2012

Robust Real-Time Recognition of Action Sequences Using a Multi-Camera Network

Rahul Ratnakar Kavi
West Virginia University

Robust Real-Time Recognition of Action Sequences Using a Multi-Camera Network

by

Rahul Ratnakar Kavi

Thesis submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2012

Keywords: Camera Networks, Machine Learning, Action Recognition, Video Surveillance, Computer Vision

Abstract

Robust Real-Time Recognition of Action Sequences Using a Multi-Camera Network

by

Rahul Ratnakar Kavi
Master of Science in Computer Science

West Virginia University

Vinod K. Kulathumani, Ph.D., Chair

Real-time identification of human activities in urban environments is increasingly becoming important in the context of public safety and national security. Distributed camera networks that provide multiple views of a scene are ideally suited for real-time action recognition. However, deployments of multi-camera based real-time action recognition systems have thus far been inhibited because of several practical issues and restrictive assumptions that are typically made such as the knowledge of a subjects orientation with respect to the cameras, the duration of each action and the conformation of a network deployment during the testing phase to that of a training deployment. In reality, action recognition involves classification of continuously streaming data from multiple views which consists of an interleaved sequence of various human actions. While there has been extensive research on machine learning techniques for action recognition from a single view, the issues arising in the fusion of data from multiple views for reliable action recognition have not received as much attention. In this thesis, I have developed a fusion framework for human action recognition using a multi-camera network that addresses these practical issues of unknown subject orientation, unknown view configurations, action interleaving and variable duration actions.

The proposed framework consists of two components: (1) a score-fusion technique that utilizes underlying view-specific supervised learning classifiers to classify an action within a given set of frames and (2) a sliding window technique that is used to parse a sequence of frames into multiple actions. The use of a score-fusion technique as opposed to a feature-level fusion of data from multiple views allows us to robustly classify actions even when camera configurations are arbitrary and different from training phase and at the same time reduces the required network bandwidth for data transmission permitting wireless deployments. Moreover, the proposed framework is independent of the underlying classifier that is used to generate scores for each action snippet and thus offers more flexibility compared to sequential approaches like Hidden Markov Models. The amount of training and parameterization is also significantly lower compared to HMM-based approaches. This Real-Time recognition system has been tested on 4 classifiers which are Linear Discriminant Analysis, Multinomial Naive Bayes, Logistic Regression and Support Vector Machines. Over 90% accuracy has been achieved by this system in Real-Time recognizing variable duration actions performed by the subject. The performance of the system is also shown to be robust to camera failures.

Acknowledgements

First, I want to thank my committee chair and advisor, Dr. Vinod K. Kulathumani, for guiding me in the my research and providing me the opportunity to work with him and his other graduate students. This thesis work has been made possible with his constant support and guidance. I also want to thank Dr. Arun A. Ross and Dr. Xin Li for being a part of the my committee. I have had discussions with them which were important in my understanding of identifying and soliving certain research oriented problems in my Thesis.

I would like to thank my current co-workers for helping me out in collecting the data-set for this work. I want to thank my previous co-workers Sriram Sankar and Sree Charan with whom I've had the pleasure of working with. They have been extremely helpful, supprotive in buidling my understanding of the subject. I've learnt loads from them in discussions with them and also the valuable code debugging sessions we've had together.

Last but not the least, I want to express my gratitude to my family. My parents have been very encouraging on my decision to go to grad school. Their support has been relentless and a constant motivation to my desire of pursuing Computer Science in school.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter a brief overview of real-time action recognition is presented. The issues encountered in single-camera based surveillance and how they can be solved using multiple cameras is presented. A brief overview of challenges in realizing an automated multi-camera real-time action recognition system is discussed. Then the contributions made to solve certain challenges posed in multi-camera network are presented.

1.1 Overview

Automated human action recognition is a widely studied topic. It has got wide range of applications from surveillance, gaming, monitoring parking lots, sign language learning, augmented reality and applications in smart home environments.
Its a widely studied area where ideas from computer vision, machine learning and distributed systems congregate. One can categorize human actions into three categories i.e., *gestures*, *actions* and *activities* [1]. Gestures can be considered as a very short duration actions. Example of a gesture can be signs in American Sign Language (ASL). Activities can be considered as complex actions and long interleaved actions. Example of an activity would be opening a door, entering a room, pulling out a chair, sitting on the chair in an interleaving manner. Recognizing human actions can be done using motion sensors, cameras, infrared sensors, pressure sensors, etc. However in this study, we deal with human action recognition using a camera network.

Single camera based surveillance systems have been very popular in the past. Researchers have extensively worked on single camera based human action recognition for the past two decades. However, they completely fail in certain cases where hardware malfunction of the camera occurs or in case of an occlusion in the scene. The occlusion can be due to an object in between the sight of the camera and a subject or due to self-occlusion by the subject. In order to solve this problem multiple camera systems have been proposed. However, these cameras have to be monitored by a human operator for effective use.

Multiple views of a scene provide a better representation of a scene. It solves the problem of failure of cameras, incomplete view of an action (self-occlusion by the subject for certain actions), and other occlusions in the view. At the same time, it presents a very complex scenario where the approach to handle the problem has to process more data than in a single view. Action recognition systems that use a single camera based approach can be prone to failure. Multiple cameras can definitely help. An efficient and automated real-time visual surveillance can be realized through a network of cameras monitoring a scene. But, one doesnt get a chance to see many working applications in the real world. One cannot design a camera network with same set of principles that apply to traditional single camera based surveillance system. One of the reasons for this is that, there are many challenges that need to be addressed before deploying a highly efficient multi-view automated real-time action recognition system.

A few of the challenges posed in deploying multi-camera network for real-time surveillance of human activities is presented. In a camera network, in order to retain maximum information about a scene, one cannot overloadthe network by transferring raw data to be monitored at the central server or the base-station. One has to extract relevant information in an image of the scene from a view and only transfer this information to the base-station. Only relevant information from a video stream must be exchanged in order to come to a collective decision across multiple cameras. To solve this problem, the system will need some kind of local processing power at the camera nodes. One cannot deploy powerful servers at every camera, so low power processors need to be used and they should be able to handle automated real-time surveillance. The information processing module at local camera nodes need to be highly efficient, at the same time consume less processing power in order to retain high frame rate. We need to avoid computational overload of data by information processing module as it will lead to low frame capture rate and during this phase, the subject may have left the scene.

One cannot assume to have access to a large buffer while processing data, which can be used by the underlying classification system to recognize human actions in a camera network. This is because; we have to handle a large stream of image data at each of the camera. Our classification technique has to be a light weight approach that has access to a small buffer of data collected in real-time. The classification strategy has to take advantage of all cameras effectively.

The surveillance system cannot assume the pose of the subject in the scene. The real-time action recognition framework has to deal with subjects facing arbitrary directions. Symmetry between training and testing phases should not be assumed as in a real-time scenario it may not be possible to deploy cameras in such manner. Multiple cameras sample data at different rates making it tougher to generate a consistent feature vector in order to identify an action performed by the subject. The information processing strategy chosen at the camera node must be able to handle variable number of frames. The information extraction strategy and classification algorithm must go hand in hand in order to solve this problem. The classification algorithm plays a vital role in determining the performance of the system. We need to choose an algorithm that is computationally feasible at the camera nodes (with low processing power) and the algorithm must be able to clearly distinguish between various actions that a subject may be performing the scene. The system needs to be independent of the underlying classification strategy used or at least provide a very wide range of classification algorithms at disposal.

In the real-time scenario, the system doesnt know the exact starting and ending points of an action. Thus, feature vector extracted from the video stream of unknown length has an impact on the manner in which images are captured and classified. Some strategies to solve this problem require extracting feature vectors from each frame; others extract features from a short video sequence. This remains an important decision that needs to be made while building a real-time action recognition system. The real-time multi-view action recognition system should also be able to easily plug-in into other software modules that are primarily dependent on visual surveillance. For example, activity recognition (longer duration actions) would be dependent on making decisions on series of interleaved actions and making sense out of them.

This work presents strategies to solve certain challenges posed by the automation of real-time multi-view action recognition. Before we attempt to solve these problems, we make following assumptions to make the problem easier to solve and present a clear picture of how our system works. Our work assumes that a single subject is present in the center of 8 cameras and only moving in a restricted area to perform an action. Currently, the system only supports actions performed by a single subject in a scene. We focus on developing a framework that can perform interleaved action recognition in real-time where the subject performs an action in a controlled environment of multiple cameras connected over a network. This application can be deployed on a thin client or on an embedded platform.

In this work, a setup of 8 cameras is used which were synchronized over the network using NTP protocol. These cameras are setup in the along the walls facing the center of the room where the subject is performing an action. This software framework developed using above described setup and configuration can be used for real-time action recognition. With appropriate changes made to this framework, it can be extended to activity recognition in a controlled environment. Extension of this work to activity recognition will require an abstract layer, which will keep a track of series of action being performed and recognizes an activity based on classified actions. Context free grammars can be used [2] to model subject to subject interactions, series of actions performed by a subject in a scene.

1.2 Summary of Contributions

This section primarily discusses about main contributions of this work. There is a lot of research conducted in the area of action recognition however; a lot of these are in the area of single view action recognition. Also, most of existing techniques ignore or make a lot of assumptions that are not trivial while deploying the system in a multi-view scenario in real-time. The main contribution of this work is to identify actions performed in a scene with high accuracy.

This framework developed can be deployed in real-time and identify variable duration actions. This work can be extended into an activity recognition system by making minimal changes to existing architecture. The framework can distinguish between actions with high accuracy as well as determine if an action performed is not one of the actions on which it was trained. The system was trained in an offline manner using view-specific classifiers. The recognition is done by using classification scores from multiple cameras. The system can be deployed in a real-time setting where the subject is performing an action and the system is simultaneously classifying the actions performed in real-time. The system was trained on 9 sets of *unit action*s. Each subject has performed the action 10 times in a sequential and interleaved manner. 4 such sets of data were collected. The system was trained offline with 20 sets of unit samples per action. The real-time testing of the system was performed offline by providing the algorithm with originally collected continuous stream of data. The cameras were synchronized using the NTP protocol while collecting the data.

1. Real-time action recognition framework: the framework developed with this work can be used in real-time multi-view action recognition. We have previously demonstrated that this framework can be deployed in real-time [3] with high accuracy. The current framework can handle continuous stream of data with the help of sliding window technique [4] by sampling the stream of data with pre-defined window sizes. Previous system [3] was not able to handle this case. The highest likelihood action is then selected from the set of samples after the classification algorithm has been applied. The highest likelihood action is the action classified at the local node. This information can

then be fused using the fusion rule to make a unanimous decision across the network. Then the window moves to the next set of frames and this process is continued.

2. Efficient local processing: supervised learning techniques were employed along with a sliding window technique [4] to make a decision on captured data. A series of frames are captured at each of the local nodes and the feature descriptor is built. The feature descriptor is a weighted motion energy image [5]. The weighted motion history image is divided into an arbitrary grid of size 7 x 7 and sum of pixels is calculated at each block. The grid size was selected in a heuristic manner. This yields a 49 length feature vector that captures most of the information that is needed to make a classification on the captured data. This process is done using OPENCV framework [6]. Our previous work [3] only handled a constant length feature vector (of length 70 frames). In this work, we demonstrate that variable length frame duration doesnt primarily affect the feature vector generation process and still give high accuracy without usage of graphical modelling techniques such as *Hidden Markov Models* [7].

3. Information fusion from multiple cameras: in our previous work [3], we have demonstrated this information fusion strategy works for real-time deployment. Each camera node makes its own decision regarding the captured data. The captured data has its own time stamp. The local decision made at the camera node consists of the maximum likely action and its associated probability. This information can be exchanged over the network with its respective time stamp information. The central server then receives information from 8 cameras regarding the captured frames maximum likely action and its associated probability. This strategy was used in an offline mode and information is fused over multiple cameras (summed up) and a collective decision is made.

