) is well-studied, and, for $X_{ji}(m,n) \geq 0$,

as is our case, has a closed-form solution:

$$\text{vec}[H_{ji}] = \frac{\text{vec}[X_{ji}]}{\|\text{vec}[X_{ji}]\|_2^2} \,. \tag{5.2}$$

The above solution, found in the continuous domain, is binarized to identify feature correspondences. Binarization is done by putting a threshold in the continuous solution. If the solution in the continuous domain is H_{ji} and the chosen threshold is t, then binarized solution H_{ji}^b is obtained as Equation 5.3.

$$H_{ji}^b(m,n) = \begin{cases} 1 & \text{if } H_{ji}(m,n) \geq t \\ 0 & \text{if } H_{ji}(m,n) < t \end{cases} \quad m = 1, ..., M_j.\ n = 1, ..., M_i. \tag{5.3}$$

While most of the related work on matching uses a heuristic threshold for converting continuous solutions to the discrete domain, we *learn* a suitable threshold as the latent variable in the latent large-margin learning. Learning the hidden variables is explained in the following chapter.

Chapter 6: Latent Large-Margin Learning

In this chapter, we explain the latent large margin learning framework aiming to compute $\boldsymbol{w}$. We want to learn $\boldsymbol{w}$ that satisfy the viewpoint and action class-based constraints in Equation 4.4 and 4.5. However, when dealing with real videos, it may not be possible to simultaneously satisfy all constraints in Equation 4.4 and 4.5. We therefore relax the constraints, using the notion of loss function, and formulate learning of $\boldsymbol{w}$ within the latent large-margin framework [56].

In particular, we define two loss functions corresponding to the two types of constraints:

$$\Delta_{ijk}^{(1)} = \Delta(y_i, z_i, H_{ji}^*, H_{ki}^*, \hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki}), \tag{6.1}$$

$$\Delta_{ikl}^{(2)} = \Delta(y_i, z_i, H_{ki}^*, H_{li}^*, \hat{y}_i, \hat{z}_i, \hat{H}_{ki}, \hat{H}_{li}) \tag{6.2}$$

where the loss in Equation 6.1 includes class-view hit v_j and class-hit v_k, whereas the loss in Equation 6.2 includes class hit v_k and class-miss v_l for video v_i. The symbol H^* indicates the estimates of the latent variables when the ground truth action class y_i and viewpoint z_i are given:

$$(H_{ji}^*, H_{ki}^*) = \underset{(H_{ji}, H_{ki})}{\text{argmax}}\ \boldsymbol{w}^T \Phi_{ijk}(y_i, z_i, H_{ji}, H_{ki}) \tag{6.3}$$

$(\hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki})$ in Equation 6.1, refers to the joint estimation of the labels (both

action class and viewpoint) and hidden variables:

$$(\hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki}) = \underset{(y,z,H_{ji},H_{ki})}{\text{argmax}} \; \boldsymbol{w}^T \Phi_{ijk}(y, z, H_{ji}, H_{ki}) \tag{6.4}$$

Symbols in second loss function $\Delta^{(2)}_{ikl}$, in Equation 6.2 carry same meaning as that in Equation 6.1.

Using these two loss functions, we define the following latent large-margin learning objective. Minimizing the loss function itself might result in over-fitting. To avoid that, we impose additional L_2 regularization penalty on $\boldsymbol{w}$.

$$\arg\min_{\boldsymbol{w}} \;\; \frac{1}{2}\|\boldsymbol{w}\|^2 + C_1 \textstyle\sum_i \sum_{jk} \Delta^{(1)}_{ijk} + C_2 \sum_i \sum_{kl} \Delta^{(2)}_{ikl} \tag{6.5}$$

C_1 and C_2 are regularization constants which control the relative importance of the regularization term over the empirical loss. Larger value of regularization constants implies smaller empirical error. We choose values for C_1 and C_2 by cross-validation.

Following the derivation presented in [56], we specify a hinge-loss style upper bound of the loss function in Equation 6.1:

$$\begin{aligned} \Delta^{(1)}_{ijk} \leq & \max_{(\hat{y}_i,\hat{z}_i,\hat{H}_{ji},\hat{H}_{ki})} \left[\boldsymbol{w}^T \Phi_{ijk}(\hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki}) + \Delta^{\text{hloss}}_{ijk} \right] \\ & - \max_{(H_{ji},H_{ki})} \boldsymbol{w}^T \Phi_{ijk}(y_i, z_i, H_{ji}, H_{ki}), \end{aligned} \tag{6.6}$$

where $\Delta^{\text{hloss}}_{ijk} = \Delta(y_i, z_i, \hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki})$ denotes the standard hinge loss:

$$\Delta^{\text{hloss}}_{ijk} = [1 - \boldsymbol{w}^T \Phi_{ijk}(\hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki})]_+ \tag{6.7}$$

The operator $[z]_+$ is defined as $\max\{0, z\}$.

Similar to Equations 6.6 and 6.7, we also derive the upper bound of the loss function $\Delta^{(2)}_{ikl}$ from Equation 6.2.

$$\begin{aligned}\Delta^{(2)}_{ikl} \leq & \max_{(\hat{y}_i, \hat{z}_i, \hat{H}_{ki}, \hat{H}_{li})} \left[\boldsymbol{w}^T \Phi_{ikl}(\hat{y}_i, \hat{z}_i, \hat{H}_{ki}, \hat{H}_{ki}) + \Delta^{\text{hloss}}_{ikl} \right] \\ & - \max_{(H_{ki}, H_{li})} \boldsymbol{w}^T \Phi_{ikl}(y_i, z_i, H_{ki}, H_{li}),\end{aligned} \tag{6.8}$$

where $\Delta^{\text{hloss}}_{ikl}$ is defined in a same way as Equation 6.7:

$$\Delta^{\text{hloss}}_{ikl} = [1 - \boldsymbol{w}^T \Phi_{ikl}(\hat{y}_i, \hat{z}_i, \hat{H}_{ki}, \hat{H}_{li})]_+ \tag{6.9}$$

From the Equation 6.5 and the upper bounds of the loss functions in Equations 6.6, 6.8, we obtain the following latent large-margin formulation:

$$\begin{aligned}\arg\min_{\boldsymbol{w}} \quad & \left[\frac{1}{2} \|\boldsymbol{w}\|^2 \right. \\ & + C_1 \sum_i \sum_{j,k} \max_{(\hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki})} \left(\boldsymbol{w}^T \Phi_{ijk} + \Delta^{\text{hloss}}_{ijk} \right) \\ & - C_1 \sum_i \sum_{j,k} \max_{(H_{ji}, H_{ki})} \boldsymbol{w}^T \Phi_{ijk} \\ & + C_2 \sum_i \sum_{k,l} \max_{(\hat{y}_i, \hat{z}_i, \hat{H}_{ki}, \hat{H}_{li})} \left(\boldsymbol{w}^T \Phi_{ikl} + \Delta^{\text{hloss}}_{ikl} \right) \\ & \left. - C_2 \sum_i \sum_{k,l} \max_{(H_{ki}, H_{li})} \boldsymbol{w}^T \Phi_{ikl} \right],\end{aligned} \tag{6.10}$$

where the arguments of vectors Φ_{ijk} and Φ_{ijk} are uniquely determined by the arguments of the max operators in each term. Note that, in Equation 6.10, the second and fourth

terms headed by the "+" sign, including with first term can be grouped into a convex function of $(\boldsymbol{w}, \hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki})$. Similarly, the third and fifth terms in Equation 6.10 headed by the "$-$" sign can be grouped into a concave function of $(\boldsymbol{w}, H_{ki}, H_{li})$. This rearrangement of Equation 6.10 results in final formulation:

$$\begin{aligned}
&\arg\min_{\boldsymbol{w}} \quad \Bigg[\frac{1}{2}\|\boldsymbol{w}\|^2 \\
&+ C_1 \sum_i \sum_{j,k} \max_{(\hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki})} \left(\boldsymbol{w}^T \Phi_{ijk} + \Delta_{ijk}^{\text{hloss}}\right) \\
&+ C_2 \sum_i \sum_{k,l} \max_{(\hat{y}_i, \hat{z}_i, \hat{H}_{ki}, \hat{H}_{li})} \left(\boldsymbol{w}^T \Phi_{ikl} + \Delta_{ikl}^{\text{hloss}}\right) \Bigg] \\
&- \left[C_1 \sum_i \sum_{j,k} \max_{(H_{ji}, H_{ki})} \boldsymbol{w}^T \Phi_{ijk} + C_2 \sum_i \sum_{k,l} \max_{(H_{ki}, H_{li})} \boldsymbol{w}^T \Phi_{ikl}\right],
\end{aligned} \tag{6.11}$$

