ONLINE AND SEMI-AUTOMATIC ANNOTATION OF FACES IN PERSONAL VIDEOS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET CELALEDDİN YILMAZTÜRK

IN PARTIAL FULFILLMENTS OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

MAY 2010

Approval of the thesis:

ONLINE AND SEMI-AUTOMATIC ANNOTATION OF FACES IN PERSONAL VIDEOS

submitted by **MEHMET CELALEDDİN YILMAZTÜRK** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering, Middle East Technical University** by:

Prof. Dr. Canan Özgen _____
Dean, Graduate Schoolof **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Assistant Professor İlkay Ulusoy _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu _____
Electrical and Electronics Engineering Dept., METU

Asst. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Aydın Alatan _____
Electrical and Electronics Engineering Dept., METU

Asst. Prof. Dr. Şenan Ece Güran Schmidt _____
Electrical and Electronics Engineering Dept., METU

Instructor Dr. Ayşenur Birtürk _____
Computer Engineering Dept., METU

Date: **05.05.2010**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Mehmet Celaleddin Yılmaztürk

Signature :

ABSTRACT

ONLINE AND SEMI-AUTOMATIC ANNOTATION OF FACES IN PERSONAL VIDEOS

Yılmaztürk, Mehmet Celaleddin

M.Sc., Department of Electrical and Electronics Engineering

Supervisor : Asst. Prof. Dr. İlkay Ulusoy

May 2010, 60 Pages

Video annotation has become an important issue due to the rapidly increasing amount of video available. For efficient video content searches, annotation has to be done beforehand, which is a time-consuming process if done manually. Automatic annotation of faces for person identification is a major challenge in the context of content-based video retrieval. This thesis work focuses on the development of a semi-automatic face annotation system which benefits from online learning methods. The system creates a face database by using face detection and tracking algorithms to collect samples of the encountered faces in the video and by receiving labels from the user. Using this database a learner model is trained. While the training session continues, the system starts offering labels for the newly encountered faces and lets the user acknowledge or correct the suggested labels hence a learner is updated online throughout the video. The user is free to train the learner until satisfactory results are obtained. In order to create a face database, a shot boundary algorithm is implemented to partition the video into semantically meaningful segments and the user browses through the video from one shot

boundary to the next. A face detector followed by a face tracker is implemented to collect face samples within two shot boundary frames. For online learning, feature extraction and classification methods which are computationally efficient are investigated and evaluated. Sequential variants of some robust batch classification algorithms are implemented. Combinations of feature extraction and classification methods have been tested and compared according to their face recognition accuracy and computational performances.

Keywords: Online Learning, Facial Feature Extraction, Face Recognition.

ÖZ

KİŞİSEL VİDEOLARDAKİ YÜZLERİN ÇEVRİMİÇİ VE YARI OTOMATİK İSİMLENDİRİLMESİ

Yılmaztürk, Mehmet Celaleddin

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü

Tez Yöneticisi : Yard. Doç. Dr. İlkay Ulusoy

Mayıs 2010, 60 Sayfa

Görsel malzemenin miktar ve erişilebilirliğinin hızla artması sonucu, video isimlendirme uygulamalarının önemi de arttı. Görsel içerik aramayı verimli bir şekilde gerçekleştirebilmek için, etiketleme işinin önceden yapılması gerekir ve bu oldukça zaman alıcı bir uğraştır. İnsan yüzlerinin, videodaki kişileri tanımak için otomatik olarak etiketlendirilmesi, içerik-tabanlı videodan bilgi çıkarma yöntemleri için büyük bir zorlayıcı etkendir. Bu tez, etkin öğrenme yöntemlerinden yararlanılarak, yarı otomatik yüz etiketlendiren bir sistem gelişitirilmesine odaklanmıştır. Sistem yüz tespit ve takip yöntemleri kullanarak videodaki yüzlerden oluşan bir veritabanı oluşturur ve kullanıcıdan aldığı isimlerle etiketleme yapar. Bu veritabanı kullanılarak bir öğrenme modeli eğitilir. Eğitim süreci boyunca da, sistem yeni karşılaşılan yüzler için kullanıcıya isim önerileri yapar, kullanıcı ise bu isimleri onaylar ya da doğru isimleri girerek düzeltir, böylece sistem sürekli olarak eğitim bilgileriyle güncellenir. Kullanıcı sistemden tatmin edici sonuçlar alana kadar eğitimi sürdürebilir. Yüz veritabanını oluşturmak için, videoyu anlamsal bağlamda bütünlük içeren parçalara bölmek adına , sahne sınırlarını tespit eden bir işlemsel süreç uygulanır ve kullanıcı videodaki sahne sınırlarını belirleyen

film kareleri arasında gezer. Yüz tespit ve takip yöntemleriye sahne sınırında bulunan yüzleri bir sonraki sahne sınırına kadar takip ederek veri toplayan bir işlemsel süreç uygulanmıştır. Etkin öğrenme için, işlem yükü açısından verimli öznitelik çıkarma ve sınıflandırma yöntemleri incelenmiş ve değerlendirilmiştir. Güvenilir sonuçlar veren ve verileri toplu işleyen bazı sınıflandırma yöntemlerinin ardışık veri işleyen türevleri uygulanmıştır. Öznitelik çıkarma ve sınıflandırma yöntemlerinin bileşimleri denenmiş ve yüzleri tanımadaki başarılarıyla işlem yükleri göz önünde bulundurularak kıyaslanmıştır.

Anahtar Sözcükler: Çevrimiçi Öğrenme, Yüz Özniteliklerinin Çıkarımı , Yüz Tanıma.

To my family

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor Asst. Prof. Dr. İlkay Ulusoy and my project supervisor Assoc. Prof. Dr. Nihan Kesim Çiçekli for their guidance, assistance and encouragement.

I also would like to thank my family members for their support, encouragement and love throughout my life.

I also would like to express my special thanks to my colleagues and friends in the Computer Vision and Intelligent Systems Laboratory, Metin Burak Altınoklu, Örsan Aytekin, Tülay Akbey, Erdem Akagündüz, Neslihan Özmen, Ömer Eskizara, Yasemin Özkan Aydın, Okan Akalın, Gökhan Yaprakkaya and Rasim Aşkın Dilan for their support and guidance.

I would like to express my deepest gratitude to Ege Saygıner for her support and encouragement.

This work is partially supported by The Scientific and Technical Council of Turkey Grant ‘‘TUBITAK EEEAG-107E234.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Due to the development of high-speed internet as well as the increase in the number and variety of high-quality digital platforms, access to huge amounts of multimedia content has become as easy as has never been. People can acquire video clips, movies and all sorts of audio/video content with a click of the mouse, as they share the content they have produced with the community in return. This led to the accumulation of multimedia content in all sorts of different media platforms including cyberspace, personal archives, DVD's etc. But huge amounts of data demands an effective search method to access the desired content. Semantic annotation is one solution to group and label content efficiently. In this work the focus is on the visual cues extracted from the multimedia data. If visual content can be annotated semantically within the context of an ontological structure, i.e., if keywords and textual descriptions are entered by the user as the metadata, queries could be much more specific and target oriented. Instead of receiving vague results, precise information can be obtained through data retrieval techniques.