4. Diverse options of machine learning algorithms: this framework has been tested with *four* machine learning algorithms. Theoretically, any machine learning algorithm whose output can be represented in form of a probability/likelihood can be used. This framework was tested with *Linear Discriminant Analysis*, *Multinomial Naive Bayes*, *Logistic Regression* and *Support Vector Machines*. This has been made possible with

the usage of sliding window technique [4]. This works extends our previous work [3] which was tested only with Linear Discriminant Analysis with Nearest Neighbors.

5. Recognize untrained actions: We define *untrained actions* as *Standing* or other actions that are performed by a subject, but are not used in training of the system. This framework has successfully demonstrated that it can negatively identify actions. The system was trained using 9 actions (bowling, clapping, jogging, jumping, kicking, pickup, standing, throwing, 1 hand waving and 2 hand waving). The system was successfully able to recognize standing action (on which it was not trained).

1.3 Organization of rest of the thesis

The remaining part of this document is arranged as follows. Background information and existing literature is covered in the 2nd chapter. Overview and system description is provided in the 3rd chapter. In 4th chapter implementation details, offline recognition results, recognition results on streaming data are provided. In the 5th chapter (final chapter) a brief analysis and possibility of extension of this work to activity recognition and future work is discussed.

Chapter 2

Background work and existing literature

This chapter discusses the existing research work in the area of machine learning, distributed systems and computer vision applied to human action recognition. The popular approaches to applying machine learning techniques to streaming data are then discussed. After the discussion on existing research, the important aspects that differentiate this work from others are discussed.

2.1 Approaches to feature extraction from videos

Feature vectors contain vital information of a scene which needs to be monitored. Therefore, it is very important on how we choose feature vectors for a given scenario. Depending on various assumptions of the scene to be monitored, researchers have considered modeling the feature vectors based on the following methods [8].

2.1.1 Volumetric methods

Spatio-Temporal filtering approaches: In these approaches, spatio-temporal filter banks are applied on the video sequence. These methods consider entire video or a partial video as a single data entity [8]. Then, information is extracted on pixel by pixel basis on the entire video. For a given video, features are extracted from *V(x,y,t)* where *V* is the video (set of frames) of time duration t and x, y are pixel coordinates (on a frame by frame basis). Local appearance models using *Gabor Filter banks* (in many orientations) can be computed for a given video *V(x,y,t)* to identify actions [9]. Researchers have also considered local histograms of normalized space-time gradients at many temporal scales [10]. It is interesting to note that filtering approaches are easy to implement as efficient algorithms for convolutional are available [8].

Part based approaches: Video can be considered as a collection of local parts. These local parts may follow a distinctive motion pattern. These patterns can be tracked over space-time to recognize actions. Laptev and Lindeberg have applied Harris interest point detector on video to recognize actions [11]. In this work [12], Neibles used space time interest points along with bag of word model to recognize actions in an unsupervised manner. Interest points in a video can be used along with *Support Vector Machines* (SVM). Laptev and Schuldt[13] have used space-time features along with SVM to classify human actions such as walking, jogging, running, etc. These approaches are sensitive to noise, view variance and occlusion. Also in some cases finding these interest points may be harder due to similarity between background and human subject. Certain actions and smooth human actions which are hard to perceive will not give rise to distinctive features in the video sequence which are required to classify the human actions[14].

Sub-Volume based approaches: *Haar-Features* have been successfully applied in object detection in still images. Researchers have also used 3D Haar type features (volumetric features) to analyze video [15]. Responses obtained from these filters have been used along with ensemble machine learning techniques such as boosting to improve recognition performance of the system. In the work done by Shechtman et al [16], it was shown that correlation in space-time motion can be used to match actions with a template. Simple actions like walking, waving, jumping, etc. were recognized. It should be noted that this technique is not scale invariant and rotation invariant. Variations in intensity due to changing background can affect performance of sub-volume based approaches. These techniques however, can be extended to be invariant to changes in appearance over time with use of other techniques such as optical flow [15].

Tensor based approaches: *Tensors* can be considered to be generalization of matrices to multiple dimensions. In tensors, space-time volume i.e. *V(x,y,t)* can be considered as tensors with 3 independent dimensions. Other information such as identity of a subject can also be encoded. Vasilescu [17] modeled human action, subject identity and joint angle trajectories as independent dimensions of a tensor. A dimensionality reduction technique was then applied to extract relevant data and it was used for recognition.

2.1.2 Non-Parametric methods

These types of methods work on a frame by frame basis and extract features from them. These features are matched to a stored template. Bobick and Davis [18] were one of the first to introduce the concept of motion energy images (MEI) and motion history images (MHI) by observing a video sequence for a short amount of time. MHI and MEI capture the region in the video sequence which was active in the video sequence by looking the video on frame by frame basis. They can be obtained by using a simple threshold technique and obtain series of blobs. An entire video sequence can be reduced to a single image using MHI and MEI. They are really good in discriminating smaller actions such as sitting, standing, etc. but not so good in handling longer duration complex activities. From the given MHI and MEI, scale invariant, translation invariant features such as Hu-moments [19] can be calculated and be used along with other algorithms to match a specific video sequence.

A video sequence containing an action can be considered as an object/subject performing an action. This data is represented in form of 2d in an image over series of images. The outer contour of the object can be projected onto 2d over time giving rise to a volume in spatio-temporal space (x,y,t). These features are called as *spatio-temporal volume* (STV) [20]. Geometric properties of the surfaces are studied and feature descriptors are calculated. These feature descriptors can be used for action recognition. Simple actions such as falling, running, kicking, standing, sit-down, etc. can be recognized.

In many cases, the techniques used to better represent the features in a video sequence are of very high dimensions. Dimensionality reduction techniques such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are often used to reduce dimensionality of the feature descriptors.

2.1.3 Parametric methods

Parametric methods impose a specific model on the temporal dynamic on the motion in the video. These methods have been successful in modeling the actions and activities in a scene to a large extent.

Hidden Markov Models have been widely used and are one of the most popular parametric methods. Hidden Markov Models are composed of states (which are hidden and cannot be seen) and these states output observation symbols (which can be seen). Transition probabilities (which model state to state transitions) and observation probabilities have to be estimated initially.

Baum-Welch algorithm is used to train parameters of an HMM. Then one can use *Viterbi Decoding* algorithm to estimate state transitions. An HMM is trained per action and maximum likely HMM (which can be chosen by using forward-backward technique) can be chosen as most likely action. Yamato [7] has demonstrated that states can be considered analogous to frames. It was demonstrated that in certain cases recognition accuracy was as high as 96%. HMMs have quite been successful in recognizing actions with varying duration. Hidden Markov Models and its variants have also been used for gesture recognition [21] [22] and complex action (using coupled HMMs) recognition [23].

Conditional Random Fields can be considered an extension of Hidden Markov Models and have received considerable attention recently. Conditional random fields solve some of the problems like label bias problem. However, they are computationally expensive to train [4].

2.2 Approaches to process streaming data

Processing streaming data is a different challenge altogether as the algorithms have to handle video of arbitrary length. Traditional methods do not always address this issue. This aspect is discussed in the following techniques.

2.2.1 Sliding window techniques

Sliding window techniques employ a moving window of static or a dynamic sized which contains a series of frames (video sequence of certain length). The data contained in this window is transformed to a feature vector using traditional techniques to extract feature vectors from the video sequence. In cases of windows of arbitrary length (or dynamic) the feature vectors may be of very different value compared to the ones with static length on the same video sequence. One needs to be careful in choosing the feature vector that can be computed in time varying window. The technique to match these feature vectors also is very vital in lassification of the given video sequence. The prime advantage of this technique is that given a feature vector, any machine learning algorithm can be used for classification of the video sequence [4]. A series of frames gives rise to a single feature vector. This feature vector is classified as an action. However, relationship between different consecutive classifications is not exploited.

2.2.2 Graphical models based techniques

Hidden Markov Model is a popular statistical technique that can classify sequential/streaming data. In order to use them, one needs to properly define the structure of the model, estimate transition probabilities, estimate observation probabilities then train and test the designed Hidden Markov Model on given parameters. Different initial parameters make HMM converge to local minima, thus we obtain different accuracies [24].

Yamato [7] has demonstrated that image frames can be considered as states and the series of observations in a HMM can be mapped to a specific action. A HMM can be trained for each of the actions. Then for a given observation sequence O, $P(O|\lambda_i)$ is predicted where i is the set of parameters for ith HMM. Then for the given observation sequence, maximum likely HMM predicts the given action. i.e., Action = $argmax P(O|\lambda_i)$.

Representation and inclusion of diverse data belonging to same class of data has profound effect on performance of Hidden Markov Model to use it for action recognition [7]. The process of representing the states of a Hidden Markov Model is affected by inclusion of diverse data (by use of standard vector quantization). Conditional random fields are yet another way to classify sequential data which address some limitations posed by Hidden Markov Models but, they are expensive to train [4].

2.3 Related work

This section describes on how this work is different from the existing research work. Lots of research has been conducted in the area of action recognition. Most popular methods being non-parametric methods and graphical model based methods.

Through this work it is demonstrated, how a non-parametric approach based spatio-temporal technique can be used with multiple views of a scene in a camera network to perform lightweight but highly accurate action recognition. Extensive research has been carried out in Single Layered Approaches to human action recognition [25]. Single layered approaches directly use the data to make decisions on the action classification. Spatio-Temporal approaches have been very popular way of classification of human actions [26][11].

This work demonstrates multiple views of the same action in a scene add to better performance of the action recognition system. A weighted motion energy image captures the regions in a human blob silhouette which contain most activity. This feature can be calculated easily and can be used for classification of human actions. For a given video sequence, the algorithm has access to blob-silhouette of the subject in the scene. Frame differencing is applied to these images and the resultant image is added over time. The resultant feature descriptor now contains the regions of the blob-silhouette that covers the entire region of the image where change was observed as well as the magnitude of change in each pixel.

Many researchers have worked on multiple view action recognition lately [27][28][29][30]. Work presented in [29] [30] generate feature vectors that can be invariant to any point. We believe that addition or removal of a view may affect the feature vector as different actions are portrayed differently in different views. Also a case may arise, where one has retrained the system if a view is removed. This work doesnt suffer from the above mentioned pitfalls.

Graphical model based methods have received a lot of attention lately due to their inherent property of handling video sequences of arbitrary length and their ability to model complex relationships between multiple variables . HMMs treat each frame individually to represent a state. Series of states are mapped to series of observations. Classifications are made based on correlation between observations. This work captures correlation at the point of generation of feature descriptor. Our feature descriptor captures regions in an image that have been changed over time as well the frequency of change of pixels over time. Structure of the graphical model is vital in performance of the system. Hidden Markov Models performance is dependent on initialization parameters, structure, direction of traversing, etc. and the model converges to local minima [31]. Unlike Hidden Markov Models, this system does not lay high emphasis on the structure of the underlying classifier. Once a feature descriptor is generated, rest is taken care by the classifier.

View specific probabilistic classifiers were chosen (i.e. Classifiers whose output can be determined in terms of probability of belonging to a certain class) in this work. It has also been demonstrated that by placing a threshold on the probability, one could easily distinguish a non-action from the set of actions with which the system was trained. In this work, the computational complexity of generating the feature vector is low and classification is low. The prototype system takes about *0.8 seconds* to generate a feature vector (which is fairly high) and about *0.015 seconds* to fuse and classify the feature descriptor. We strongly believe that this will further decrease once the system is implemented in C/C++ giving rise to faster action classification system.