This formulation of Equation 6.11 allows us to readily use the Concave-Convex Procedure (CCCP) [57] for learning [56].

The CCCP is an optimization algorithm which can be applied to a wide range of optimization problems with the guarantee of convergence [57, 39]. To apply CCCP for optimizing an objective function, the objective function needs to be decomposed into the sum of a convex function and a concave function: $E(x) = E_1(x) - E_2(x)$, where $E_1(x)$ is a convex function and $-E_2(x)$ is an concave function. So both $E_1(x)$ and $E_2(x)$ are convex functions. An example of this decomposition is shown in Figure 6.1. The algorithm proceeds by matching points on the two functions which have the same tangents. For an input point x_0 we calculate the gradient $\nabla E_2(x_0)$ and find the point x_1 such that $\nabla E_1(x_1) = \nabla E_2(x_0)$. Then we find another point x_2 such that $\nabla E_1(x_2) = \nabla E_2(x_2)$. We repeat this process until convergence. In our case the objective function

(E) is given in Equation 6.11, which is to be minimized. We decompose it into a convex function

$$\begin{aligned} E_1 = \Bigg[& \frac{1}{2}\|\boldsymbol{w}\|^2 \\ & + C_1 \sum_i \sum_{j,k} \max_{(\hat{y}_i, \hat{z}_i, \hat{H}_{ji}, \hat{H}_{ki})} \left(\boldsymbol{w}^T \Phi_{ijk} + \Delta_{ijk}^{\text{hloss}} \right) \\ & + C_2 \sum_i \sum_{k,l} \max_{(\hat{y}_i, \hat{z}_i, \hat{H}_{ki}, \hat{H}_{li})} \left(\boldsymbol{w}^T \Phi_{ikl} + \Delta_{ikl}^{\text{hloss}} \right) \Bigg] \end{aligned} \tag{6.12}$$

and a concave function

$$E_2 = -\left[C_1 \sum_i \sum_{j,k} \max_{(H_{ji}, H_{ki})} \boldsymbol{w}^T \Phi_{ijk} + C_2 \sum_i \sum_{k,l} \max_{(H_{ki}, H_{li})} \boldsymbol{w}^T \Phi_{ikl} \right], \tag{6.13}$$

Inside CCCP, we find a hyperplane which is a upper bound of the concave function. Computation hyperplane requires latent variable inference as mentioned in Equation 6.3. So we are to compute the weighted matchings between two pairs of videos v_j, v_i and v_k, v_i where first pair of videos are more similar to each other than the videos of the second pair. However these matchings are controlled by binarizing thresholds as mentioned in Equation 5.3 in Chapter 5. These thresholds are determined such a way that, for a specific set of supervoxels of video v_i, difference between $\mathcal{A}_{ji}$ and $\mathcal{A}_{ki}$ is maximized. Here, $\mathcal{A}_{ji}$ and $\mathcal{A}_{ki}$ are affinities of video pairs v_j, v_i and v_k, v_i respectively. After computing the latent variables, rest of the problem becomes a standard Structured SVM problem which can be solved using the cutting plane method [56].

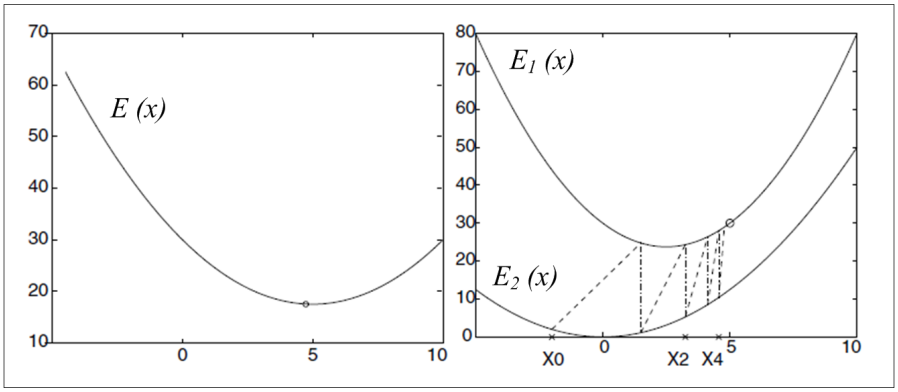

Figure 6.1: An illustration, showing how the CCCP algorithm decomposes an objective function and iteratively optimizes it [57].

Chapter 7: Results

This chapter presents our results on three datasets and comparisons with state-of-the-art methods. We perform experiments on three datasets: INRIA IXMAS [48], NIXMAS [46] and i3DPost [14]. IXMAS dataset has videos of 12 actors performing 13 daily-life actions, including checking watch (cw), crossing arms (ca), scratching head (sh), sitting down (sd), getting up (gu), turning around (ta), walking (wa), hand wave (hw), punching (pc), kicking (ki), pointing (po), picking up (pu) and throwing(th). Every action is performed three times by a single actor and each video is recorded from five different camera views: four side views and a top view. Among these 13 actions, videos for 'throwing' action is not complete. So, we consider 12 actions except the throwing. The second dataset, NIXMAS, has 11 out of 13 action classes of IXMAS dataset. The videos of this dataset were recorded from five viewpoints: four side views and a frontal view. There are objects such as tables, chairs etc., present in the background, while recording the videos. Every two out of three actions were recorded with objects which partially occlude the actors. Presence of cluttered background and occlusion makes this dataset challenging. In i3DPost there are videos of eight actors performing 10 different actions. Among these actions six are single actions *walk, run, jump, bend, hand-wave and jump-in-place* and others are combined actions *sit-stand-up, run-fall, walk-sit and run-jump-walk*. All the videos are recorded from eight calibrated camera views as shown in Figure 7.1. We choose these datasets as these are the standard datasets for evaluating

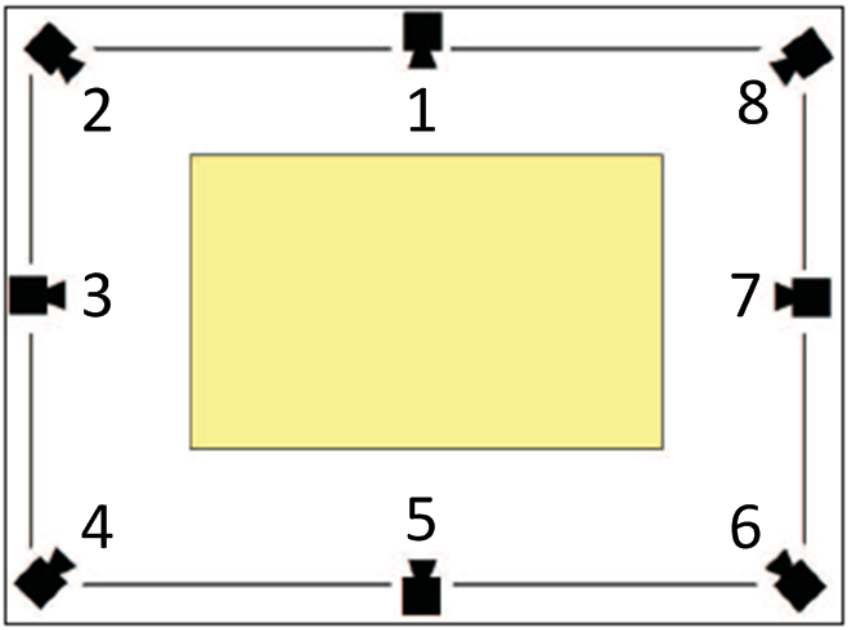

Figure 7.1: Camera settings of the i3DPost dataset.

multi-view action recognition approaches.