One major branch of semantic annotation involves the labeling of people in videos with metadata. This is a broad subject covering annotation of people including faces, upper bodies, pedestrians, etc. Each type of target definition brings its own challenges and application requirements within its context. Although fruitful, annotation is a time-consuming and exhaustive process if manually performed. For example, in order to annotate occurrences of an actor/actress in a TV-series, dozens of episodes have to be hand-labeled one by one, where each of the episodes has tens or even hundreds of scenes. And, for example, if the location of the actor's face is to be recorded, then most probably

the effort would be limited to recording one sample location per scene, unless the annotator is willing to hand label each frame of the scene, one by one. It is understandable that users are resistant to the task of data entry [31]. An alternative approach is content-based indexing and retrieval which provides some degree of automation for this process by automatically extracting features directly from the data. For personal photo and video archives, no additional information such as text or voice is assumed to be available. Thus, the only information comes from the visual data. By using the intrinsic visual attributes of images, such as color, structure, texture, and composition, users can search collections by instructing the system to retrieve images that are visually similar to the sample image. The disadvantage of this approach is that these systems only extract low-level syntactic features, which are not semantically useful to consumers as keyword-based annotations would be. In this study emphasis was put on semi-automatic annotation methods where some annotation is requested from the user for the purposes of training the system. The image and video contents are analyzed automatically and the user provides the annotation for the analyzed data only. The learning effort from this limited information and the annotation of the rest of the video are performed automatically.

For personal multimedia archives, the most interesting contents are human faces in the photos and videos. Thus, in general, face annotation can be regarded as an extended face detection and recognition problem if one considers only the visual information. Fortunately face identification is one of the most promising sub-branches of automatic annotation of people since faces are, unlike others, unique in the sense that recognition can be performed using discriminative features extracted from faces. A person on the video may change his/her clothing, hairstyle etc. but faces have to remain more or less the same thus faces can be thought of as samples from a higher dimensional face-manifold which covers a continuum of face image representations along size, expression, illumination, resolution and pose axes. These parameters can yield quite distorted face representations which make face recognition yet quite challenging.

1.2 Background

There is a large amount of work for face recognition in photos [32]. Also, face annotation for personal photo archives has been studied extensively [13, 14, 15, 16, 35, 36, 37]. However, the recognition of faces in videos is more challenging because of the

much more dynamic nature of the videos involving bizarre conditions which distort the faces. The face annotation in video requires not only precise face detection on frames but also precise face tracking through frames. The state of the art methods for face detection, tracking and recognition in video [16, 19] usually include a single face moving in front of the camera.

Thus, semi-automatic annotation methods based on faces are also possible. First of all, methods use the state of the art face detectors to detect frontal or close to frontal faces in videos, especially at shot boundaries. Then, face trackers are employed to attach images of the same face and to extract the sequence of face within a shot although the face has various poses and expressions throughout this sequence. Some of the tracks are labeled manually and used as the training set. Finally, the rest of the tracks are labeled automatically based on the manually labeled set [9].

The proposed methods are tested on the videos of movies, news and TV series which include many characters and scenes. In this way, the proposed methods are used for automatic naming of characters [8, 29, 38, 39, 43, 44]. As the related work, we consider methods that provide annotations based on only the facial region and exclude methods considering hair, body and clothing [38, 39, 43] because these may show many more variations than the appearance of the face. In [33], a neural network based face detector and skin color based face tracker are used. Each detected face is normalized to 64 x 64 pixels and the face sequence matching is done by the appearance based face matching. Automatic labeling is done by finding the pair of query and gallery samples which are closest to each other where the entire query track is labeled with the class label of the gallery sample. Four methods (Eigenface, Fisher's Linear Discriminant, subspace and kernel methods) are tested for the performance of face recognition and Fisher's Linear Discriminant is found to perform the best [23]. In [38], Viola-Jones face detector [1] and Kanade-Lucas-Tomasi face tracker [20] are used. Hundreds of faces are labeled for learning facial features using Adaboost where Haar-like features are used [1]. The facial appearance is represented by the descriptions obtained around the facial features. The pixel values or SIFT features [39] are extracted as the descriptions. The automatic labeling is done in the same way as [18]. In [39] Viola-Jones face detector [1] and color based face tracker are used. The tracks are clustered and the automatic labeling is performed in the same way as [18]. Viola-Jones face detector [1] and particle filter face

tracker are used in [9]. Discrete Cosine Transform (DCT) coefficients are extracted from the faces [3] and nearest neighbor classifier is used for automatic labeling.

Each study in the literature performs different tests on different data sets and presents its results in a different way. The common method of testing is that some part (nearly one quarter) of the video is manually annotated (i.e. the tracks are labeled) and the rest of the video is automatically labeled and the performance of the method is presented based on correct labeling [8, 9, 38, 39, 43, 44]. This method is also suitable for personal video annotation. Some studies working on TV series episodes use an episode for training and use other episode or episodes for testing [18,39].

Various performance measurements are used in different studies. For example, in [38, 43, 44] precision-recall curves are used where recall is defined as the proportion of tracks which are assigned a name after applying the "refusal to predict" mechanism and the precision is defined as the proportion of correctly labeled tracks out of all labeled tracks. If all of the test tracks are labeled (i.e. recall is %100) then the precision is between %63 and %69 for various episodes of "Buffy the Vampire Slayer". Similarly, in [39], precision less than %60 is achieved for %100 recall for various episodes of "Friends". In [18, 34] the accuracy of labeling is plotted against the number of training sequences. The highest accuracy reached for various episodes of "Oshigoto-desu!" is %60 in [18]. In [9] TV series "Coupling" is used for testing and precisions less than %40 is achieved for each character at %100 recall.

The systems proposed in the literature are all off-line [5, 8, 9, 18, 34, 38, 39, 43, 44], i.e. the annotation is performed off-line and then the results are provided to the user at a speed close to real time. Usually, many post processing steps are applied for accurate tracking and face clustering. Besides, batch learning of labeled data is performed first and then the rest of the data is labeled automatically. Nearly none of the studies presents a performance for the computational time of the training and testing steps, except in [18] where four methods are compared both in terms of accuracy and CPU time of the training sessions.

In this study, a semi- automatic video annotation tool based on faces is developed for personal videos where the annotation is done online with very minor user interaction. This application scenario is different from the previous studies. The goal is to develop an

automatic face annotation tool with an online learning stage. The system interacts with the user to get annotations for the unknown faces in the beginning of the video. These initial annotations are used for the training of the system. After a period of training time, the system automatically recognizes the learned faces in the remaining part of the video.

The system also allows the user to correct any of the automatic annotations, if necessary, and these corrections are added to the training set to improve the success of the system. Instead of batch classification or learning, which are used in the literature for off-line applications, sequential methods are used in this study, since learning continues throughout the annotation process in the form of either user labeling or user feedback. In order to build a real-time learning system, the parts of the system have to be fast and robust. The timing constraint limits the use of some algorithms such as classification based on facial features using gabor wavelets [28] since they are found to be computationally expensive according to the preliminary tests. Thus, each stage of the system (shot detection, face detection at shot boundaries, face tracking throughout the shot, learning of the manually labeled face tracks, labeling of faces on the other tracks) is optimized in terms of both accuracy and time.