Detecting events/abnormalities in a scene based on pixel intensities were performed in [32] which performed fairly on surveillance videos. This technique captures events at pixel level. In a weighted motion energy image, highly active pixels in a video sequence are given higher weight than the other parts. This pixel level activity is fairly unique for the given actions in the dataset. In a way, the weighted motion energy image from which feature descriptor is extracted captures the shape of the blob silhouette, magnitude of change at pixel level. Clapping can be distinguished from 2 hand waving, etc. However, this system may not distinguish between rubbing of hands and performing gestures with 2 hands.

Using a probabilistic classifier along with this feature descriptor at each camera node, this system has successfully demonstrated that non-actions can be clearly distinguished from actions with high level of accuracy. The dataset consists of 4 sets of data, in each of which 2 subjects are performing 10 actions each. The system was able to distinguish in real-time if the action perfectly overlaps the sliding window of certain size from the same action partially overlapping the sliding window with an indication on likelihood (fused probabilities from 8 cameras).

In our technique there is no need for symmetric deployment of the cameras in training and testing phase. We are essentially applying 8 classifiers in a circular manner (from which 8 possible configurations arise). We need to pick the configuration which has the highest likelihood (highest fused score among 8 possible ways). This was previously demonstrated in [3]. Since we are identifying actions performed at atomic level, this framework naturally lends itself to activity recognition system which can recognize longer term activities by making minimal change to existing architecture.

Chapter 3

Overview of the system and description

In this chapter, overview of the system is provided. Description of experiment setup, feature descriptor extraction from multiple views, fusion of information to make a classification is provided. This chapter also includes a brief overview of underlying classification algorithms used in building the system.

3.1 System Model

In this subsection, an overview of the implemented system model is provided. A total of $N_c = 8$ cameras are taken in this current setup. These N_c cameras provide an view of a area of R from different viewing angles. Any two consecutive cameras when chosen provides an overlapping view of a scene. We assume that the relative orientations between the cameras are known in the system. For the sake of simplicity we assume the cameras are deployed in a symmetric manner as shown in Figure 3.1. However, in the real-time scenario, the camera setup doesnt have to be the symmetric.

The subject is assumed to be standing in the center of the region R. The subject is then assumed to be performing one of the $N_a = 9$ actions. The N_a actions are mentioned in the Table 3.1. It is also assumed that there is only 1 subject in the Region R. This limitation is brought is discussed in Future Work in Section 5. It can be relaxed if one uses

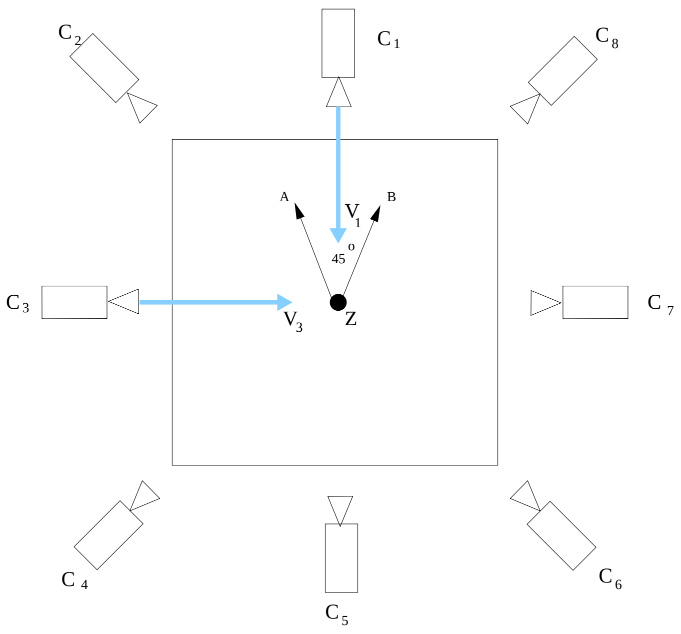

Figure 3.1: The 8 camera layout used in the experiment. The subject is shown at location Z and Subject can be at the center of the square region.

better segmentation algorithm or technique to isolate subjects in a given area. The subject performs series of interleaved actions of certain duration. The subject performs an action in a small area Z, which is roughly at the center of the Region R. For a given Action N_i it is assumed that all *unit actions* belonging to it, need not be of same duration. It also may vary from camera to camera. It is important to notice this fact as different cameras process frames at different rates.

In the training phase, the subject performs series of interleaved *unit actions*. The duration of these actions is in the range of 3 to 5 seconds. Any two *unit actions* need not be exactly the same duration. Actions such as Jumping or Jogging usually last about 5 seconds and smaller actions such as clapping last about 3 seconds. The N_c cameras were setup along the boundary of a Region R of size 50 feet x 50 feet in a closed environment. Each of the cameras are setup at height of 8 feet from the ground. The cameras are denoted by C_i where $1 \leq i \leq 8$. The cameras are setup in a manner that the actions performed by the subject are captured properly in all of the 8 cameras.

Figure 3.1 depicts the camera deployment used in the Training phase. The cameras are assumed to be symmetric arrangement. The experiments are carried out using Logitech

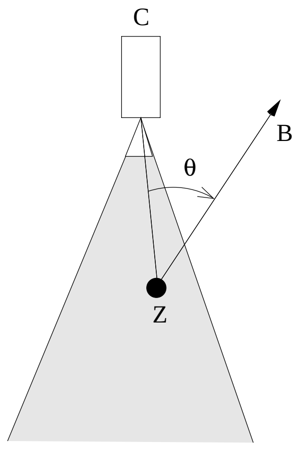

Figure 3.2: The view-angle of camera C with respect to action being performed. A subject is standing at point Z and performing an action A_a while facing direction shown by ray ZB.

9000 USB cameras. The cameras capture frames ranging from 15 fps (frames per second) to 20 fps. The frame size was set to 960 x 720. A camera node consists of a PC (based on Intel Atom processor) connected with the Logitech 9000 USB camera. These camera node also consist of a wireless card (802.11 Wi-Fi) which is used for transmitting data over the network.

The view-angle of a camera C_i is defined as the angle made by optical axis of the camera with direction along which subject performs the action sequence [3]. The view-angle of camera C_i is measured in a clock-wise manner from the ray originating from location of subject performing the action parallel to optical axis of the camera [3]. A camera view angle is depicted Figure 3.2 For the sake of simplicity, we have N_v sets of view angles. A $N_v = 8$ is used in this experiment. During the training phase, the subject is expected to perform series of interleaved *unit actions* facing a certain camera. The view-angle sets are denoted with V_j $(1 \leq j \leq 8)$ which is depicted in Figure 3.3. A view V_j of the subject performing an action in Region R, is provided by the camera C_i.

In a real-time deployment of the system, one can vary the number of Cameras and the position of the cameras depending on the requirement. Not all cameras are required to be active; some may be switched off or may fail during operation. The performance with some cameras switched off is provided in Section 4. If two cameras are placed very close to each other facing the subject, it can be assumed that the cameras provide the same view V_j.

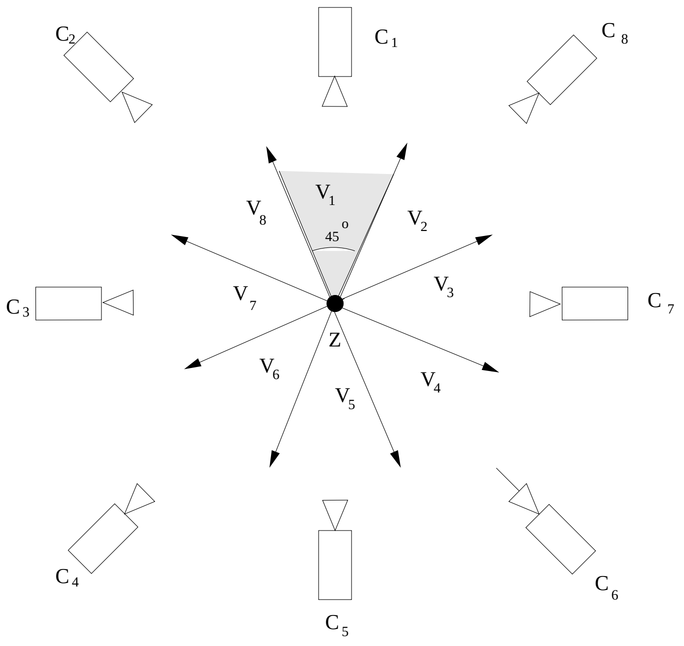

Figure 3.3: Depicts 8 view-angle sets for camera C_i with respect to the action being performed. The subject is located at point Z and could be facing any direction. View-angles of camera C_i are grouped into 8 sets as shown in the Figure.

3.2 System description

This section describes the system setup that has can be used to design a real-time action recognition system. This section can be broadly categorized as Collection of training and testing data, Extraction of feature descriptors from data.

3.2.1 Collection of training and test data

A series of image frames are collected for the training phase with the subject standing in the location Z in Region R. The subject can face any camera in the given camera network. Camera C_1 is taken as a reference. The subject performs a series of interleaved actions facing camera C_1. Training data for each *unit action* is collected at a given view-angle belonging to V_1 with respect to camera C_1. Each of the camera C_i ($\forall i : 1 \leq i \leq 8$) provides a view corresponding to view V_j ($\forall j : 1 \leq j \leq 8$). A video sequence called *unit action* is extracted per action, which exactly contains the action performed (starting point of the action to the ending point of the action only). A subject performs a total of 10 *unit actions* per Action in

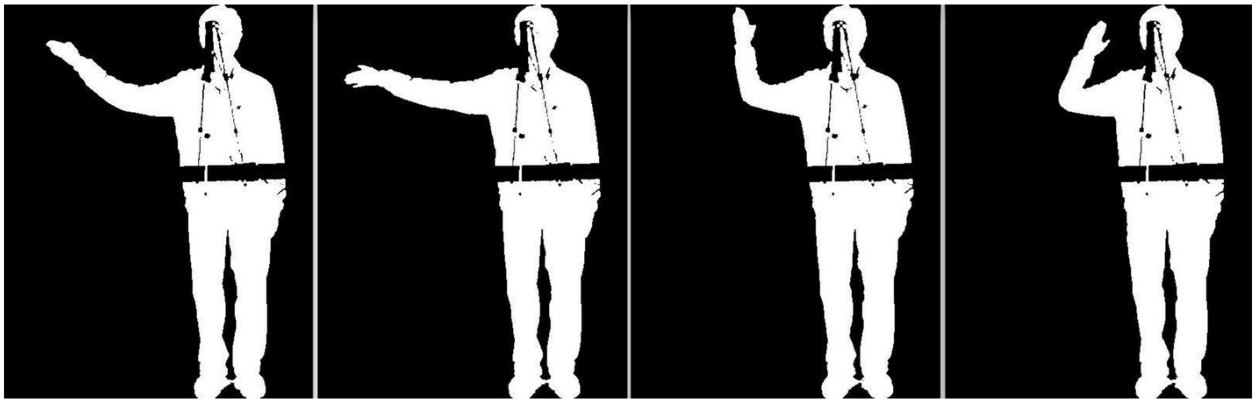

Figure 3.4: Extracting local feature descriptors. A bounding box that encloses all background subtracted silhouette is drawn around each silhouette. Only binary information is retained for each block in a grid of 7 x 7. For each block in motion energy of image of the video sequence, sum of pixels in each block is stored.

a given set. These actions are interleaved with a gap of 2-3 seconds each. Once the training data is collected, a short sequence of frames belonging to an *unit action* is extracted. Four sets of data are extracted from two subjects. Each set contains 9 unique actions and each of them has 10 *unit actions*. So we have about 40 *unit actions* per action performed by subjects on which system could be trained and tested. This system is trained offline with 4 classifiers namely *2-Class Linear Discriminant Analysis*, *Logistic Regression*, *Multinomial Naive Bayes* and *Support Vector Machine*. The training and testing set each contain about 20 feature vectors extracted from 20 *unit actions*.