We select a set of exemplars for learning the feature weights of exemplars. For a single exemplar v_i, we select a set of triplet of exemplars v_j, v_k, v_l, to determine the constraints. For every exemplar, we chose 10 such triplets of closest videos v_j, v_k and v_l, to impose the constraints on the learned weight. Notice that one set of triplet v_j, v_k, v_l, gives rise to a pair of constraints: viewpoint based constraint (v_i with v_j, v_k) and action class based constraint (v_i with v_k, v_l). So we consider 20 constraints for a single exemplar.

For a target video we compute weighted affinities of all the exemplars to it. The affinities are computed based on the learned weights. Then a 'K' nearest neighbour (K-

NN) classifier with the maximum voting scheme is used to determining the action class of the target video.

7.1 Input Parameters Testing

We evaluate our approach varying three of the input parameters to examine how the performance depends on them.

Input parameters:

i. Number of training exemplars: For our experiments we choose 10, 15, 20, 25, 30 exemplars from each action class having equal number of videos from each viewpoint. Which is 6% (for 10 per class), 18% (for 30 per class) of total number of video on the IXMAS dataset [48] and 9% , 29% respectively on the NIXMAS dataset [46]. We show how the accuracy varies with increasing number of exemplars in Figure 7.2 on the NIXMAS dataset [46].

ii. K in K-NN classifier for testing: We examine how the accuracy changes with 'K', which selects the number of closest exemplars for the K-NN classifier to make the final decision. Figure 7.3 shows that accuracy increases with 'K' but saturates after K = 10.

iii. Number of supervoxels per video: We vary the number of supervoxels extracted from the videos and observe how the accuracy changes with it. Results are shown in Figure 7.4. We see that accuracy increases with the increasing number of supervoxels and then drops after reaching the highest mark. This tells us that both under-segmentation and over-segmentation fail to capture visual cues from the video

and optimal segmentation is somewhere in between them. However, performance is less insensitive to the number of supervoxels on the i3DPost dataset, due to higher resolution of videos. Highest accuracy achieved for NIXMAS, IXMAS and i3DPost datasets are 83.1%, 89.1% and 86.2% respectively. We provide results for the best parameter settings. As it is seen from the results our approach is insensitive to a wide range of input parameter variation.

7.2 Baseline Comparison

We compare our approach with three common baselines and our approach performs better than all three of them on standard datasets.

Baselines:

i. B1: Cubelets based approach: To justify the use of supervoxels, we compare our approach where regular cubelets are extracted from video are considered as features instead of supervoxels. We extract the same average number of cubelets as supervoxels from videos. Correspondence between cubelets from videos is measures using matching as in our approach. We see significant improvement of accuracy, in Figure 7.2 and 7.3, while supervoxels are used as features instead of the cubelets.

ii. B2: Without viewpoint based constraint: We perform our experiments without the viewpoint based constraint to analyse how this constraint affects the performance. It can be seen in Figure 7.2 and 7.3, that the accuracy is considerably less when we remove these constraints from learning algorithm. This shows the importance of the view information in multi-view action recognition.

iii. B3: Large margin distance learning: We compare with another exemplar based large margin distance learning framework called *large margin nearest neighbour* (LMNN)[43]. Their goal is to learn the suitable metric for K-NN classifier. They learn a linear transformation of the feature space and pose the learning as the semidefinite programming. Our 'latent' large margin distance learning framework performs better as we are finding the feature correspondence as a latent variable and then learning the weights for them. Results in the Figure 7.2 and 7.3 show the importance of considering the latent variable while learning the weights.

7.3 Comparison with State of the Art

Comparison with state of the art approaches are done on the IXMAS dataset as most of the state-of-the-art results are available on this dataset. Results on other two datasets are presented later. We compare our approach with an exemplar based approach [44], transfer learning based approaches [25] and [26], geometric model based approaches [54] and [53], view invariant feature finding approach [23], robust descriptor finding approach [46] and another latent SVM based approach [49].

We perform the experiments as *leave one actor out* as a standard experiment setting where we learn with videos from all but one actor and test on the videos of the remaining actor. We consider 12 action classes for all the experiments. Except [49], all presented approaches perform experiments on 11 classes excluding the confusing 'pointing' class. Table 7.1 shows results where videos from all viewpoints are used for training and testing is done on videos from each of the viewpoint. We marginally outperform [49] but

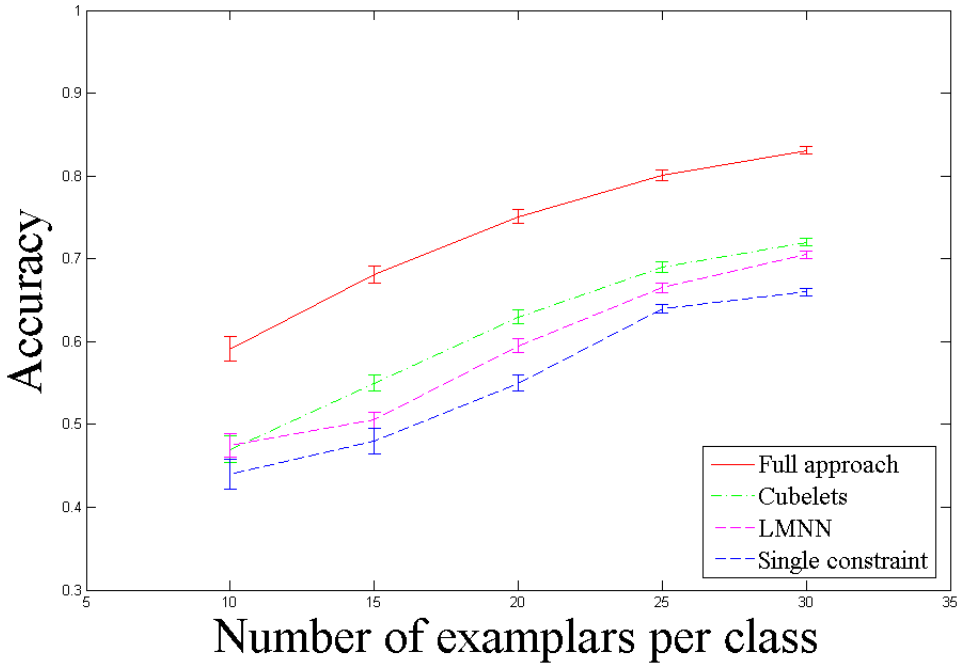

Figure 7.2: Plot of accuracy vs. number of exemplars per class for training on the NIX-MAS dataset. Here 'Cubelets' represents our approach using the cubelets instead of the supervoxels (B1), 'Single constraint' represents our approach without considering the viewpoint based constraint(B2), 'LMNN' represents our approach in an another large margin distance learning framework [43] (B3) and 'Full approach' is our proposed approach. Axes are scaled for better visualization.