The work is organized as follows: Chapter 2 presents an overview of the proposed system and the main steps of the method are discussed. Chapter 3 involves the description of the algorithms used for face recognition and comparative performance evaluation for each method is presented in terms of execution times and recognition accuracies. In Chapter 4, the online automatic annotation tool is tested for video annotation using an episode of a TV series. Extensive tests are made to compare the performances of the considered algorithms for online annotation application. Chapter 5 presents the concluding remarks and possible future work.

CHAPTER 2

PRELIMINARY METHODS

2.1 Overview

The main steps of the proposed system are shown in Figure 2.1. The shot boundaries in the video are automatically detected and presented to the user one at a time. For a real-time application, the shot boundary detector should have a reasonable performance. In this study, after investigating various methods [27], the digitized color histogram difference method has been implemented. Adoption of complex features like edges for shot boundary detection has been shown to be inadequate to outperform the performance of the simpler algorithms, yet complex features require much more computational power [27]. Histogram-based methods are used frequently in many similar video face identification systems [8, 17, 43].

Figure 2.1. The Flowchart of the Overall System

The faces on the shot boundary frames are detected automatically by a state of the art face detector [1] which is proved to perform well in previous applications [9, 38, 39]. In order to improve the performance of the system, the face detector is applied to edge enhanced images. Only frontal faces or faces very close to frontal pose are detected. We do not use additional cues such as hair or clothing [38, 39] because hair or clothing may change a lot in a personal archive.

The user is asked to label the detected faces at the shot boundaries through a user interface. Then the faces are tracked through the frames within the shot. In our method, human faces are tracked in order to populate a database of target faces which we want to recognize. So it is essential to have high quality face tracks. A good track must contain only the face and the least amount of background clutter. Also the facial features (eyes, mouth, etc.) which are crucial for the consequent recognition step must be contained in the spatial boundary of the track, if they are already visible in the scene. Methods like Continuously Adaptive Mean Shift [6] and Particle Filtering [25] may be used for face tracking; however there are some problems in using them for our purposes. They are not suitable for real-time applications. They may result with tracks including noisy background beside faces. Also, they may depend on manually set parameters. Therefore, in this study a well-known face tracker, which is based on generic features such as corners [20, 21], is used with some modifications. In this way, faces are tracked faster and better.

The tracks of the detected faces are saved to the database with the labels input by the user. After some time, which may also be defined by the user, the system learns the faces in the database. Then, other faces are automatically detected, tracked and recognized in the rest of the video. The recognized faces are presented to the user to receive feedback about the correctness of automatic labeling. Based on the user feedback, the database is revised. For example, if the recognized face is wrongly labeled by the system, the user will correct the label and the corresponding face tracking information will be added to the correct label class. The classification is performed once more with this new information.

The face recognition methods which are highly popular for annotation applications are classification methods such as Nearest Neighbor (NN) [3] and Linear Discriminant Analysis (LDA) [23] and learning methods such as Support Vector Machine (SVM) [26].

Although there is a consensus in the literature about the appropriate face detector and tracker for automatic annotation applications, a commonly agreed method doesn't exist for face recognition. There are many recent studies which compare various face recognition methods used for annotation applications. For example, NN and SVM are compared in [43], the Eigenfaces [22], Fisher's linear discriminant, subspace and kernel function subspace methods are compared in [18] and Single Kernel SVM and Multiple Kernel SVM are compared in [44] by using face sequences obtained from videos. In this paper, NN, LDA, SVM and MK SVM are compared for our real-time annotation application. However, the sequential approaches are used, since the classification is done throughout the annotation process when user labeling or user feedback is received. The sequential learning is faster and more suitable for such a real-time application. Because, instead of re-training or re-classifying the whole database with the new information, training is done incrementally so that the previously learned model is updated with the new information. Thus, except for NN, sequential versions of LDA and SVM are implemented and compared for our application. The faces are not directly used in these methods, instead some informative features such as Discrete Cosine Transform (DCT) [3], Local Binary Patterns (LBP) [2] and Histogram of Gradients (HOG) [42] are extracted from face images and these are used with all classification and recognition methods and the performances are compared.

2.2 Shot Boundary Detection

A shot is a visually continuous collection of frames starting from a scene change in the video to the next scene change, or a switch of the cameras. A variety of shot boundary detection algorithms exist in the literature [10, 27]. The basic idea of shot boundary detection is to compare consecutive frames to detect a significant change in terms of a given metric. These include methods which compare consecutive frames by calculating direct pixel differences, digitized color histogram differences and others such as calculation of edge change ratios within frames.

In our work we have chosen the digitized color histogram difference method because it is fast and performs well when compared to the other methods [27]. Additionally, most of the personal videos and TV shows include direct cuts as shot boundaries and implemented algorithm is found adequate for detection of the shot boundaries although some false shot boundaries may be declared. But, the accuracy of the shot boundary

detection algorithm is not essential for the performance of the overall system since the main point is that shot boundaries are used as beacon points for detection and tracking of faces. Extra shot boundaries would only mean an additional face detection-tracking step. And a missing shot boundary will be noticed by the facial feature tracker. The face tracker would stop at the shot boundary frame due to the abrupt change in the pixel values between consecutive frames.

Each color channel (R,G,B) is digitized to 8 values and the color histograms of the frames are extracted. The digitization step is a precaution to reduce susceptibility to noise. Next, the histograms of consecutive frames are compared via the histogram difference method as shown in Equation 2.1.

$$\boldsymbol{d(h,g)} = \sum_{\boldsymbol{R}} \sum_{\boldsymbol{G}} \sum_{\mathrm{B}} (\boldsymbol{h(r,g,b)} - \boldsymbol{g(r,g,b)}) \qquad (2.1)$$

Here *h* and *g* are the RGB color histograms of two consecutive frames. If the difference value ***d(h,g)*** exceeds a threshold then a shot boundary is declared. In order to cope with transient shot changes which may last for more than one frame, only a local maximum within a window is considered and the other results are discarded. The default value for this window is chosen to be five frames.

2.3 Face Detection

There are a number of robust face detection algorithms in the literature for near frontal faces. The most commonly used one is the Viola-Jones face detector [1] which is a cascade of Haar-like features boosted by adaboost tool. Viola-Jones face detector is run at each shot boundary frame. If the shot boundary frame does not return a detected face, the next frame is loaded and another face detection trial is made. This procedure is repeated until a face is detected. If a face cannot be detected within a certain number of frames (chosen as 10) following the shot boundary frame, the current shot boundary is

presented to the user as-is, namely without any marked face locations but the user may mark the missed faces manually or continue browsing through shot boundary frames by skipping that frame and the search is restarted from the next shot boundary frame.