3.2.2 Extraction of feature descriptors from data

In this section, we first discuss the extraction of feature descriptors from training data, testing data and then we discuss extraction of feature descriptors from streaming data. Training data consists of lots of *unit action* samples. It should be noted that duration of each action varies from action to action, subject to subject and sample to sample. It should be assumed that no two samples of *unit action* performed by a subject are of exact same duration. Therefore, we should choose a feature descriptor along with a suitable classification technique that can uniquely identify the action sequence performed which is of variable window size.

Figure 3.4 shows a partial video sequence of blob silhouette obtained from camera C_1. For a *unit action* video sequence of length L, a maximum bounding box is drawn. These blob silhouettes are added with each other (being of same size now) resulting in a weighted motion energy image. Each weighted motion energy image is then divided into a grid of 7 x 7 sized boxes. Sum of pixels in each grid box is computed and a 49 length feature vector is obtained. Similarly all *unit action* video sequences are transformed into a 49 length feature vector. Now our training and testing set consists of 20 feature vectors for each of the 9 actions. It was observed that different actions perfomed varied in length. Also for subjects performing the same action twice, it was observed that lengths may not be equal. Table 3.1 consists of Actions with which system was trained and their associated symbols.

Symbol	Action
A_1	Waving 1 arm (right)
A_2	Waving 2 arms
A_3	Jogging
A_4	Kicking
A_5	Pickup
A_6	Jumping
A_7	Clapping
A_8	Bowling
A_9	Throwing

Table 3.1: Action List

In the real-time streaming mode, for a given video stream of indefinite length, we capture the frames with above mentioned window sizes in an incremental manner. Feature vector for each of those windows is computed and classified. Then starting frame is incremented by a size of average duration of classified action. We describe the classification approaches in the next subsection.

3.3 Classification Algorithms

This subsection briefly describes each of the classifiers used in this system and then how a collective decision is made for a given video sequence. After obtaining a feature vector from the weighted motion energy image, it is then fed into the view specific classification algorithm at each camera. The classification algorithm outputs the classified action and its associated probability. We look at *Linear Discriminant Analysis*, *Multinomial Naive Bayes*, *Logistic Regression* and *Support Vector Machine* classifiers in respective order as follows:

3.3.1 Linear Discriminant Analysis (LDA)

It is a statistical dimensionality reduction technique. The goal of LDA classification technique is to represent data in lower dimensions where different classes of data can be clearly separable (in Euclidean space). For a given data of N classes of data, inter-class or in-between class scatter (S_b) and intra-class or within-class scatter (S_w) is calculated. For the data to be differentiable clearly, we need S_b/S_w as large as possible. Then a projected vector y = w^T x is obtained which is of reduced dimensionality (*N-1* dimensions). Projected vectors in reduced dimensionality for sample data are illustrated in Figure 3.4. So, the goal of LDA training phase is to find weight vector w for each class (in a 2-class LDA).

Training a 2-class LDA classifier (where *N=2*) for a given class of data is performed as follows:

1. Calculate S_b.
2. Calculate S_w.
3. Calculate Eigenvectors of (${S_w}^{-1}$ S_b) and arrange eigenvectors in descending order.
4. Weight vector w = first *N-1* eigenvectors.
5. Project training data for a given class into lower dimensions. Projected data = dot product (w^T, training data).
6. Calculate positive class cluster center (PC_i) and negative class cluster center (NC_i) for the given class C_i.

Now classification for a feature vector belonging to an unknown class can be classified using the weight vectors obtained in training phase. The feature vector is projected

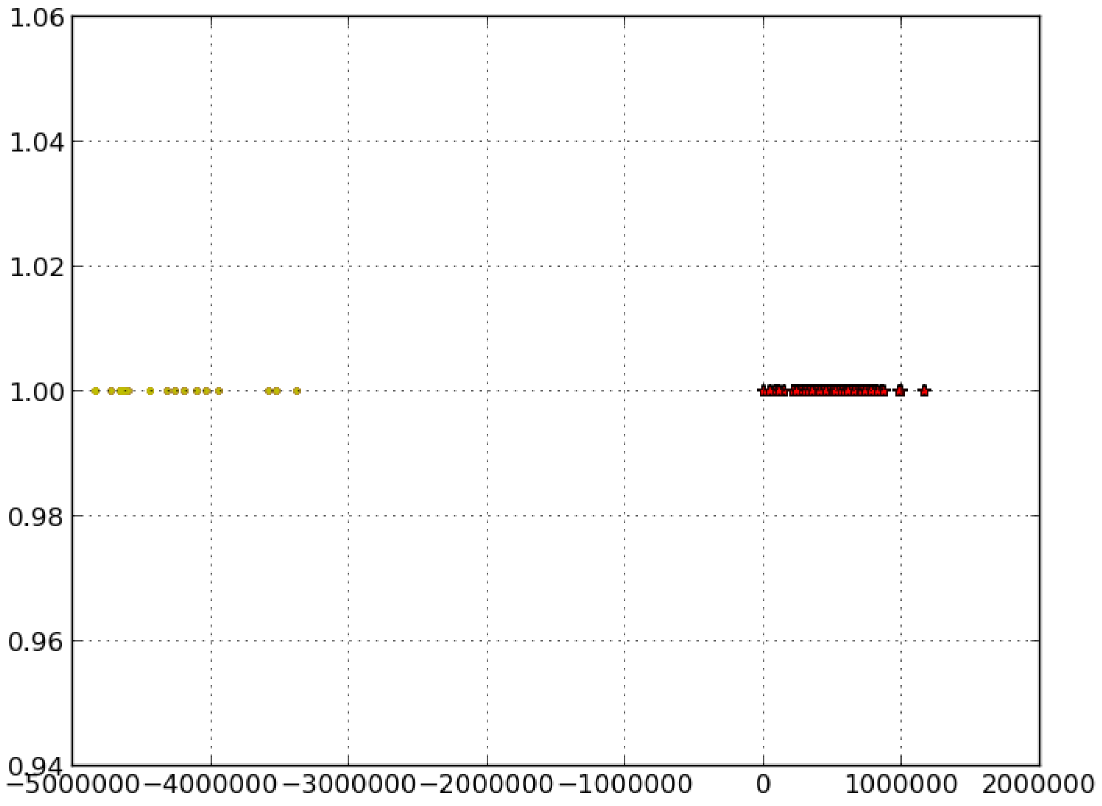

Figure 3.5: LDA project vectors plotted on a graph. On the left (in yellow) is 1 hand waving feature vectors projected in 1 dimension. On the right is rest of the actions projected in 1 dimension.

into lower dimensions and Euclidean distance to cluster centers (PC_i and NC_i) is calculated. The data is classified to belong to certain class to whose Euclidean distance is the least (i.e., Nearest Neighbor). The Euclidean distance can be normalized to [0, 1] to represent it as a probability where 0 represents least likely and 1 represents most likely. In our case, for each action class A_a in each view, a two-class Linear Discriminant Analysis based projection vector is obtained by grouping together data belonging to that particular action against data from all other actions corresponding to the respective view. During the process of training 2-class LDA classifier for a given action A_a, the weight vector wa is obtained. The dot product of 49 length feature (of dimension 49 x 1) vector and transpose of weight vector wa (wa is of dimension 49 x 1) will give rise to projected vector of dimension 1.

Cluster centers are calculated for each class (PC_a and NA_a for positive and negative clusters repsectively) for a given action A_a is calculated in reduced dimensions. Let $\lambda_{a,j}$ correspond to the LDA projection vector corresponding to A_a ($\forall 1 \leq a \leq N_a$) using data from view V_j ($\forall 1 \leq j \leq 8$). In the testing phase at each camera node, for the unknown feature vector of length 49, feature vector is projected to lower dimensions assigned to the class (A_a) to whose Euclidean distance is the least in reduced dimensionality. This is done by computing a dot product of the feature vector and $\lambda_{a,j}$. Then the classified action and its associated probability (normalized Euclidean distance) are obtained.

3.3.2 Multinomial Naive Bayes (MNB)

It is a Bayesian classifier which is quite popular in natural language processing [33] domain (in classifying documents, spam filtering, etc.). From a natural language processing perspective, *Multinomial Naive Bayes* (MNB) works by calculating relative word frequency (number of times a certain word x_i appears) in a document d (or in a text) and using this information to classify an unknown document into one of the classes C_i.

Let $P(C_j)$ represents prior probability of a class C_j. Then P($x_i|C_j$) represents the likelihood of certain word x_i belonging to the class C_j. In the training phase, P(C_j) is obtained for each class and P$(x_i|C_j)$ is obtained for each word x_i. Let frequency of word x_i in a given class be Fx_i. Let frequency of all words in the given class be T_c. Let U_w be the total number of unique words. Now for a given document d_i (feature vector) whose classification is unknown, we can say the document belongs to the maximum a posteriori class:

$$C_{map} = argmax_{C_j \in C} P(C_j) * \prod_{i=1}^{n} P(x_i|C_j)$$

where

$$P(x_i|C_j) = \frac{N(x_i, C=C_j)}{N(C=C_j)} = \frac{Fx_i}{T_C}.$$

In certain cases, we may observe that a word x_i may not appear in the document. In such a case P$(x_i|C_j) = 0$ and this makes the likelihood to be zero. We can solve this problem by using Laplacian smoothing. We can modify the definition of P$(x_i|C_j)$ from frequency of word x_i in class C_j as follows:

$$P(x_i|C_j) = \frac{Fx_i+1}{T_C+U_w}.$$

In our action classification problem, document is analogous to the feature vector and class is analogous to action A_a. In the training phase we obtain prior probabilities of each action A_a and likelihood of each unit of the feature vector belonging to certain class A_a. Using this information, at each camera node a feature vector of length 49 is passed to MNB classifier. The classification output at each camera node and its associated probability is obtained. MNB is can be used as a multi-class classifier naturally. So, in our case we train a MNB classifier for all the actions (classes) at a specific camera/view. So let MNB_j be the Multinomial Naive Bayes classifier trained at view j. We would have 8 classifiers trained in 8 views in the current implementation.

3.3.3 Logistic Regression (LR)

Logistic Regression is a classification technique that is quite similar to linear regression (line fitting). The goal of Logistic Regression is to learn a hypothesis $h_\theta(x)$ for the given data.

Where we have $h_\theta(x) = g(\theta^T x)$ and $g(z) = \frac{1}{1+e^{-z}}$ and $x \in X$ and $(0 \leq h_\theta(x) \leq 1)$.

In a machine learning problem, a cost function defines the degree of dissimilarity between a predicted value and original value. The goal of a classification or regression technique in machine learning is to minimize the cost function. The process of finding the vector is an optimization problem and can be solved by minimizing the cost function using gradient descent or other advanced optimization algorithm like BFGS, L-BFGS, etc. The cost function for which we need to optimize the value of in Logistic Regression classification is defined as:

$$J(\theta) = \sum_{i=1}^{m}[(-y^i * log(h_\theta(x^i))) - ((1 - y^i) * log(1 - h_\theta(x^i)))]$$

Where m is the total number of training examples, y is target label that needs to be learnt. Let represent the learnt hypothesis. The value of is optimized by minimizing the above cost function. Then classification is performed by setting a threshold thresh, where if $h_\theta(x) \leq thresh$ then x^i is classified as 0(if negative) and if $h_\theta(x) \leq thresh$ then x^i is classified as 1(if positive). It should be noted that Logistic Regression is a binary classification technique.