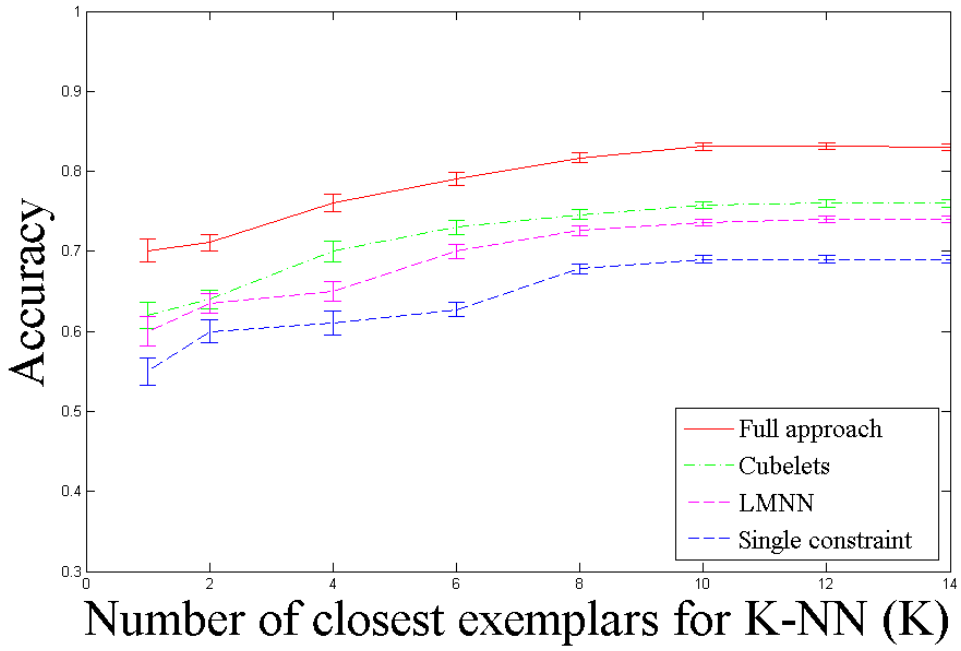

Figure 7.3: Plot of accuracy vs. number of the closest exemplars for K-NN classifier in testing on the NIXMAS dataset. Here 'Cubelets' represents our approach using the cubelets instead of the supervoxels (B1), 'Single constraint' represents our approach without considering the viewpoint based constraint(B2), 'LMNN' represents our approach in an another large margin distance learning framework [43] (B3) and 'Full approach' is our proposed approach. Axes are scaled for better visualization.

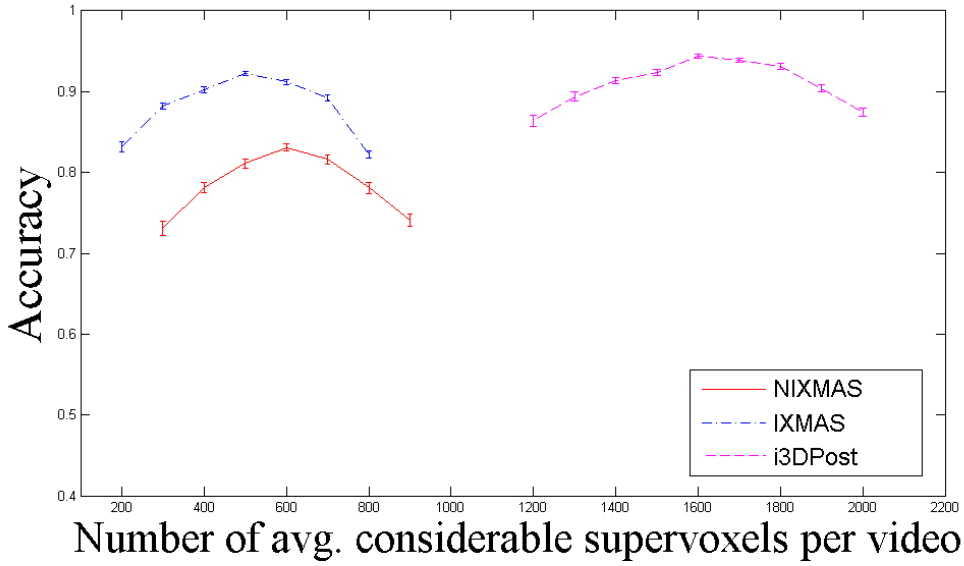

Figure 7.4: Plot of accuracy vs. average number of supervoxels per video for three datasets: NIXMAS, IXMAS and i3DPost. Axes are scaled for better visualization.

	c1	c2	c3	c4	c5	Average
[44]	65.4	70.0	54.3	66.0	33.6	57.86
[46]	86.7	89.9	86.4	87.6	66.4	83.4
[23]	74.8	74.5	74.8	70.6	61.2	71.2
[49]	**95.14**	89.58	91.67	90.28	-	91.67
Ours	94.1	**95.6**	**92.3**	**95.8**	**84.4**	**92.44**

Table 7.1: Comparison of accuracy(%) on the IXMAS dataset when training is done using all views and testing is done for a specific view. Our average accuracy 92.44 % $>$ 91.67 %, best state of the art [49]

.

	c1	c2	c3	c4	c5	Average
[25]	85.1	82.1	82.2	85.7	77.6	82.54
[26]	86.6	81.1	80.1	83.6	82.8	82.8
[49]	86.11	93.06	73.61	80.56	-	83.34
Ours	**87.21**	**93.1**	**87.3**	**88.2**	**82.1**	**87.58**

Table 7.2: Comparison of accuracy(%) on the IXMAS dataset when training is done using the four views and testing is done for the 5^{th} view. Our average accuracy 87.58 % $>$ 83.34 %, best state of the art [49]

in their case accuracy is computed without considering the challenging top view. In the similar experiment settings we are 9.04% ahead of the previous best approach [46]. Table 7.2 shows results where we perform training with videos from four viewpoints and test on videos from the other viewpoint. Finally the confusion matrix for the 12 action classes is presented in Figure 7.5.

We find informative view combinations by performing experiments where training using videos from multiple view combinations and present results in Table 7.3. Results are compared with [53] and [44]. We observe that side view combination (1,2,3) performs better than the side views with top view (1,2,3,5) together. So, we infer that the

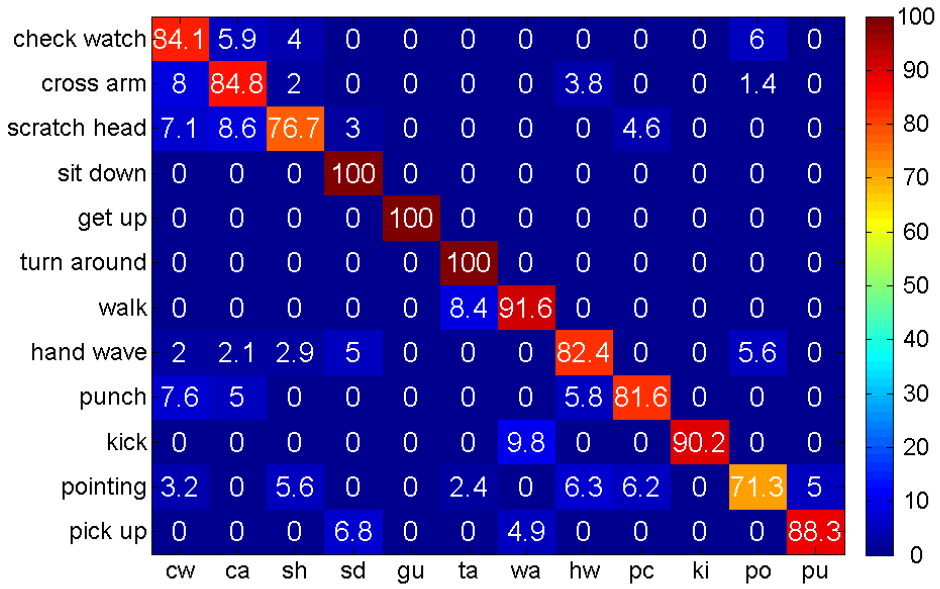

Figure 7.5: The confusion matrix for 12 actions on the IXMAS dataset

top view is not discriminative and adds confusion in recognition. Detailed recognition results for multiple view settings for the specific action classes are presented in 7.6.