2.4 Face Tracking

In our system, human faces are tracked in order to populate a database of target faces which we want to classify. So the quality of the face tracks is essential for the performance of face recognition. A good track must contain the minimum possible amount of background clutter. Also the facial features (eyes, mouth, etc.) which are crucial for the consequent recognition step must be contained in the spatial boundary of the track, if they are already visible in the scene. The speed of tracking is another important issue for a real-time annotation system. Therefore, we investigated possible trackers in detail in order to choose the right method. The most commonly used methods, such as Continuously Adaptive Mean Shift algorithm [6], Particle Filtering [25, 30] and Optical Flow Feature Tracking [20], are compared in terms of performance and speed.

2.4.1 Continuously Adaptive Mean Shift (Camshift)

Camshift [6] is a very fast algorithm however it requires its parameters to be tuned depending on the scene characteristics. This tracker makes use of the color histograms of the scene and the rectangular face region. A 2D (Hue x Saturation) model histogram is formed using the initial face rectangle. Camshift algorithm assigns probability scores to the pixels indicating their likelihood of belonging to a facial color. For each pixel in the search region, the number of occurrences of the given color is counted in the model histogram (which is extracted from the initial face rectangle) and this score is assigned to the pixels with the given color on a map which has the size of the search region. Then Mean Shift algorithm [7] is applied to determine the mode of the biggest blob in an iterative manner. The new size of the rectangular region is also determined from this map by summing up the total score under the designated center of the face.

In order to differentiate the color distribution of the face region from the background region, some color values need to be trimmed and filtered. For example, hue values, which are very effective in differentiating human skin color, are quite unreliable given low saturation or low brightness values. The threshold values for these filters have to be determined for each scene.

Instead of manually determining the threshold values, an automatic approach may be used to determine the appropriate threshold values. For this purpose, histograms of hue, saturation and value (HSV) channels are extracted for the background scene excluding the face region and a separate set of histograms are similarly extracted for the detected face region. The background scene is chosen as the surrounding rectangle with an area four times larger and centering the face rectangle. Our aim was to decide the thresholds for HSV channels which would enable us to separate the face region colors from the background colors as much as possible. 1D Gaussian distributions are fit to the face color histograms independently for each channel. The threshold value for each channel is set where the Gaussian model for the face rectangle intersects the actual distribution for the background histogram. If this intersection occurs for a number of values, the one which is closest to the mean value of the Gaussian (and smaller than the mean value) is selected. But this method is found out to be susceptible to small deviations and the threshold could not be determined accurately enough.

Another problem with the Camshift algorithm is that only color information is used throughout the whole tracking process. This may cause the tracker to engage to the neck (Figure 2.2), balding heads, or hands moving near or in front of the face in addition to the already problematic human skin-color resembling background.

Figure 2.2. A problem with CamShift. The face rectangle grows rapidly due to skin colored pixels in the neck region.

Thus, the method is found to be inadequate for our application because a small change in the parameters may cause a drastic change in the behavior of the tracker and the tracker ends up capturing regions other than the faces easily.

Figure 2.3. Output of the particle filter face tracker. Faces are not centered. There is a significant amount of background clutter.

2.4.2 Color Based Particle Filter Face Tracker

We have used a particle-filter color tracker implementation available in [30]. This method is not fast enough to be used in a real-time application. Also the success of this tracker is found to be inadequate and the tracker cannot compensate for its high computational demands. Although independent from parameter tuning, the face boundaries within the tracks often contain significant amounts of background clutter. The tracker is susceptible to noise, so intruder objects passing in front of the face cannot easily distract the tracker but the steady nature of the tracker is also a problem when the face itself is moving throughout the scene rapidly where the tracker does not change its state as fast as the face itself. A sample output track can be observed in Figure 2.3.

Some post processing was applied to improve the performance of the tracker. For example, a human-skin color recognizer based on a Support Vector Machine classifier was applied to each tracker output so that the background clutters could be trimmed from the central face region. But this led to a significant amount of decrease in the performance (in terms of speed) since each face image in the track needed to be processed by the skin color recognizer. Another reason for this approach not being satisfactory was that no significant trimming could be achieved if the background clutter resembled the human skin color.

2.4.3 Lucas-Kanade Pyramidal Optical Flow Feature Tracker

The third face tracker algorithm that we have implemented is the Lucas-Kanade optical flow tracker [21]. In order to use the optical flow algorithm, certain features need to be selected. These features have to be selected appropriately so as to increase the success of tracking. For this purpose, we select a number of feature points around the target facial feature areas according to [20]. The rectangular search regions around the eyes and mouth are selected according to the size of the face which is obtained from the face detector. In each of these three regions, a number of (chosen as 4) feature points are determined by detecting the corners. Having selected a total number of 12 corners, with 4 corners for each region, the optical flow tracker algorithm finds the best matches for these 12 points in the following frame of the video. The mean value for each feature quadruple is claimed to be the location for the corresponding feature. The minimum bounding rectangle of the three feature centers is enlarged by a scale factor, which is chosen heuristically after inspecting several face images, and this rectangle is saved as the face rectangle. When there is an amount of optical flow error bigger than the designated threshold or if one or more of the feature points are lost, the tracking is completed.

In optical flow tracking, the neck region, balding heads and background clutter resembling human skin color cannot affect the tracking quality since the color information is irrelevant. In addition, obstacles which may intervene by occluding the face, like moving hands, stop the tracking process. Otherwise such obstacles would have caused the accumulation of corrupt images in the database. Samples from the optical flow tracker can be observed in Figure 2.4. These results are unprocessed and direct outputs of the face tracker. It is clear that the optical flow tracker is robust under a wide variety of conditions such as low illumination (3^{rd} and 4^{th} rows), changes in facial expression (1^{st} row and 6^{th} row) and low resolution (bottom row).

As a conclusion, the optical-flow facial feature tracker performs best for our application yielding the best face localization where the background clutter causes important degradation in the recognition performance. Also optical-flow feature tracker is not affected by the human-skin resembling colored backgrounds whereas color based trackers suffer a lot. There is a trade-off between the quality of the tracks and the length of the tracks, where selection of parameters for the termination criteria of the tracking

determines whether the tracking is to be terminated when the face goes through a big change. Long tracks may contain much more references of the target person and also the recognition could be started much earlier if the samples are collected in big steps, on the other hand if the termination criteria for the tracks are loose, the track may contain lots of different poses which may include poses that are not suitable to construct a model for the target face, such as profile faces or even some feature points may engage false corners outside the face region, resulting in taking samples which are erroneous. These may critically endanger the success of the recognition.

Figure 2.4. Example face tracks from the TV series "How I Met Your Mother"

CHAPTER 3

FACE RECOGNITION

3.1 Overview

In this chapter, the methods of face recognition which are used in this thesis will be evaluated. Each facial feature extraction algorithm and classification method is explained in detail. Results of extensive tests which investigate recognition accuracies and execution times are presented for offline testing sessions in order to compare the methods at the base level. We have implemented NN, LDA and SVM (with single and multiple kernels) as face recognition methods with features such as DCT, LBP and HOG.