Therefore to solve a N class problem, we need to group data as one vs rest of the classes in training phase and train a Logistic Regression classifier per class. In the testing phase, we pick the classifier that maximizes the probability: $argmax_i\ h^{(i)}\theta(x)$. In our problem of action classification we train a classifier per action A_a in the above mentioned technique. At each camera node for the feature vector of length 49 (which is of unknown action), we pick the action that maximizes the probability $argmax_a\ h^{(a)}\theta(x)$. Finally, the classified action and its associated probability are obtained. Logistic Regression classifier available in Sklearn [34] was used. Logistic Regression classifier can be used as a multiclass classifier (using one versus all strategy). We train a single classifier for all the actions at a specific camera/view. So, let LR_j be the Logistic Regression classifier trained at view j. We then have 8 classifiers for all actions LR_j $(1 \leq j \leq 8)$.

3.3.4 Support Vector Machines (SVM)

Support Vector Machine (SVM) based approach is a popular classification technique, which have received much attention in past decade [35]. SVM attempts maximize the separation between two classes of data in higher dimensions by drawing a hyperplane. The hyper plane can be a linear or polynomial hyper plane. The data points in higher dimensions that lay close to the hyper plane the best separates 2 classes of data are called support vectors. The goal of the classifier (linear SVM) can be re-stated as finding weight vector w such that

$$Y = w.x + b$$

Where x is the feature vector to be classified and b is a constant. We can then classify using the following statement:

$$\text{If } w.x + b >= +1 \text{ then } y = +1.$$
$$\text{If } w.x + b <= -1 \text{ then } y = -1.$$

The value of weight vector w is found using advanced optimization algorithms. For our case of action recognition, in the training phase the weight vector w is obtained. In the test phase, the classification is made using above mentioned equations. It should be noted that SVM only work for 2 class classification. So during training, we obtain a classifier for each class. SVMs are a classification technique which output class of the input feature vector and dont output the probability. One can obtain probability in this case by fitting a linear regression classifier internally to learned linear hyper plane. This is taken care by Sklearn [34] toolkit using which SVM was implemented in python.

At each camera node, for the feature vector of unknown class (and of length 49), the classification output and its associated probability is obtained. Support Vector Machine classifier available in Sklearn [34] was used. A SVM classifier can be used as a multi-class classifier (using one versus all strategy). We train a single classifier for all the actions at a specific camera/view. So, let SVMj be the SVM classifier trained at view j.

3.4 Score Fusion

The fusion technique presented in this work plays an important role in combining the feature vectors generated from multiple cameras and produce a classification result. In this subsection the concept behind score fusion based classification is explained. Score Fusion strategy is dependent on existent knowledge of the window size. We explain this in more detail by examining the score fusion strategy when window size is known and the other case where the window is not known. In the first case, we assume the window size is known. The system has the knowledge of the window size. It knows exactly where the action starts, ends and its duration. Since all the cameras in the network are time synchronized using NTP protocol, at a given time stamp we can extract data with the given start and end time across all cameras.

3.4.1 Score Fusion with known window sizes

In training phase, the window size for a given *unit action* is known. A subject is performing an action at the center of the Region R at a point Z. We consider the camera C ref as the reference camera which provides a view V_j when an action is being performed by a subject in the Region R at the position Z. It is assumed that the angles between principal axes of pair of cameras ref,s is known. We cannot assume in real-time scenario that camera orientation is symmetric. However, we can confidently say that 1,j belongs to one of the 8 possible view-angle sets. Therefore, we determine other views of other cameras using relative orientations between the cameras. This gives rise to a set of ϕN_v possible configurations.

Let set ϕ denote N_v possible configurations possible for each test action being performed and

$$\phi = \{\{\phi 1\}, \{\phi 2\}, .., \{\phi N_v\}\} \tag{3.1}$$

If we retain the cameras with the symmetric deployment during the test phase, we get Nv possible cyclic configurations. In this case $N_v = 8$ possible configurations where,

$$\phi = \{\{V1, V2, .., V8\}, \{V2, V3, .., V1\}, ..., \{V8, V1, .., V7\}\} \tag{3.2}$$

In the equation 3.2, two views may provide similar views in case the deployment is not similar. In case Cameras C_2 and C_3 are deployed very closely to each other, these two cameras provide same views of the subject performing an action in the Region R at point Z. In this case, the 8 possible configurations will be defined as

$$\phi = \{\{V1, V2, V2, ..V8\}, \{V2, V2, V3.., V8\}, ...\} \tag{3.3}$$

If a camera fails or is completely absent, it will result in $N_c \leq 8$. Therefore, each set in ϕ consists of fewer elements. In order to fuse scores, one has to determine the configuration set. Since we are unaware of the subjects orientation in Region R, we try each of the possible configurations and pick the best one as the most likely configuration. Let feature vector generated at camera Ci be denoted as FV_i. Now we have 2 separate cases (for 4 different classifiers) where we need to classify an action and its associated score/likelihood.

Score generation in case of a LDA classifier: it should be noted that a 2-class LDA was used in this work. The basic idea behind this classification scheme is to compute lower dimensional feature vector and use Nearest Neighbor for classification. For a specific configuration k at a camera i, a product of FV_i and $\lambda_{a,j}$ is performed and this score is normalized to [0,1] (as described in section 3.1.1). Let S_a be the most likely score generated for a given feature vector FV_i.

Score generation in case of MNB, LR and SVM classifiers: MNB, LR and SVM classifiers were implemented as multi-class classifiers (internally using Sklearn [34]). So, there arises no need for designing a classifier per action per view as in case of LDA. In case of a MNB classifier, we determine the maximum a posteriori class/action (as described in section 3.1.2) and its associated likelihood for a feature vector FV_i. In case of a LR classifier, we determine dot product $h_{\theta,j}$ and FV_i is obtained at each camera/view j. This will determine the classification and its associated probability at each view j (as described in section 3.1.3). In case of a SVM classifier, we determine $y = w.x \pm b$ where w is the weight vector learnt by the SVM classifier. This gives rise to classification and its associated probability at each camera (as described in section 3.1.4). Let S_a be the score for most likely classified action for the given feature vector FV_i. Let $S_{a,k}$ be the score generated for the most likely action after applying LDA, MNB, LR and SVM classifiers in each of configurations $\{\phi\{k\}\}$. For a given Action A_a, a matching score $S_{a,k}$ is obtained. The score $S_{a,k}$ represents the likelihood that the test action belongs to the A_a in the given configuration of $\{\phi\{k\}\}$. One can compute $S_{a,k}$ as follows:

$$S_{a,k} = \sum_{i=1}^{N_c=8} S_{a,k,i} \tag{3.4}$$

The above score $S_{a,k}$ corresponds to score in a configuration k. In the real-time scenario, the configuration is unknown. Therefore, we compute score $S_{a,k}$ for each of the configurations k. Then the highest score S_a is picked as the best score of all the configurations. We define S_a as follows:

$$S_a = max(S_{a,k})_{k=1,2,..,8} \tag{3.5}$$

Equation 3.6 determines the classification output and its score (likelihood) obtained after applying view-specific classifiers on given feature vector FV_i. We can determine the final action as $A_F(1 \leq F \leq N_a)$ which has been classified as for the corresponding Feature Vector FV_i as follows:

$$F = argmax(S_a)_{a=1..N_a} \tag{3.6}$$

3.4.2 Score Fusion with unknown window size

In this case, it is assumed that the size of the window is unknown. Window size directly affects the feature vector and thus affects the classification accuracy. Therefore, it is vital that an efficient feature extraction technique should be used in order to generate appropriate feature vectors for classification in real-time mode. In this subsection, the process of generation of the feature vector (a weighted motion energy image) in the streaming mode (real-time) is discussed in the proposed Sliding Window algorithm. In the process of generation of feature vector (in a streaming mode), the duration of action performed nor the starting point of an action is known. In such a scenario, Bobick and Davis [18] have used a heuristic algorithm to capture motion energy images in real-time. Our sliding window algorithm has been based on this idea. Sliding window method has ability to perform online, however we have chosen to implement this offline for the sake of simplicity. Processes of matching the starting and ending points of an action performed in real-time is a hard problem to solve. This problem is overcome in 3 steps which are defined as follows:

1. Obtain feature vectors from all cameras for a specific window size at base station.
2. Obtain classification and probability for a given window sizes using appropriate classification technique.
3. If fused scores are above a certain threshold accept the classification or else reject the classification.

In a realtime scenario, we dont know the exact starting point of an action. So, for simplicity one can traverse the dynamically growing frame sequences using an array or list. This array or list can be accessed to obtain the video sequence as *data[i, i+l]*. Let *FD* be the extracted frame data for a respective value of start variable and let w be the picked window size. Once *FD* is extracted for a specific window size, one can easily compute the feature vector *FV*. Using one of the classification schemes one can compute the classification output and its associated probability. Consider $classifiedoutput_w$ be the classified output for a specific value of w and let $probability_w$ be the probability of the $classifiedoutput_w$. The idea behind sliding window algorithm is to process a list of data frames for all window sizes and accept the classification for which maximum probability is assigned. Let *data* be the array of frames that camera has captured whose length D where, D is proportional to the duration of the camera remained turned on in the scene. The Sliding Window algorithm is depicted in Algorithm 1

```
$D$ := Length of $FD$;
$start$ := 0;
$Windows$ := [List of Window sizes];
while $start \leq D$ do
    for $Windows\ w$ do
        $FD = data[start : start + w]$;
        $FV = getWeightedMotionEnergyImage(FD)$;
        $classifiedoutput_w, probability_w = Classify(FV)$;
    end
    $ClassifiedAction, Probability = max(classifiedoutput_w, probability_w)$;
    if $Probability \geq Threshold$ then
        $Accept$;
        $start = start + w$;
    end
    else
        $Reject$;
        $start = start + N_{min}$;
    end
end
```

Algorithm 1: Heuristic Sliding Window Algorithm

Using algorithm described in Figure 3.6, one can approximately estimate the start and end times in a given stream of video sequence. The value of the *Threshold* and N_{min} can be set arbitrarily. The variable N_{min} helps to recapture the frames again from a different starting point. Then the feature vector is extracted with the estimated start and end time of the action. Then, for each of the window sizes a feature vector is generated and classified according to the technique presented in section 3.4.1. Then we obtain the window size and classified action for which the maximum probability was assigned. The above algorithm is executed at each camera in a camera network where all cameras are synchronized over NTP.

It is also assumed that the buffers (data variable in above algorithm) contain frames of the scene which are approximately associated with the same time stamp across all cameras. The classified action and its associated probability are retained and can be sent to the score fusion server. Equation 3.6 determines the fused maximum likely action A_F and its associated score S_a in all the possible configurations. Let $F_{stream,a}$ be the score generated for the most likely action A in streaming mode (the testing phase where starting and ending of an action is not known). We can use the calculated fused score F to set thresholds on $F_{stream,a}$ to reject or recognize untrained actions that were not trained as a part of the system. In the training phase, we calculate $F_{avg,a}$ which determines the average fused score for action A when the window sizes are known (as in section 3.4.1).

$$F_{thresh,a} \leq F_{avg,a} \tag{3.7}$$

In order to identify untrained actions or reject a classified action in the streaming mode, we determine $F_{thresh,a} = m * F_{stream,a}$, where m can be arbitrarily set. Equation 3.7 attempts to determine if a classified fused score in streaming mode is to be trusted or not. The effect of m i.e., Threshold on average recognition rate has been discussed in Section 4. The score fusion strategy developed in this work can be deployed on a server. To recognize an untrained action or reject a classification made on FV_i, we check the following condition:

Chapter 4

Implementation and performance evaluation

In this section, we discuss the implementation details of this work and systematically evaluate the performance of the system. With the heuristic sliding window algorithm and score fusion scheme, it is shown that the system can handle failure of cameras and still perform with high recognition accuracy. The system was tested in offline mode (where start and end of an action was already known) and in streaming/real-time mode where the data was available as a continuous stream (where starting and ending of the action was not known). Results in both modes are included in this section.