7.3.1 Results on the NIXMAS dataset

On the NIXMAS dataset our average accuracy of 83.1% is 7.4% higher than the previous best approach [46]. In Table 7.4 we show results of recognition accuracy when training is done with the videos from all the viewpoints and testing is done on the videos from a particular viewpoint.

Table 7.5 presents the recognition accuracy when training is done with the videos

Cameras	2,4	1,3	3,5	1,3,5	1,2,3	1,2,3,5	1,2,3,4
[53]	71	71	-	-	60	-	78
[44]	81.3	-	61.6	70.2	-	75.9	81.3
Ours	**83.1**	**80.2**	**74.8**	**78.9**	**87.7**	**85.1**	**92.8**

Table 7.3: Recognition accuracy (%) of our approach on the IXMAS dataset for multiple camera combination.

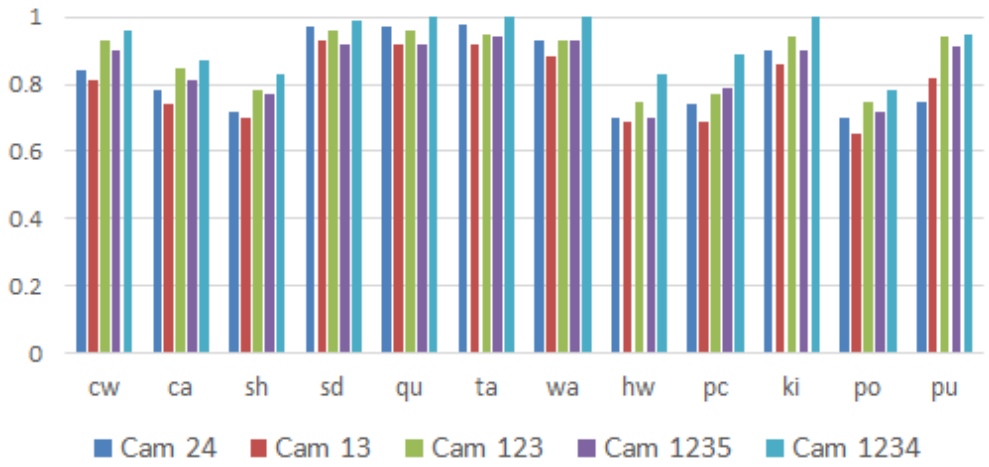

Figure 7.6: Recognition accuracy (%) for the specific action classes on the IXMAS dataset for multiple view combinations.

from four viewpoints and testing is done on the videos from the remaining viewpoint.

Recognition accuracy for multiple viewpoints are presented in Table 7.6, where training is done with the videos from specific viewpoint combinations. We see frontal views are more informative than the side views as self-occlusion is more probable in the side views. Detailed recognition results for multiple view settings for the specific action classes are presented in Figure 7.7.

Finally we present the confusion matrix for action recognition on the NIXMAS

Camera	c1	c2	c3	c4	c5
Accuracy	86.4	78.2	84.1	79.2	87.1

Table 7.4: Recognition accuracy(%) on the NIXMAS dataset when training is done using all the views and testing is done on a specific view.

Camera	c1	c2	c3	c4	c5
Accuracy	80.6	74.4	79.2	73.2	83.2

Table 7.5: Recognition accuracy(%) on the NIXMAS dataset when training is done using four views and testing is done for the remaining view.

dataset in Figure 7.8.

7.3.2 Results on the i3DPost dataset

From i3DPost dataset we choose eight actions: walk (wk), run (rn), jump in place (jp), jump forward (jf), bend (bd), sit (st), fall down (fl) and wave one hand (wo)). Our average accuracy on this dataset is 94.3 % which is 3.5% the higher than the previous best approach [22].

In Table 7.7 we present results of recognition when training is done with the videos from all the viewpoints and testing is done on videos from a particular viewpoint.

Table 7.8, shows recognition accuracy when training is done with videos from seven

Cameras	2,4	1,3	3,5	1,3,5	1,2,3	1,2,3,5	1,2,3,4
Accuracy	70.2	76.2	74.4	80.8	78.7	83.6	79.8

Table 7.6: Recognition accuracy (%) on the NIXMAS dataset when training is done with multiple camera combinations.

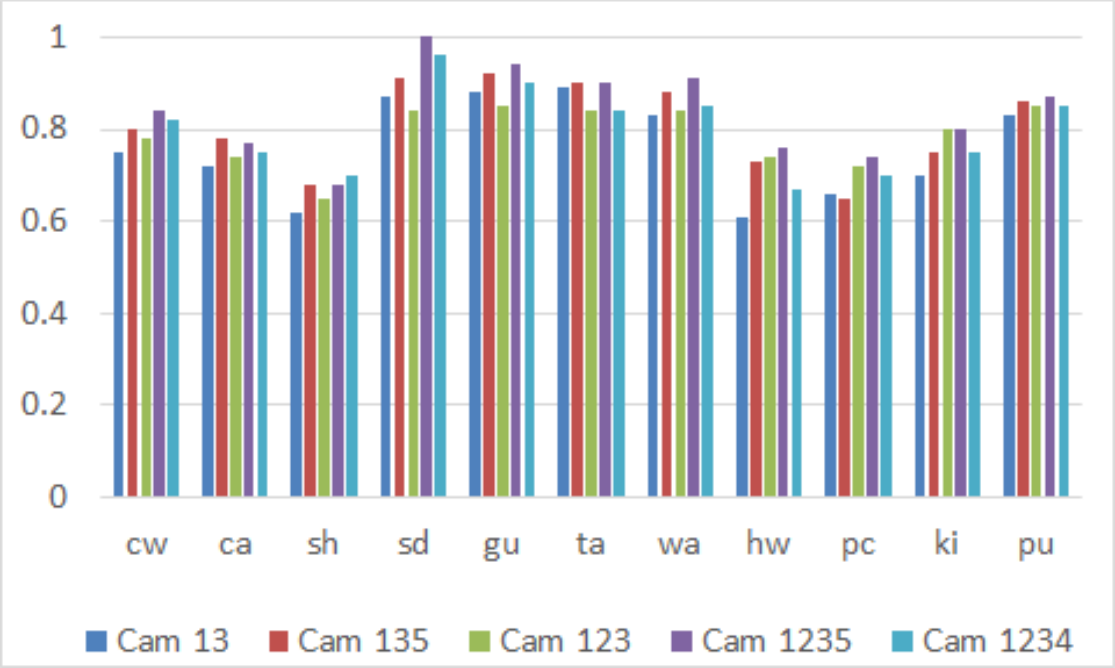

Figure 7.7: Recognition accuracy (%) for the specific action classes on the NIXMAS dataset for multiple view combinations.

viewpoints and testing is done on videos from the remaining viewpoint.