In order to compare execution times and recognition precisions, all feature extraction methods are tested with all classification methods. The execution times of all methods are plotted for both training and testing phases as the number of training and testing samples changes. In order to calculate the precision of face recognition algorithms, several tests have been made to compare the accuracies of the methods under different conditions.

Accuracies are calculated as the number of target classes is changed. Similarly the variation of accuracy is plotted as the amount of training data is altered. All tests are made for two distinct face datasets. The first dataset is a collection of hand-labeled face detection outputs. Viola-Jones face detector [1] has been run for every single frame throughout an episode of "How I Met Your Mother" TV series and the detected faces of target people are manually clustered. The second dataset is created by using a face tracker algorithm on the same episode. OpenCV implementation of Camshift Color Based face tracker [6] is used in order to track faces throughout the video. The resulting

face tracks are also manually labeled and clustered. Samples from both datasets are fed to an illumination compensation algorithm before recognition is performed.

The face detections database is composed of more-controlled and better-posed face images since the face detector detects frontal upright faces. The face tracks database on the other hand, captures a much more diverse set of face images as the tracker can collect rotated or otherwise distorted faces which may be missed by the face detector. But the higher diversity is expected to cause degradation in the precision. The face detections dataset gives us a prior judgment about the accuracies of the methods used. However, having the ultimate goal of designing an online system where encountered face tracks are to be classified, face tracks database gives a more valuable information about the performance of our methods in such severe conditions where face tracks may include heavily distorted faces via illumination conditions, rotations and facial expressions.

In all tests, 5-fold cross validation technique is used. The database to be used is divided into 5 groups. One group is used as the test group while the remaining 4 are used as the training group. At each fold upon calculating the recognition precision, the test group is switched. The average of the five test results gives the final precision value.

The rest of the chapter is as follows: First the illumination compensation algorithms used is described. Then the feature extraction and classification methods are explained. And finally the execution times and recognition accuracies of the methods for different conditions are presented.

Figure 3.1. A sample for illumination compensation. (a) Original Image. (b) Resized Gray Image. (c) Illumination Compensated Image.

3.2 Illumination Compensation

Illumination compensation is performed in order to refine face images for a better recognition performance. Since personal videos are recorded in uncontrolled environments, the illumination compensation is an essential step to deal with the diversity of illumination conditions which affect the pixel values. As the first step, sample face images are converted to gray-scale and resized to have a standard size of 64 (height) x 48 (width) pixels. Then, a basic and fast algorithm of illumination compensation by homomorphic filtering [45] is applied (See Equation 3.1). Figure 3.1 shows the process of the illumination compensation. When we take the logarithm of the raw image, by using a high pass filter we can amplify the reflectance field and suppress the luminance field. After taking back the exponential, the illumination compensation is achieved.

$$I_{comp}(x,y) = exp\big(log\big(I_{raw}(x,y)\big) * H_{hp}(x,y)\ \big) \qquad (3.1)$$

3.3 Feature Extraction

Instead of using direct pixel values, which are sensitive to noise and localization errors, three alternative methods have been implemented and details of each method are presented.

3.3.1 DCT Features

DCT features are used for face recognition in the literature [3]. The resized images of size 64 x 48 pixels are divided into 8x8 pixel blocks. For each of these small blocks, two-dimensional discrete cosine transform is applied. Ten of the resultant 8x8 transform coefficients are scanned in order to reduce the dimension of the 8x8 pixel block by representing the data block with 10 coefficients instead of 64. Coefficients which correspond to the higher frequency basis functions are less significant compared to the ones which correspond to the lower frequency basis functions; hence data compression can be attained by losing some high frequency information which can be affected more

easily by noise. Skipping the bias value, which is heavily dependent on illumination in the scene, first 10 DCT coefficients are scanned in a zigzag manner (Figure 3.2). Concatenating these 10 values for each block yields a vector of length 480. As the last step, this vector is normalized to have a unit norm.

3.3.2 LBP Features

Local Binary Patterns is originally used as a texture descriptor but in the literature its use as a facial feature extraction method has also been investigated [2]. Each pixel of the resized gray image is coded with an 8-bit string according to its 8 neighboring pixels. The value of the center pixel is compared to each of its 8 neighbors. If the value of the center pixel is greater than the neighbor, a 0 is assigned to the corresponding neighbor and a 1 is assigned otherwise. This pattern of 1's and 0's is unfolded to yield an 8 bit string. In Figure 3.3 the extraction of the 8-bit strings is shown.

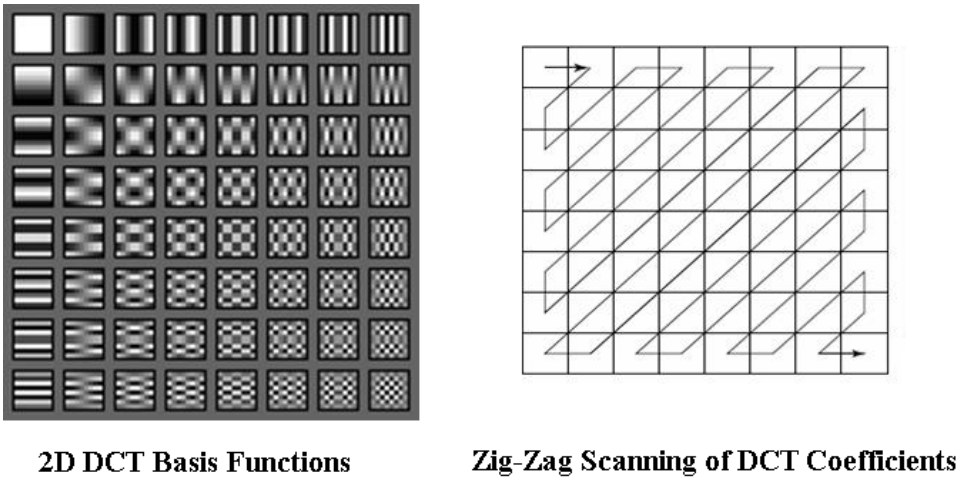

Figure 3.2. 2D DCT Basis Functions and the zigzag scanning method of the DCT Coefficients.

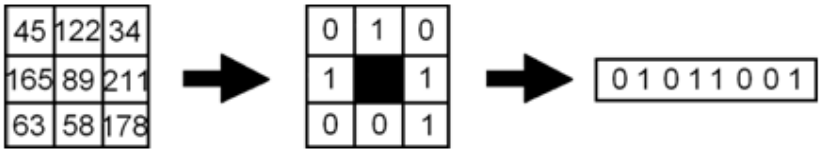

Figure 3.3. Demonstration for the extraction of 8-bit strings for Local Binary Patterns.

For each pixel a string is assigned and number of occurrences for these strings is counted to create a histogram of 8-bit strings. This histogram with 256 bins is used as a feature vector of 256 dimensions. This vector is normalized to have a unit norm.

3.3.3 HOG Features

Histogram of oriented gradients is a feature extraction method commonly used for human detection [42]. But it is also proven to be a suitable feature extraction method for human faces [44]. A gray-scale face image of size 64x48 pixels is divided into 8x8 pixel cells. For each cell horizontal and vertical gradients are extracted by 1D derivatives with simplest kernels of length 3. ([1 0 -1] for horizontal and [1 0 -1]T for vertical) then the magnitude and the angle of gradients are calculated at each pixel.