4.1 Implementation details

This chapter discusses the implementation details of the action recognition system described above. We have collected the training data in a controlled environment with 20 samples for each action class with reference to a single camera as shown in the Figure 4.1. Note that the subject remains at the center of network while collecting the training data in order to gain the advantage of symmetry with respect to all the other cameras. The test actions are also then collected in the same setting. Using this data, feature descriptors are then extracted and classification performance is evaluated systematically. This fusion technique can be modified to handle these cases when only partial data from cameras (missing

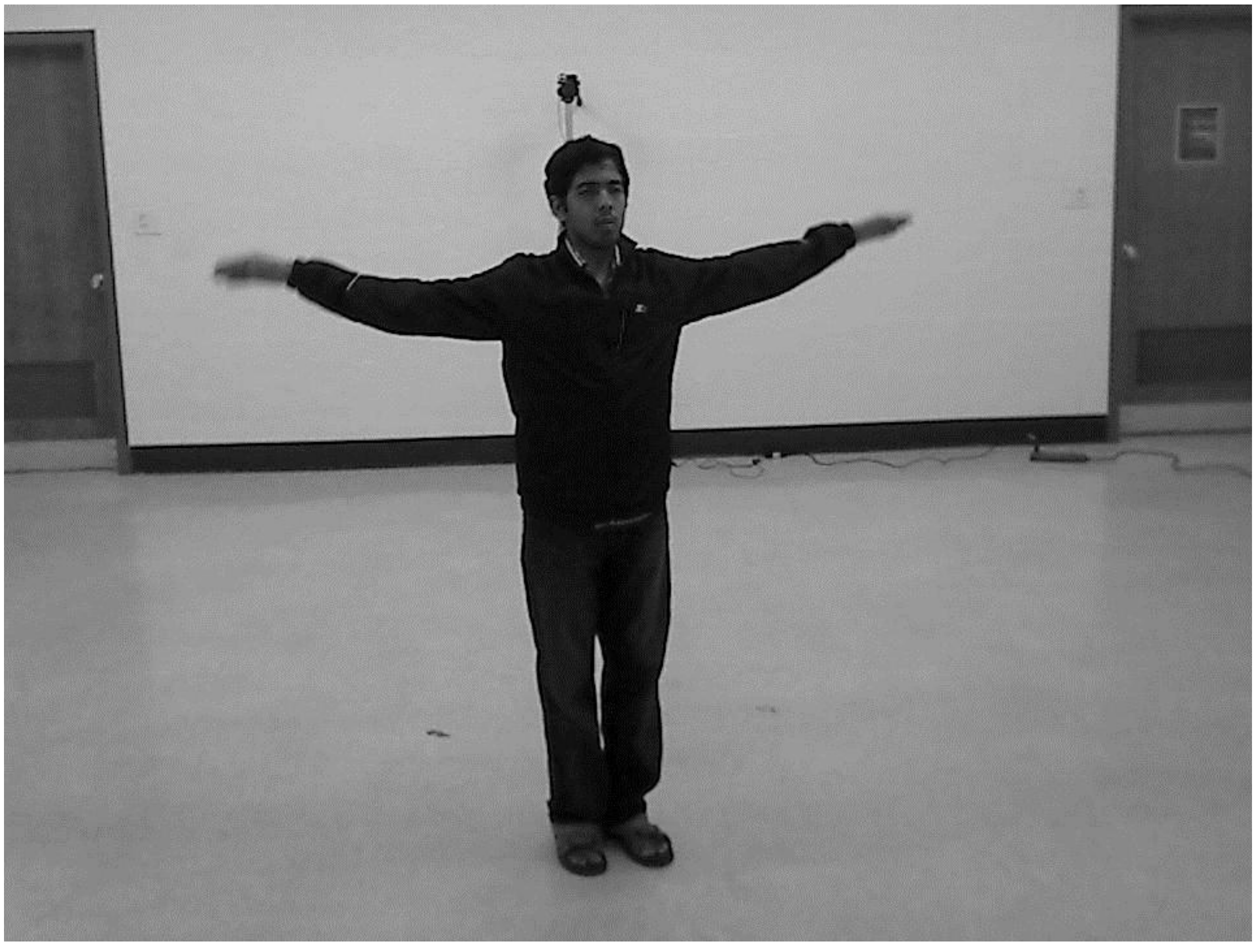

Figure 4.1: Subject performing a 2 hand waving action in the scene as seen from Camera 1

camera data) is available for action recognition. In this implementation, the actions are continuously evaluated as they are being performed.

4.2 Results in Training Phase

We first analyze the performance of recognition system in offline mode where the size of the window is known, (with known starting and ending point of an action is known) with all the views intact (8 views). Then, each of these views is systematically removed and the performance of the system is evaluated. The results are presented in the following tables. In the offline mode, the ground truth is noted down and compared with the classified action to determine the classification accuracy of the system.

It should be noted that in the offline mode, the exact duration of action is known (the starting and ending point of the action are known). In the training phase we provide the system with 20 unit actions per action per view. In the testing phase, the trained classifiers are tested against 20 unit actions. Performance in offline mode gives us an idea of how well the feature vectors and score fusion rule performs for the collected training and testing data.

	A1	A2	A3	A4	A5	A6	A7	A8	A9
A1	19								
A2		19							
A3			19						
A4				19					
A5					19				
A6						19			
A7							19		
A8								19	
A9									19

Table 4.1: Confusion Matrix for LDA classifier with all views intact

	A1	A2	A3	A4	A5	A6	A7	A8	A9
A1	19								
A2		19							
A3			19						
A4				19					
A5					19				
A6						18		1	
A7							19		
A8					2			17	
A9									19

Table 4.2: Confusion Matrix for MNB classifier with all views intact

One may observe in the above tables, see that all (the four classifiers) are good classification techniques. However, Multinomial Nave Bayes was seen to perform least of all these techniques for the given data. Now, we see the impact of loss of camera views (or camera failures) on performance of the system. The above plot was observed by removing random

	A1	A2	A3	A4	A5	A6	A7	A8	A9
A1	19								
A2		19							
A3			19						
A4				19					
A5					19				
A6						19			
A7							19		
A8								19	
A9									19

Table 4.3: Confusion Matrix for LR classifier with all views intact

	A1	A2	A3	A4	A5	A6	A7	A8	A9
A1	19								
A2		19							
A3			19						
A4				19					
A5					19				
A6						19			
A7							19		
A8								19	
A9									19

Table 4.4: Confusion Matrix for SVM classifier with all views intact

views in the order of view1, view2, ..., view8. In Figure 4.2m one can see that the system performs fairly well even with 4 views removed (except for MNB classifier). You can get an average performance of about 80% with 4 views removed. However the effect of removal of views can be seen after removing 5 views, the performance of the system drops steeply.

Now we look at the performance of the system in streaming mode. In the streaming mode, we dont know the exact duration nor the starting and ending point of a unit action being performed nor do we know the starting and ending point of the action. To solve this problem we have described a heuristic sliding window algorithm in section 3.4.2. We use this algorithm to generate the feature vectors in the streaming/real-time mode. Then classification is performed and scores are generated and fused across views. We believe that

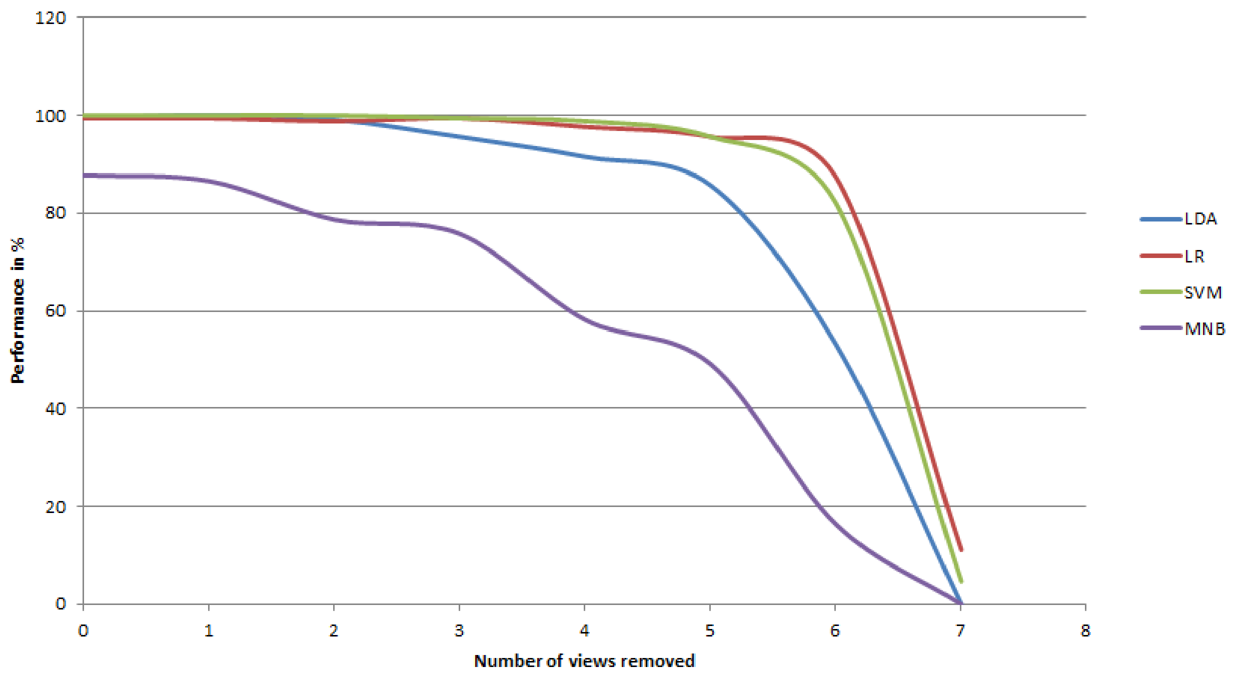

Figure 4.2: performance of the system with known window sizes against number of views removed

MNB is performing poorly compared to other classifiers as Nave Bayes is known to be a good classifier but a bad estimator (where output probabilities are close to 0 or 1).