Results for multiple viewpoints are presented in Table 7.9, where training is done with the videos from a specific viewpoint combinations. Here also we see the frontal views are more informative than side and back views due to presence of occlusion in side and back views. Detailed recognition results for multiple view settings for specific action classes are presented in Figure 7.9.

Finally we present the confusion matrix for action recognition on the i3DPost dataset in Figure 7.10.

We demonstrate the learned weights for supervoxels of the videos from different ac-

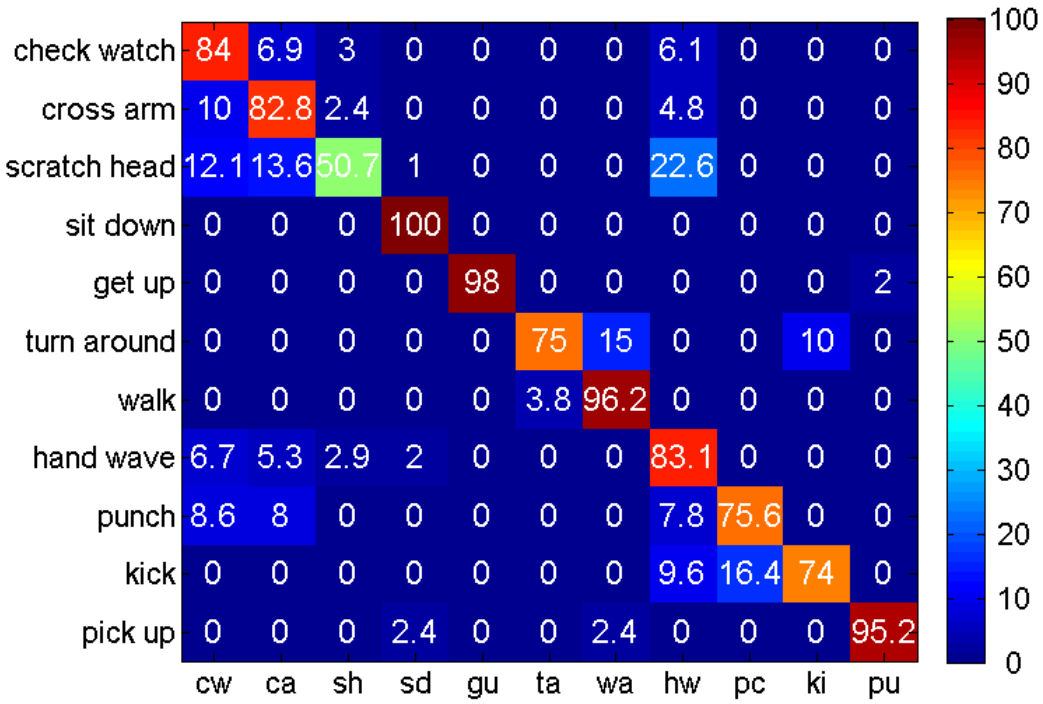

Figure 7.8: The confusion matrix for 11 actions on the NIXMAS dataset. Average recognition accuracy is 83.1 %.

tion classes in Figure 7.11. For better visualization we show only selected supervoxels having distinctive weights. The weight of a supervoxel reflects its importance in action recognition. In the first column, (a) and (b) are the frames corresponding to same action, kick, from left view and right view respectively. As it shows, different set of weights are learnt for the same action, recorded from different viewpoints. This justifies the importance of view information for multi-view action recognition. In both of the cases supervoxels corresponding to moving leg, are assigned higher weights. Supervoxels

Cam.	c1	c2	c3	c4	c5	c6	c7	c8
Acc.	97.4	97.0	95.7	93.4	86.3	92.8	95.5	96.7

Table 7.7: Recognition accuracy(%) on the i3DPost dataset when training is done using all the views and testing is done on a specific view.

Cam.	c1	c2	c3	c4	c5	c6	c7	c8
Acc.	94.8	92.7	91.1	88.2	82.8	87.6	90.8	93.8

Table 7.8: Recognition accuracy(%) on the i3DPost dataset when training is done using seven views and testing is done on the remaining view.

correspond to hands are assigned due to the movement of hands while performing the 'kick' action. In (c) our algorithm fails to differentiate weights of the supervoxels corresponding to upper and lower portions of the body. In this action of 'check watch', the whole body remains static except the left hand. Due to the similar appearance, similar weights are learnt for the supervoxels from entire body. In (d), 'kick' action is captured from the top view. In this case, left arm remains partially occluded. Hence, even the arm is moving during the action, weight of supervoxels corresponding to that arm are less. For frames from i3DPost dataset (e and f), due to higher resolution of the videos, supervoxels are extracted more accurately and thus, distinct weights are learned for the body parts, according to their importance. We conclude from the shown results that supervoxels, correspond to moving body parts, are usually assigned higher weights. But

Cameras	1,2,8	4,5,6	2,3,4	6,7,8	1,2,3,7,8
Accuracy	90.8	79.8	86.1	87.4	94.8

Table 7.9: Recognition accuracy (%) on the i3DPost dataset for multiple camera combination.

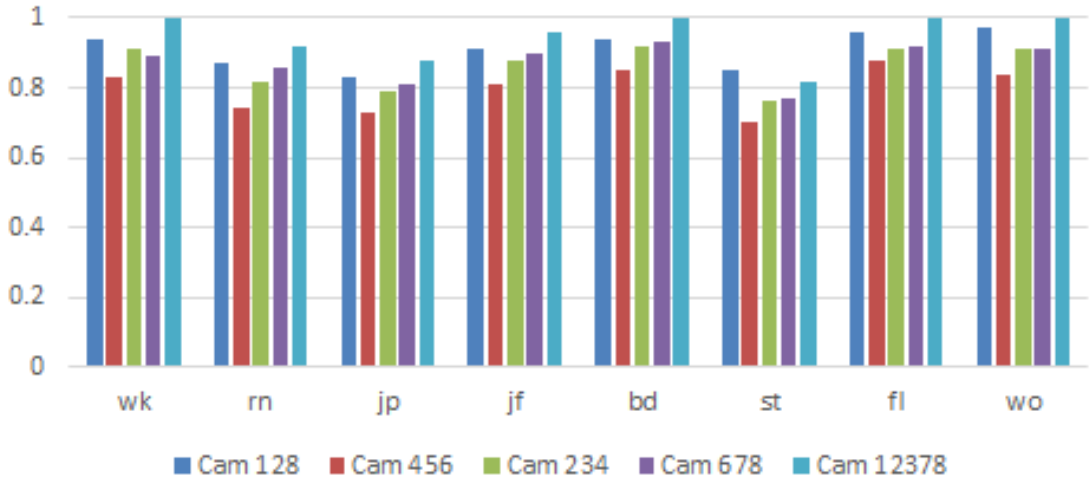

Figure 7.9: Recognition accuracy (%) for the specific action classes on the i3DPost dataset for multiple view combinations.

weights are less for occluded supervoxels.