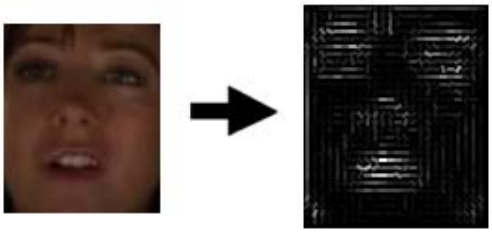

Figure 3.4. Extraction of the Histogram of Oriented Gradients feature extraction.

Orientation histogram is calculated for each cell where the scaled magnitudes are used as weights. 9 bins are used for [-π,π] interval. In order to do scaling for the weights, 2x2

cells are grouped together to yield one block of 16x16 pixels and a Gaussian mask is used to multiply the weights according to the spatial layout of the weights within the block. The center of the Gaussian mask is at the center of the blocks and the standard deviation is chosen as 8 pixels. 4 scaled histograms within the overall block are concatenated to give the overall descriptor for the particular region. This descriptor is also normalized by dividing it with its norm.

Blocks are created in an overlapping fashion where adjacent two blocks have common cells but each time scaled according to a different block. This way an overall feature vector for the face is constructed by concatenating the local feature vectors for the overlapping blocks, 35 blocks with each having 4x9 (number of cells x number of bins) dimensions result in an overall feature vector of size 1260. Figure 3.4 shows an example of the extracted gradients.

3.4 Classification

In this section, three classification method, Nearest Neighbourhood, Linear Discriminant Analysis and Support Vector with Single or Multiple Kernels will be discussed.

3.4.1 Nearest Neighbour (NN)

NN method is widely used in annotation applications especially as a baseline method. A query track is composed of several face samples i.e., F^Q=$\{ {f_1}^Q, {f_2}^Q, {f_3}^Q, \dots, {f_M}^Q \}$. Similarly the database contains the so-called "gallery" images, each of which is grouped according to its class label C, i.e., ${F_C}^G = \{ {f_{C,1}}^G, {f_{C,2}}^G, {f_{C,3}}^G, \dots, {f_{C,N}}^G \}$. A label is assigned to the query track by finding the highest similarity score among the competing gallery classes, using Equation 3.2. The similarity score is calculated using the normalized-correlation metric between query samples and the gallery samples of each class. The pair of samples which give the highest normalized-correlation value determines the overall score for the corresponding class (See Equation 3.3).

$$Label = arg \max_{C} S(F^Q, {F_C}^G) \tag{3.2}$$

$$S\left(F^Q, {F_C}^G\right) = \max_{{f_i}^Q \in F^Q} \max_{{f_j}^G \in {F_C}^G} \left(\frac{{f_i}^Q . {f_j}^G}{\|{f_i}^Q\| . \|{f_j}^G\|} \right) \tag{3.3}$$

The prominent problem with one-to-one comparison of the feature vectors is that having a growing database results in an increasing comparison time. The second problem is the possible domination of the crowded class vectors.

3.4.2 Linear Discriminant Analysis (LDA)

LDA is actually a feature extraction method usually followed by a classifier (here Nearest Neighbourhood). A projection onto a lower dimensional space is determined according to the higher dimensional raw data. The projection procedure is also applied for the query vector and the projected query vector is classified according to the labeled data again with the Nearest Neighbourhood method.

LDA yields the projection where the projected feature vectors are maximally separated between classes and minimally separated within their classes. To determine the projection matrix, Within-Class Scatter Matrix (See Equation 3.4) and Between-Class Scatter Matrix (See Equation 3.5) are constructed.

$$\boldsymbol{S_w} = \sum_{c=1}^{M} \sum_{\boldsymbol{c}} = \sum_{c=1}^{M} \sum_{\boldsymbol{x} \in \{\boldsymbol{x_c}\}} (\boldsymbol{x} - \overline{\boldsymbol{x}}_{\boldsymbol{c}})(\boldsymbol{x} - \overline{\boldsymbol{x}}_{\boldsymbol{c}})^T \tag{3.4}$$

$$\mathbf{S_b} = \sum_{c=1}^{M} n_c (\overline{\boldsymbol{x}}_{\boldsymbol{c}} - \overline{\boldsymbol{x}})(\overline{\boldsymbol{x}}_{\boldsymbol{c}} - \overline{\boldsymbol{x}})^T \tag{3.5}$$

Here M is the number of different classes, n_c denotes the number of sample data belonging to class c, $\overline{\boldsymbol{x}}_{\boldsymbol{c}}$ denotes the mean of the vectors belonging to class c, $\overline{\boldsymbol{x}}$ denotes the mean vector of all sample data. The projection direction is found by calculating the eigenvectors of Equation 3.6. The number of selected eigenvectors corresponds to the dimension of the projected space. We retain the eigenvectors with the highest eigenvalues that capture the 95 % of the total energy where total energy is sum of all the eigenvalues.

$$\boldsymbol{D = S_w^{-1} S_b} \tag{3.6}$$

Since the within-class scatter matrix is a summation of outer-product of vectors, it is generally singular. In order to guarantee that the within-scatter matrix is non-singular, hence invertible, the number of training data must be greater than the summation of the number of input dimensions and the number of classes to be determined [4].

3.4.3 Support Vector Machines with Single and Multiple Kernels

Support Vector Machine is a binary classification method which tries to linearly separate samples from 2 classes according to some criteria. A separating hyperplane is defined to partition the sample space into two regions. The hyperplane is demanded to be as far from the nearest sample vectors as possible to provide a clear gap between the two classes. Although SVM is a linear classifier, using different Kernel functions the samples can be projected to different spaces. A sample set can be linearly separated on a transformed space even though it may not be in the original space. Instead of forcing the SVM to strictly discriminate all samples, a soft margin can be used. Using a soft margin value, the separating hyperplane can be fit to discriminate linearly inseparable data, although there is a penalty term for each of the data points which reside on the wrong side of the hyperplane.

SVM is a binary classifier but it can be extended to make classification for multiple classes. There are a couple of methods for multi-class classification. The first one involves the solution of the optimization problem for all classes at once. But this method performs poorly. Another approach is the “one vs. one” where an SVM model is trained for each pair of classes. A query sample is tested with each SVM model and the class label is assigned according to the highest number of class labels presented. The problem with the method is the rapidly increasing number of SVM Models with increasing number of classes. For N classes, $\frac{\boldsymbol{N(N-1)}}{\boldsymbol{2}}$ SVM models have to be trained. The last option is the “One vs. The Rest” method. For each class, an SVM model is trained. And the query samples are tested with each model. There may be some ambiguous situations with this method where the query vector ends up belonging to a number of classes more than

one or none at all. In order to decide the winner class, classification is done according to real decision values instead of the sign of the decision values. In this work we have adopted the "One vs. The Rest" method for multiclass classification. Gaussian RBF Kernel function (See Equation 3.7) is used for the single kernel SVM. Similarly for the Multiple Kernel SVM, Gaussian RBF kernels are used as base kernels.