4.3 Recognition results on streaming data

In this sub-section, we observe the performance of the system in streaming mode. The performance of the system depends on the how the feature vectors are obtained in the streaming mode. The stream of classifications performed by the Sliding Window Algorithm can be considered as series of strings. We are interested in knowing if all the input strings are matched with the respective output strings or not. Hence we use the following technique to calculate the performance:

1. For each input string, compare result with output string.

2. If there is a match, increment correct counter else increment wrong counter.

3. Calculate *correct/total* to get the Accuracy.

Let us assume that the input sequence of unit Actions are represented as in Equation 4.1 $A_1, A_2, X, A_3, A_3, A_4, X, X, A_4, A_4, ...$ Where A_i $(\forall 1 \leq i \leq 9) and X$ represents an Non-Action or an action that is not trained as a part of the system. On applying the Heuristic Sliding Window Algorithm, we obtain a sequence of actions classified by the system as described in Equation 4.2 $A_1, A_2, X, A_9, X, X, A_4, X, X, A_4, A_4, ...$ We obtain a classification for the streaming data by using Heuristic Sliding Window Algorithm described in Algorithm 1 in Section 3.4.2. We need to match the input string represented in Equation 4.1 to output string represented in Equation 4.2. In Algorithm 2, *'=='* represents a *Match* and which is true. A *'!='* represents a *Non-Match* which is true. Since we have performed the analysis offline, we can determine the performance as follows:

```
$A_i$ := input string/unit action;
$A_j$ := output string/action classified;
for Action $A_i$ do
    if ($A_i == A_j$) then
        true positive ++;
    end
    if ($A_i != A_j$) AND ($A_j$ is not a neighboring action of given sequence) then
        false positive ++;
    end
end
for Input $X$ do
    if ($X$ matches $A_j$) AND ($A_j$ is not a neighbor of $X$) then
        false positive ++;
    end
end
```

Algorithm 2: Heuristic Sliding Window Algorithm

We obtain percetage accuracy as $\frac{truepositive}{Total} * 100$. The *false positives* represents the error in the system. First, results with all views intact are presented and the performance of the system with random views removed is shown. We have obtained the following plot in Figure 4.3 after generating the feature vectors and using the classification algorithm on the dynamically generated feature vectors. From the training data it was found that for

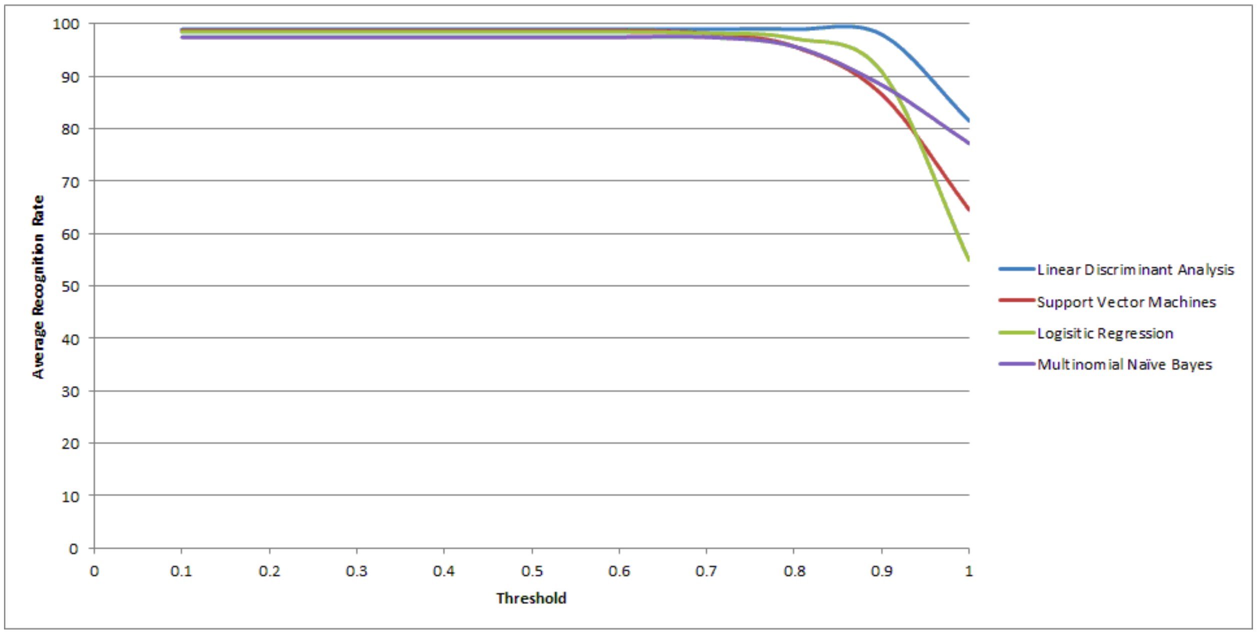

Figure 4.3: Effect of thresholds on performance with all views intact

maximum duration of an action across all cameras was 34 and least was 10. Therefore, we chose W = [10, 13, 16, 19, 22, 25, 28, 31, 34] in the heuristic sliding window technique. The system was tested with zero views removed then with view 1, view 3, view 5, view 7 removed in that order. The threshold used in the Figures 4.3, 4.4, 4.5, 4.6, 4.7 is same as m described in the equation 3.7. In Figure 4.3 one can observe that for most of the thresholds the accuracy remains unchanged from 0.1 to 0.7 and then gradually decrease. So, one can infer that fused score of an action in Streaming mode is quite close to the average threshold of an Action (obtained in Training mode). The performance of the system in streaming mode is fairly good. In many cases, it was found that the sliding window algorithm was not able to completely overlap the exact duration of the action in streaming mode. But, the system was able to recognize these actions quite successfully with high accuracy.

The data used in streaming mode belongs to same data used in offline mode (training data + testing data) across all views among different subjects. However, it should be noted that the feature vectors generated in streaming mode are quite different from the feature vectors used training and testing. The feature vectors in streaming mode depend on starting and ending frame of the unit video sequence. Since, we dont know the starting and ending point

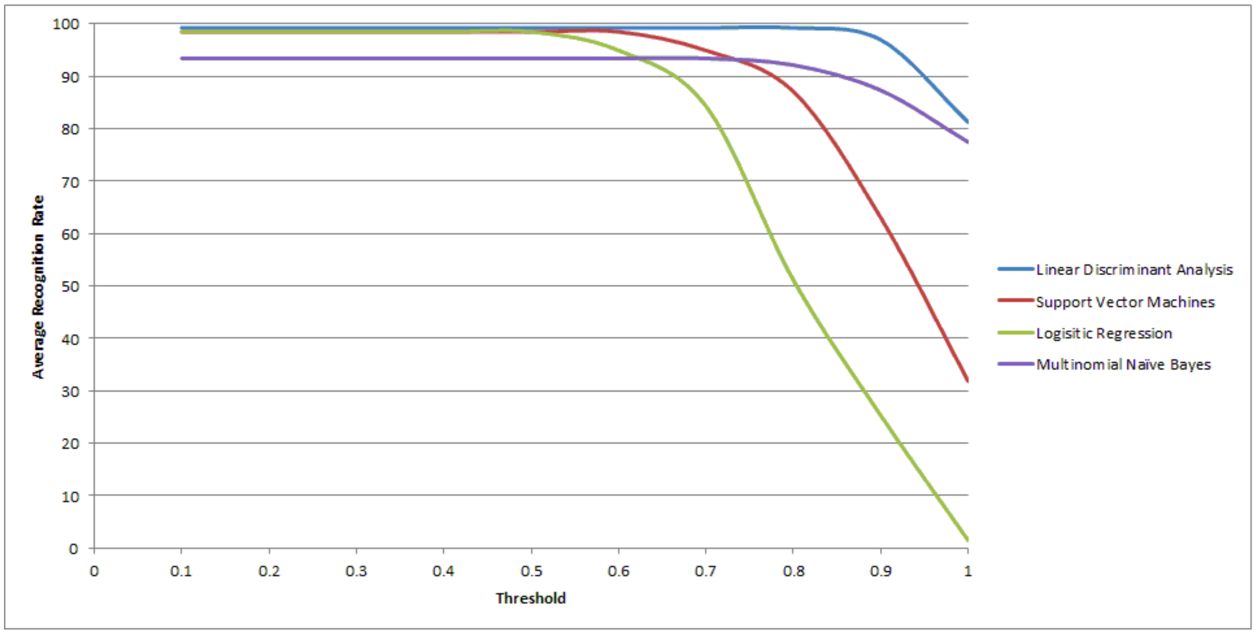

Figure 4.4: Effect of thresholds on performance with 1 view removed

of the unit video sequence in streaming mode the feature vectors were found to be different.

In Figure 4.4 the performance of the system when 4 views are removed is presented. We plot the accuracy of the system versus the varying Threshold. It is observed the performance of certain classification algorithms dropped (Logistic Regression and Support Vector Machines dropped below 50%) when threshold was close to the Average Threshold. Even with 4 views removed, the system has continued to perform well for different thresholds. Similar trend can be observed in Figure 4.5, 4.6, 4.7, 4.8 where for certain threhold values, the performance remains fairly constant then goes down. It is demonstrated that this system has perfomed excellent with upto 4 views removed from the system. The average recognition rate has been in the range of 80-90% for most threshold values.