It is interesting to see, how our approach performs on a test video from particular viewpoints, which are not considered for training. If both of the action class and viewpoint of a test video are present in training, then the test video is expected to match with videos from same action class and viewpoint. This is due to the large margin learning of weights guided by viewpoint and action class based constraints. But if the viewpoint of the test video is not seen in training, classification becomes a chance game. All the exemplars give low matching score for that test video while classifying by K-NN. But we hope, in this case, test video is better matched with the exemplars from same class instead of other classes. This is due to the presence of action class based constraints in learning. It can be seen from Table 7.1 and 7.2, average recognition accuracy on the

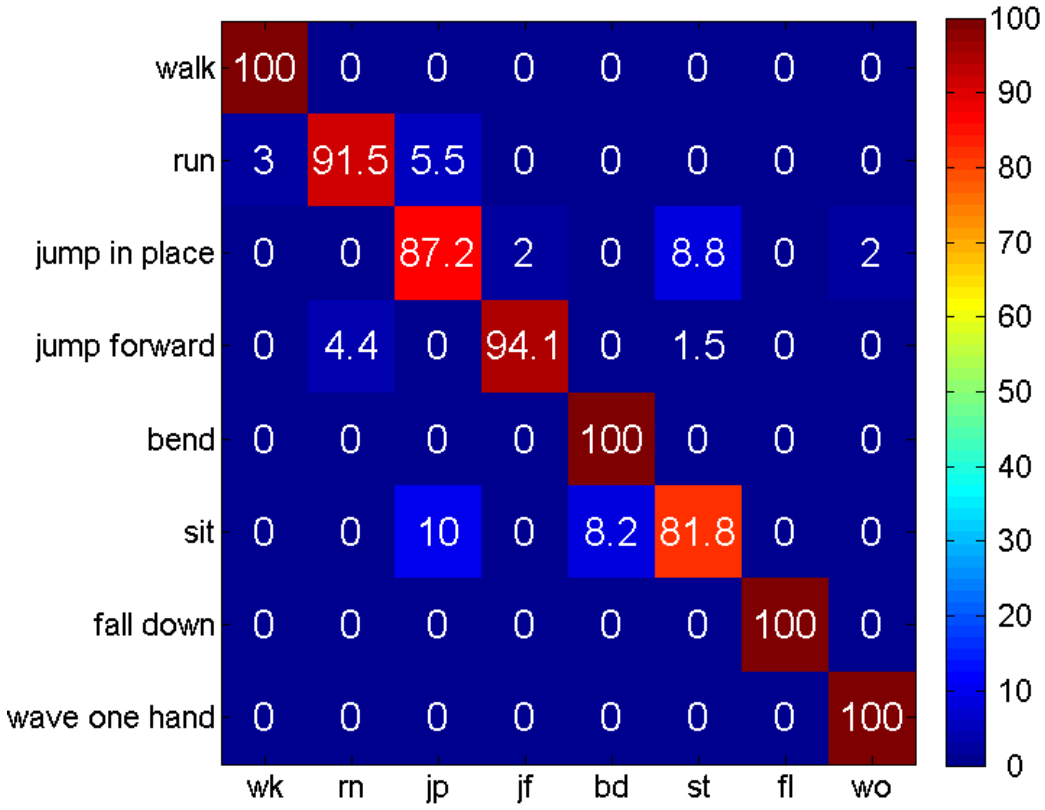

Figure 7.10: The confusion matrix for eight actions on the i3DPost dataset. Average recognition accuracy is 94.3 %.

IXMAS dataset is less when any particular view is not considered in training. Similar changes of recognition accuracy can be noticed on the NIXMAS dataset (Table 7.4 and 7.5) and on the i3DPost dataset (Table7.7 and 7.8).

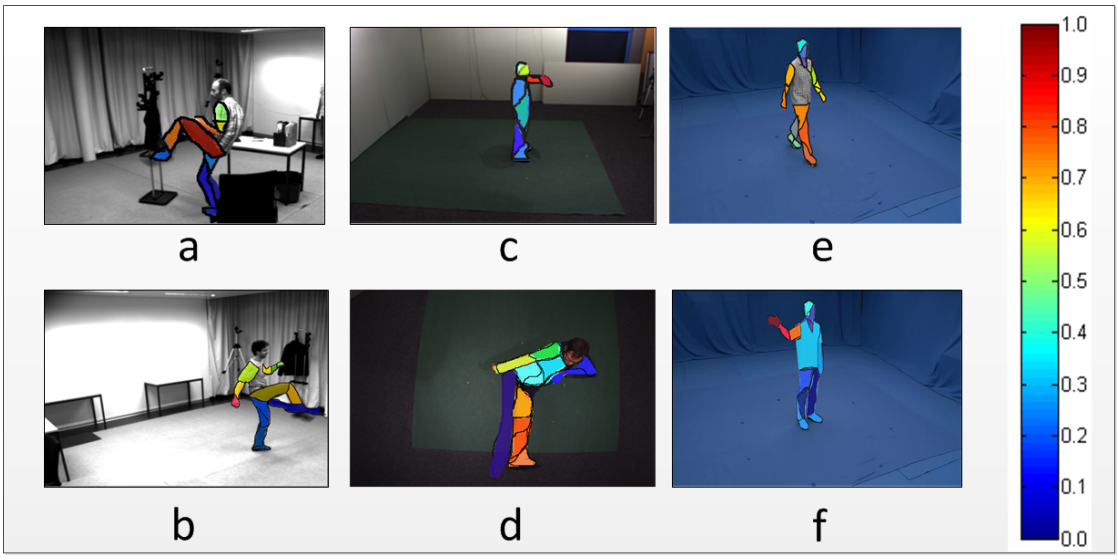

Figure 7.11: Visualization of weights, learned for supervoxels for specific action classes. Weight are defined by colors as in Matlab jet scale. Here (a), (b) are frames of the 'kick' action from NIXMAS dataset. (c), (d) are frames of 'check watch' and 'kick' actions respectively, from IXMAS dataset. And (e) ,(f) are frames of 'walking' and 'hand-wave' actions respectively, from i3DPost dataset. (Best viewed in color)

Chapter 8: Conclusion

In this thesis, we have addressed a fundamental computer vision problem, that of action recognition. The goal of action recognition is to identify the class of action performed in a video. The main challenges for action recognition is that actors move and change their pose while performing the action, and/or that the actors may be captured from different viewing angles. To address these challenges, we have developed a multi-view approach for action recognition. Our approach is able to recognize different classes of actions from the videos, which were recorded from multiple viewpoints.

We have specified an exemplar-based approach for view-invariant action recognition. We have formulated a novel latent large-margin learning of the K-NN, subject to a set of viewpoint based and action class based constraints. The main challenge for multi-view action recognition is that, *different* actions from the *same* viewpoint may appear very similar and thereby creating confusion in recognition process. Hence, we have introduced a novel viewpoint based constraint to address this issue. A mid level feature - supervoxel, which has been used to capture visual cues from the videos. Then matching is employed to find correspondence between the two set of supervoxels extracted from two videos. This matching is treated as the latent variable in our large margin learning framework.

We have done extensive experiments on three standard datasets: INRIA IXMAS, NIXMAS and i3DPost. Our approach has outperformed the state-of-the-art by 9.04%

on the IXMAS, by 7.4% on the NIXMAS and by 3.5% on the i3DPost. We have evaluated all the input parameters such as number of training exemplars, average number of supervoxels per video and number of closest exemplars (K) considered for K-NN classification. We have concluded, from the results, that our approach is robust against a wide range of parameter variations. Our approach has also outperformed the common baselines. Comparison with the 'Cubelets based approach' (B1) has justified the use of the supervoxels as mid level features. Effectiveness of including the additional viewpoint based constraints is proved as our approach has outperformed the approach 'Without viewpoint based constraint' (B2). To validate the contribution of matching of supervoxels as a latent variable in recognition process, we have compared our learning method with another large margin distance learning framework LMNN(B3) [43]. We have concluded from the baseline study that the 'viewpoint' based constraint is crucial as accuracy drops significantly if we do not consider these constraints in learning (Figures 7.2 and 7.3).

When we exclude some of the viewpoints in the training, recognizing videos from those viewpoints becomes a chance game. We have noticed that recognition accuracy is less when any particular view is not considered in training. This holds for all three datasets. Though, due to the presence of action based constraints, recognition accuracy is satisfactory for all three datasets (Table 7.2, 7.5, 7.8).