For the multiple kernel case, instead of using a single kernel, linear combinations of base kernels are considered (See Equation 3.8). Each base kernel corresponds to a different block of the feature vector. For the LBP, DCT and HOG features, these base kernels are applied for each of the data blocks created during the feature extraction stage. Hence the number of base kernels is 35 for HOG features and 48 for DCT features. For the LBP features with 256 dimensions, vectors are partitioned into 16 blocks of length 16 hence 16 base kernels are used. K_b's are the base kernels and β_b are the weights corresponding to each base kernel. N is the total number of base kernels. Multiple Kernel Learning algorithm [28, 31] can be used to select the best β_b value over a training surplus, but it has been shown that assigning equal weights for each weight provides sufficient results [44].

We have selected the soft-margin coefficient C and γ parameter of the RBF Kernel by conducting a test across a rectangular grid of C and γ variables. For the Multiple Kernel SVM, for each base kernel K_b, γ_b parameters are selected separately. Each base kernel corresponds to a distinct block in the feature vector. In order to define γ_b parameter for each block, the average Euclidean distances (See Equation 3.9) between blocks are calculated separately over a training data. For two of the training feature vectors, namely u and v, u_b and v_b are blocks of the feature vectors defined by the block number b (where parameters are selected as $\gamma_b = \frac{d_b}{8}$ for HOG features , $\gamma_b = d_b \, x \, 32$ for DCT features and $\gamma_b = d_b/2$ for LBP features.

$$\mathrm{K(u,v)} = \exp(-\gamma|u - v|^2) \tag{3.7}$$

$$K(u,v) = \sum_b \beta_b K_b(\mathrm{u,v}) \qquad \mathrm{K_b} = \exp(-\gamma_b|u - v|^2) \qquad \beta_b = \frac{1}{N} \tag{3.8}$$

$$d_b = |u_b - v_b| \qquad b \in \begin{cases} [1,16] & for\ LBP\ features \\ [1,48] & for\ DCT\ features \\ [1,35] & for\ HOG\ features \end{cases} \tag{3.9}$$

3.5 Experiments and Results

3.5.1 Overview

In order to compare execution times and recognition precisions, all feature extraction methods are tested with all classification methods. The execution times of all methods are plotted for both training and testing phases as the number of training and testing samples changes. In order to calculate the precision of face recognition algorithms, several tests have been made to compare the accuracies of the methods under different conditions. Accuracies are calculated as the number of target classes is varied. Similarly the accuracy is plotted as the amount of training data is varied. All tests are made for two distinct face datasets. The first dataset is a collection of hand-labeled face detection outputs. Viola-Jones face detector [1] has been run for every single frame throughout an episode of "How I Met Your Mother" TV series and the detected faces of target people are manually clustered. The second dataset is created by using a face tracker algorithm on the same episode. OpenCV implementation of Camshift Color Based face tracker [6] is used in order to track faces throughout the video. The resulting face tracks are also manually labeled and clustered. Samples from both datasets are fed to an illumination compensation algorithm before recognition is performed.

The face detections database is composed of more-controlled and better-posed face images since the face detector captures frontal upright faces. The face tracks database on the other hand, captures a much more diverse set of face images as the tracker can collect rotated or otherwise distorted faces which may be missed by the face detector. But the higher diversity is expected to cause degradation in the precision.

The face detections dataset gives us a prior judgment about the accuracies of the methods used. However, having the ultimate goal of designing an online system where encountered face tracks are to be classified, face tracks database gives a more valuable information about the performance of our methods in such severe conditions where face

tracks may include heavily distorted faces via illumination conditions, rotations and facial expressions.

In all tests, 5-fold cross validation technique is used. The database to be used is divided into 5 groups. One group is used as the test group while the remaining 4 are used as the training group. At each fold upon calculating the recognition precision, the test group is switched. The average of the five test results gives the final precision value.

3.5.2 Execution Times

All of the tests are performed on an Intel Core 2 Duo 2.20 GHz PC with 1 GB RAM. DCT and HOG feature extraction methods has been implemented in MATLAB. For LBP feature extraction, a MATLAB implementation available in [30] has been used. Nearest Neighborhood and LDA algorithms are implemented in MATLAB environment. For Single Kernel and Multiple Kernel SVM, a MATLAB interface for LIBSVM implementation [11] is used. Training and testing times of all methods are presented. For the calculation of the execution times, the number of classes is selected to be 6. The number of training samples is varied and the corresponding execution time for each method is calculated.

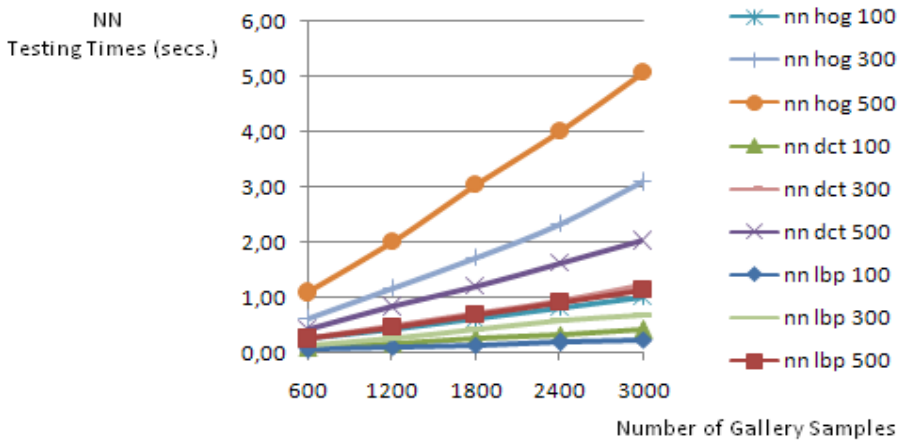

Figure. 3.5. The graph of NN testing times vs. the number of gallery samples.

3.5.2.1 Nearest Neighborhood

For NN, there is no training. The execution times are shown for different numbers of gallery samples (Figure 3.5). Gallery samples correspond to the number of training samples where each test sample is compared one by one before the nearest neighbor is found. For each method, three different graphs are plotted with different numbers of testing samples used. As expected, the execution times increase linearly with the number of gallery samples with a small bias value. Also given a fixed number of gallery samples, the execution times are observed to increase linearly with the number of testing samples.