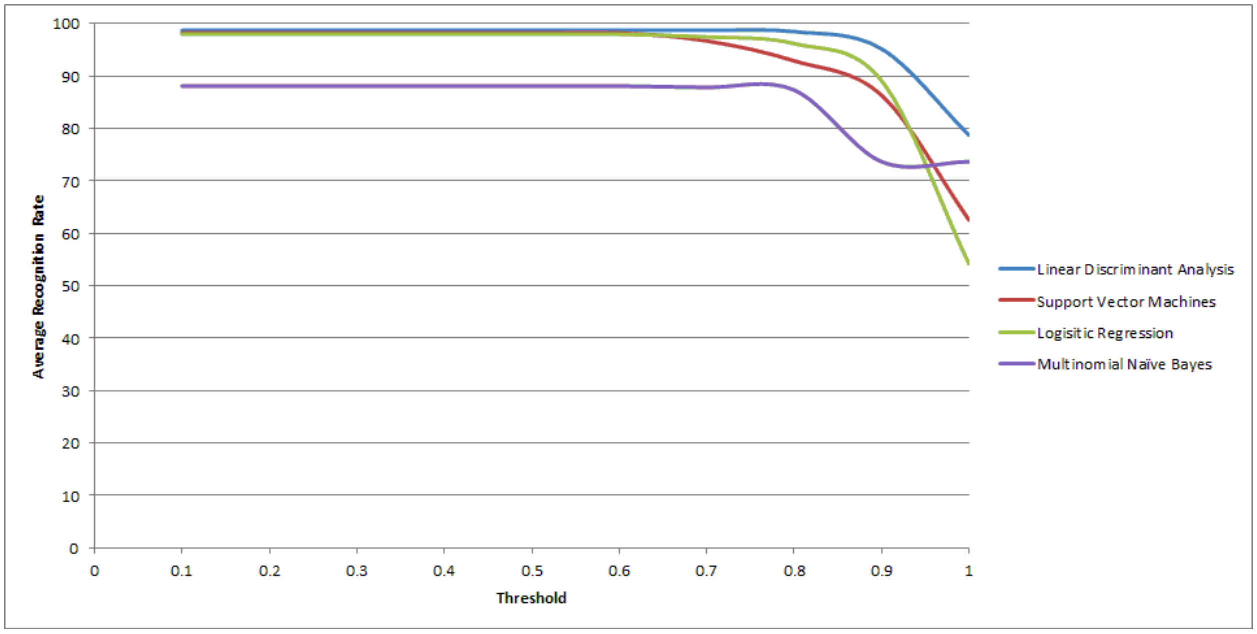

Figure 4.5: Effect of thresholds on performance with 2 views removed

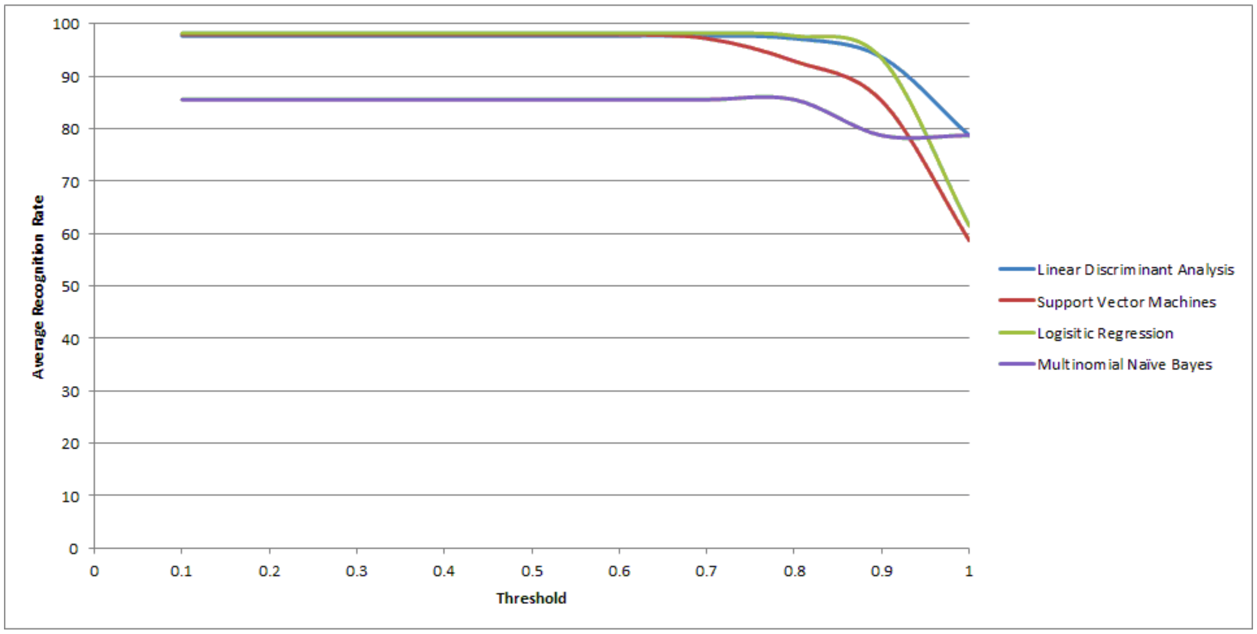

Figure 4.6: Effect of thresholds on performance with 3 views removed

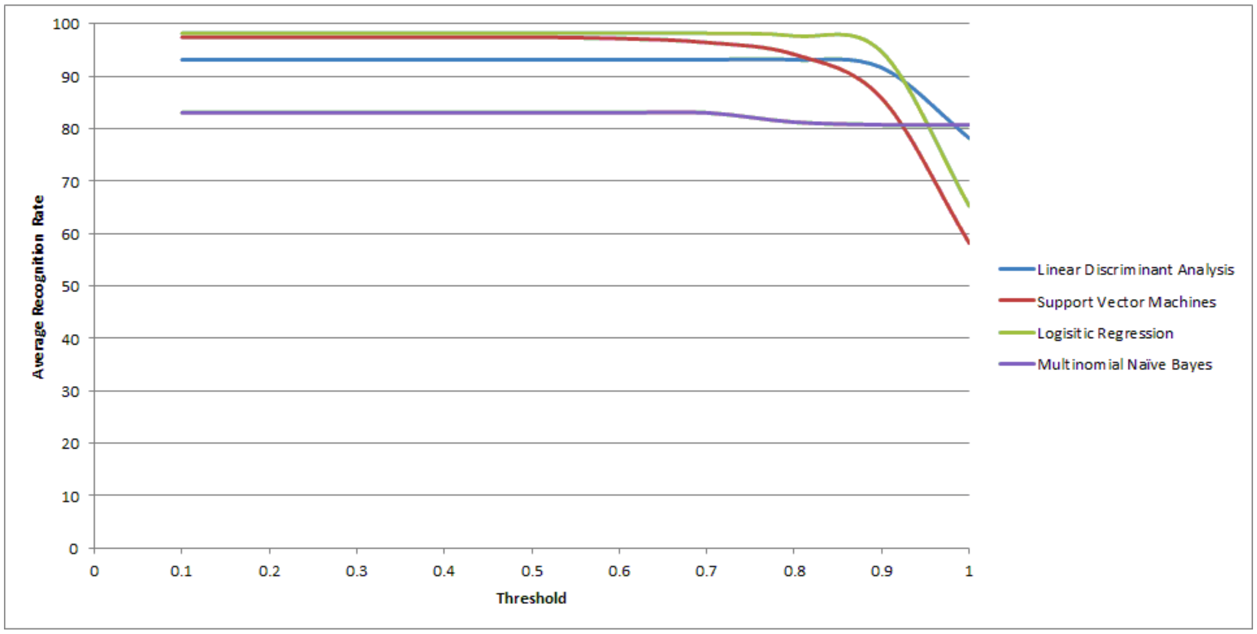

Figure 4.7: Effect of thresholds on performance with 4 views removed

Chapter 5

Conclusion and Future work

This section concludes the thesis by providing conclusions and indicates directions for future work.

5.1 Conclusion

The important contribution made in this thesis is design of highly accurate human action recognition system using multiple cameras. The system was able to recognize (with high accuracy) 9 actions with which it was trained. Standing action was not trained but it was recognized with high accuracy by setting thresholds on fused score. This system was designed using view specific classifiers trained on 8 different views of the scene. Score fusion strategy was found to be more useful rather than relying on voting to fuse multiple decisions at different cameras. Feature vector described in this work was efficient enough to recognize and discriminate actions in the given database. It is expressive enough to capture region of change and magnitude of change in a given human silhouette which is further used to recognize actions. The feature vector used was found to be computationally easy to be computable across different cameras.

The score fusion strategy was proven to work with different probabilistic classifiers and different feature vectors compared to similar work done in [3]. Any probabilistic or likelihood based classifiers can be used with this technique as we are using a score fusion strategy to combine scores across different views. The performance of the heuristic sliding window algorithm indicates that sliding window technique can be used in a camera network with streaming data. This work also shows that symmetric deployment is not needed in training and testing phases when using the score fusion strategy and view specific classifiers. The idea that, no low level view-invariant feature descriptor generation is necessary and view-specific classifiers perform fairly well with streaming data is demonstrated. The system could achieve a performance of over 95% in offline mode and over 85% in streaming mode. The performance in streaming mode was show to be high (an average of 80% accuracy) with 4 views removed from the 8 views. It was shown that it is fairly resilient to failure of cameras.

5.2 Future work

Action recognition is an exciting field right now. Congregations of research ideas from computer vision, distributed computing and machine learning make it very interesting area to pursue research. The current system was proven to be successful in a controlled environment where training and testing were performed. For ease of implementation, the system was implemented in python (due to availability of various machine learning toolkits and frameworks). The system took about 0.15 to 0.3 seconds on an average across different classification algorithms to make a decision at each camera node. This could be improved by providing an implementation in C/C++. In the current prototype, most of the time was spent in io activity and this could be improved by using a limited size buffer and by managing the buffer according to the rate at which data is being captured.

The current system only works when the subject is at the center of the scene, to improve this we need a scale invariant feature vector with high discriminative power. This problem could be further explored. The current system can be extended to identify complex activities, which can be a combination of interleaved actions recognized by the system. In order to solve the problem of involving multiple subjects in the scene, one could introduce powerful yet cheap cameras such as Kinect. However, isolating a subject in an un-controlled environment is a difficult problem to solve [36]. These cameras can be used to detect multiple subjects and current system could be integrated to perform activity recognition.

References

[1] Eunju Kim, Sumi Helal, and D. Cook, "Human activity recognition and pattern discovery," *Pervasive Computing, IEEE*, vol. 9, no. 1, pp. 48 –53, jan.-march 2010.

[2] M.S. Ryoo and J.K. Aggarwal, "Recognition of composite human activities through context-free grammar based representation," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, vol. 2, pp. 1709 – 1718.

[3] S. Ramagiri, R. Kavi, and V. Kulathumani, "Real-time multi-view human action recognition using a wireless camera network," in *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*, aug. 2011, pp. 1 –6.

[4] ThomasG. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic, and Statistical Pattern Recognition*, vol. 2396 of *Lecture Notes in Computer Science*, pp. 15–30. Springer Berlin Heidelberg, 2002.

[5] A. Bobick and J. Davis, "Real-time recognition of activity using temporal templates," in *Applications of Computer Vision, 1996. WACV '96., Proceedings 3rd IEEE Workshop on*, dec 1996, pp. 39 –42.

[6] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[7] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, jun 1992, pp. 379 –385.

[8] P. Turaga, R. Chellappa, V.S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1473 –1488, nov. 2008.

[9] O. Chomat and J.L. Crowley, "Probabilistic recognition of activity using local appearance," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1999, vol. 2, pp. 2 vol. (xxiii+637+663).

[10] L. Zelnik-Manor and M. Irani, "Event-based analysis of video," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, vol. 2, pp. II–123 – II–130 vol.2.

[11] Ivan Laptev, "On space-time interest points," *Int. J. Comput. Vision*, vol. 64, no. 2-3, pp. 107–123, Sept. 2005.

[12] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words," *Int. J. Comput. Vision*, vol. 79, no. 3, pp. 299–318, Sept. 2008.

[13] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, aug. 2004, vol. 3, pp. 32 – 36 Vol.3.

[14] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, oct. 2005, pp. 65 – 72.

[15] Yan Ke, Rahul Sukthankar, and Martial Hebert, "Efficient visual event detection using volumetric features," in *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 - Volume 01*, Washington, DC, USA, 2005, ICCV '05, pp. 166–173, IEEE Computer Society.

[16] E. Shechtman and M. Irani, "Space-time behavior-based correlation-or-how to tell if two underlying motion fields are similar without computing them?," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 11, pp. 2045 –2056, nov. 2007.

[17] M.A.O. Vasilescu, "Human motion signatures: analysis, synthesis, recognition," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 2002, vol. 3, pp. 456 – 460 vol.3.

[18] Aaron F. Bobick and James W. Davis, "The recognition of human movement using temporal templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 3, pp. 257–267, Mar. 2001.

[19] Ming-Kuei Hu, "Visual pattern recognition by moment invariants," *Information Theory, IRE Transactions on*, vol. 8, no. 2, pp. 179 –187, february 1962.

[20] Alper Yilmaz and Mubarak Shah, "Actions sketch: a novel action representation," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, june 2005, vol. 1, pp. 984 – 989 vol. 1.

[21] J. Schlenzig, E. Hunter, and R. Jain, "Recursive identification of gesture inputs using hidden markov models," in *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, dec 1994, pp. 187 –194.

[22] A.D. Wilson and A.F. Bobick, "Learning visual behavior for gesture analysis," in *Computer Vision, 1995. Proceedings., International Symposium on*, nov 1995, pp. 229 –234.

[23] M. Brand, N. Oliver, and A. Pentland, “Coupled hidden markov models for complex action recognition,” in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, jun 1997, pp. 994 –999.

[24] L.R. Rabiner, “” on application of vector quantization and hidden markov models to speaker- independent, isolated word recognition”,” ”Bell Labs Journal.”, 1982.

[25] J.K. Aggarwal and M.S. Ryoo, “Human activity analysis: A review,” *ACM Comput. Surv.*, vol. 43, no. 3, pp. 16:1–16:43, Apr. 2011.

[26] Yan Ke, Rahul Sukthankar, and Martial Hebert, “Spatio-temporal shape and flow correlation for action recognition,” in *In 7th Int. Workshop on Visual Surveillance*, 2007.

[27] H. Aghajan and Chen Wu, “Layered and collaborative gesture analysis in multi-camera networks,” in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, april 2007, vol. 4, pp. IV–1377 –IV–1380.

[28] Oytun Akman, A. Aydin Alatan, and Tolga iloglu, “Multi-camera visual surveillance for motion detection, occlusion handling, tracking and event recognition,” 2008.

[29] P. Natarajan and R. Nevatia, “View and scale invariant action recognition using multiview shape-flow models,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, june 2008, pp. 1 –8.

[30] Vasu Parameswaran and Rama Chellappa, “View invariance for human action recognition,” *Int. J. Comput. Vision*, vol. 66, no. 1, pp. 83–101, Jan. 2006.

[31] L.R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257 –286, feb 1989.

[32] Tao Xiang and Shaogang Gong, “Activity based surveillance video content modelling,” *Pattern Recogn.*, vol. 41, no. 7, pp. 2309–2326, July 2008.

[33] Andrew Mccallum and Kamal Nigam, “A comparison of event models for naive bayes text classification,” 1998.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine Learning in Python ,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[35] Corinna Cortes and Vladimir Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sept. 1995.

[36] Shuiwang Ji” ”Ming Yang, “”detecting human actions in surveillance videos”,” ”TREC Video Retreival evaluation workshop”, 2009.