We have compared our approach with the state-of-the-art approaches, and in most cases our approach outperformed them. In rare cases we do not outperform the state-of-the-art methods when the supervoxels become noisy. Another cause of errors is that we consider a subset of all possible constraints while learning the feature weights of the

exemplars. This reduces the learning time.

Some of the limitations of our approach include, dependency on the supervoxel segmentation and time complexity of the learning algorithm. We use supervoxels as the features, so any segmentation error while extracting supervoxels could effect the performance of our approach. Also, ideally, there are exponentially many constraints to be considered while learning the feature weights of the exemplars. As this is not practicable due to its time complexity, we only consider a subset of closely violated constraints, which result in efficient learning of the weights. To address these issues, one might consider a better method for supervoxel extraction and more sophisticated framework for learning the weights. These are beyond the scope of this thesis.

Bibliography

[1] JK Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.

[2] Dhruv Batra, Tsuhan Chen, and Rahul Sukthankar. Space-time shapelets for action recognition. *Motion and video Computing, 2008. WMVC 2008*, pages 1–6, 2008.

[3] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *ICCV*, 2:1395–1402, 2005.

[4] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267, 2001.

[5] R. Chaudhury, A. Ravichandran, G. Hager, and Rene Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. *CVPR*, 2009.

[6] Shinko Y Cheng and Mohan M Trivedi. Articulated human body pose inference from voxel data using a kinematically constrained gaussian mixture model. *CVPR*, 55, 2007.

[7] I. Cohen and H. Li. Inference of human postures by classification of 3D human body shape. *AMFG*, 2003.

[8] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8, 2002.

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[10] Somayeh Danafar and Niloofar Gheissari. Action recognition for surveillance applications using optic flow and SVM. *ACCV*, pages 457–466, 2007.

[11] A. Farhadi and M. Tabrizi. Learning to recognize activities from the wrong view point. *ECCV*, 2008.

[12] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. *ICCV*, 2007.

[13] Andrea Frome, Yoram Singer, and Jitendra Malik. Image retrieval and classification using local distance functions. *NIPS*, 19:417, 2007.

[14] N. Gkalelis, H. Kim, A. Hilton, N. Nikolaidis, and I. Pitas. The i3DPost multi-view and 3d human action/interaction database. *CVMP*, 2009.

[15] Amir Globerson and Sam T Roweis. Metric learning by collapsing classes. *NIPS*, pages 451–458, 2006.

[16] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. *NIPS*, 2005.

[17] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. *CVPR*, 2010.

[18] Michael B Holte, Thomas B Moeslund, Nikos Nikolaidis, and Ioannis Pitas. 3d human action recognition for multi-view camera systems. *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*, pages 342–349, 2011.

[19] Michael B. Holte, Thomas B. Moeslund, Cuong Tran, and Mohan M. Trivedi. Human action recognition using multiple views: A comparative perspective on recent developments. *J-HGBU*, 2011.

[20] Somboon Hongeng and Ramakant Nevatia. Large-scale event detection using semi-hidden markov models. *ICCV*, pages 1455–1462, 2003.

[21] K. Huang and M. Trivedi. 3D shape context based gesture analysis integrated with tracking using omni video array. *In CVPR Workshops*, 2005.

[22] Alexandros Iosidis, Nikos Nikolaidis, and Ioannis Pitas. Movement recognition exploiting multi-view information. *MMSP*, 2010.

[23] I. Junejo, E. Dexter, I. Laptev, and P. Perez. View-independent action recognition from temporal self-similarities. *PAMI*, 33(1):172–185, 2011.

[24] I. Laptev. On space-time interest points. *In IJCV*, pages 107–123, 2005.

[25] R. Li and T. Zickler. Discriminative virtual views for crossview action recognition. *CVPR*, 2012.

[26] Jingen Liu, Mubarak Shahz, Benjamin Kuipersy, and Silvio Savaresey. Cross-view action recognition via view knowledge transfer. *CVPR*, 2011.

[27] Behrooz Mahasseni, Sheng Chen, Alan Fern, and Sinisa Todorovic. Detecting the moment of snap in real-world football videos. *IAAI*, 2013.

[28] M Ángeles Mendoza and Nicolás Pérez De La Blanca. Applying space state models in human action recognition: A comparative study. In *Articulated Motion and Deformable Objects*, pages 53–62. Springer, 2008.

[29] I. Mikic, M. M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *IJCV*, 53(3):199–223, 2003.

[30] K-R Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf. An introduction to kernel-based learning algorithms. *Neural Networks, IEEE Transactions on*, 12(2):181–201, 2001.

[31] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. *ECCV*, pages 392–405, 2010.

[32] Vasu Parameswaran and Rama Chellappa. View invariance for human action recognition. *IJCV*, 66(1):83–101, 2006.

[33] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[34] Ronald Poppe and Mannes Poel. Discriminative human action recognition using pairwise csp classifiers. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–6. IEEE, 2008.

[35] Ariadna Quattoni, Sybor Wang, L-P Morency, Michael Collins, and Trevor Darrell. Hidden conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1848–1852, 2007.

[36] C. Rao, A. Yilmaz, and M. Shah. View-invariant representation and recognition of actions. *IJCV*, 50(2):203–226, 2002.

[37] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. *NIPS*, 16:41, 2004.

[38] Cristian Sminchisescu, Atul Kanaujia, and Dimitris Metaxas. Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, 104(2):210–220, 2006.

[39] Bharath K. Sriperumbudur and Gert R. G. Lanckriet. On the convergence of the concave-convex procedure. *NIPS*, 2009.

[40] Du Tran and Alexander Sorokin. Human activity recognition with metric learning. *ECCV*, pages 548–561, 2008.

[41] Ivor W Tsang, Pak-Ming Cheung, and James T Kwok. Kernel relevant component analysis for distance metric learning. *Neural Networks, IJCNN'05.*, 2:954–959, 2005.

[42] Namrata Vaswani, Amit K Roy-Chowdhury, and Rama Chellappa. Shape activity: a continuous-state HMM for moving/deforming shapes with application to abnormal activity detection. *Image Processing, IEEE Transactions on*, 14(10):1603–1616, 2005.

[43] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.

[44] D. Weinland, E. Boyer, and R. Ronfard. Action recognition from arbitrary views using 3D exemplars. *ICCV*, 2007.

[45] Daniel Weinland and Edmond Boyer. Action recognition using exemplar-based embedding. *CVPR*, pages 1–7, 2008.

[46] Daniel Weinland, Mustafa Özuysal, and Pascal Fua. Making action recognition robust to occlusions and viewpoint changes. *ECCV*, 2010.

[47] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Automatic discovery of action taxonomies from multiple views. *CVPR*, 2:1639–1645, 2006.

[48] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *CVIU*, 104(2-3):249–257, 2006.

[49] Xinxiao Wu and Yunde Jia. View-invariant action recognition using latent kernelized structural SVM. *ECCV*, 2012.

[50] Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. *NIPS*, 2002.

[51] Chenliang Xu and Jason J. Corso. Evaluation of super-voxel methods for early video processing. *CVPR*, 2012.

[52] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. *CVPR*, pages 379–385, 1992.

[53] P. Yan, S. M. Khan, and M. Shah. Learning 4D action feature models for arbitrary view action recognition. *CVPR*, 2008.

[54] A. Yilmaz and M. Shah. Actions as objects: A novel action representation. *CVPR*, 2005.

[55] A. Yilmaz and M. Shah. Recognizing human actions in videos acquired by uncalibrated moving cameras. *ICCV*, 2005.

[56] Chun-Nam John Yu and T. Joachims. Learning structural svms with latent variables. *ICML*, 2009.

[57] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, pages 915–936, 2003.