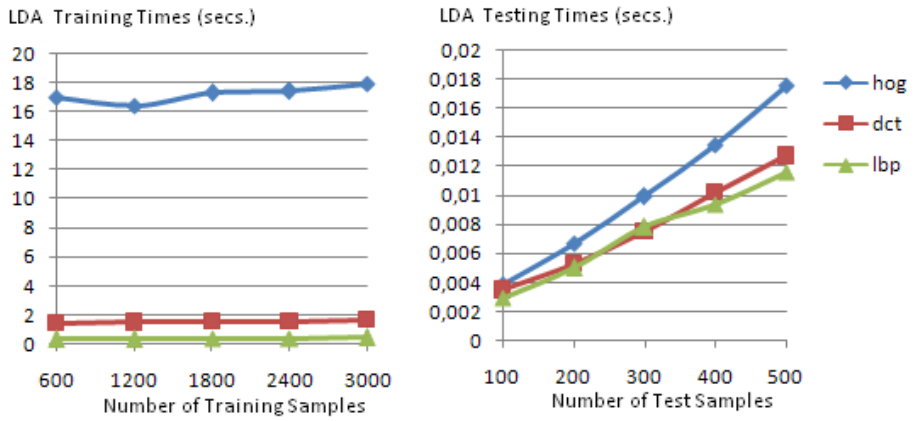

Figure 3.6. LDA Training/Testing times vs. the number of Training/Testing Samples

3.5.2.2 LDA

For the LDA training, the graphs of execution times (See Figure 3.6) are considerably flat with respect to the number of training samples, on the other hand a major change is observed for different features. This is expected since most of the computation is due to the construction of the scatter matrices and calculation of the eigenvectors where the number of feature vector dimensions determines the size of the scatter matrices. 3000 samples are used to construct an eigenspace model, and this model is used to conduct tests for the LDA execution times. The testing times, which have the smallest values

among all classification methods, increase linearly with the increasing number of test samples.

3.5.2.3 SVM

SVM training times are shown on Fig. 6. The execution times increase rapidly with the increased number of training samples. The execution times are for the training of 6 SVM models. Most of the computation is due to the kernel construction. For each test with the designated number of training samples, a single common kernel is constructed and the corresponding training labels are adjusted for the training of different SVM models. Hence SVM training times do not increase linearly with the "number of classes". 6 SVM models are constructed using 3000 samples for each model. As in training, most of the computation is due to the construction of kernels; hence the number of returned support vectors is indicated for each method. The linear behavior with respect to the number of testing samples can be seen from the graph (See Figure 3.7). Having returned a smaller number of support vectors, Multiple Kernel SVM models have smaller testing times compared to the SVM with DCT features.

3.5.3 Recognition Precisions

The number of samples per person is varied (with 6 people as target classes) and the corresponding classification performance is measured for both detection and tracking databases (Figures 3.8-3.10). Similarly the number of target classes is varied (with 500 samples for each class) and the resulting classification rate is measured for the detections and the tracks databases. The performance is measured by the precision at 100 % recall which means that all of the face tracks are labeled and the percentage of correct labeling is taken as the performance.

The general trend of decreasing recognition precision in Figures 3.8-3.10 is due to the fact that while the number of samples per person increases, the diversity of the model also becomes more complex with different looking samples of the face. If the variety of the samples were kept constant, the addition of similar samples would increase the precision.

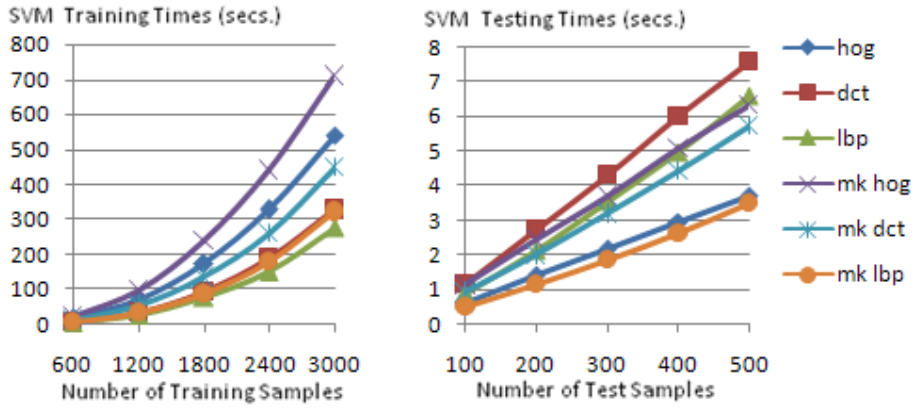

Figure 3.7. SVM Training/Testing times vs. the number of training/testing samples.

In all tests, LDA classification is observed to degrade and perform poorly with the insufficient number of training samples. But when the number of samples is adequate, LDA classification also works properly. SVM performs as the best classification method, but SVM training is a heavy process compared to NN and LDA. Single Kernel SVM with DCT features works best for both detections and tracks datasets. But the testing time is higher than other SVM methods. This is due to the fact that the extracted support vectors are higher in number than any other SVM method. The highest recognition accuracies for each classification method are tabulated. The feature extraction method which yields the highest accuracy is indicated in parentheses. Also the training and testing times for these best methods are listed in Table 1 for comparison.

3.5.4 Observations

We have observed that single kernel SVM trained with DCT features gives the highest recognition accuracy (See Figure 3.8). On the other hand in this method, the number of Support Vectors found is great and this yields relatively long testing times. SVM with Multiple Kernels, on the other hand, have comparable recognition accuracy to the single kernel SVM, though training times are longer due to a more complex process of kernel construction. But tests show that in multiple Kernel SVM methods, fewer number of support vectors is sufficient to define the separating hyperplane which led to shorter testing times.

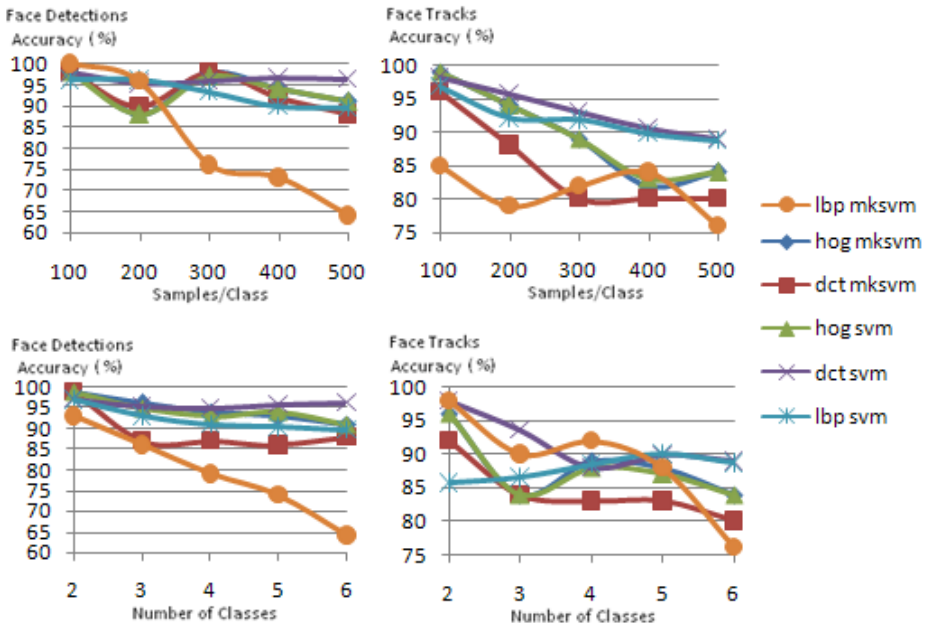

Figure 3.8. Recognition accuracies of SVM methods for Face Detections and Face Tracks Datasets vs. Samples/Class and Number of Classes.

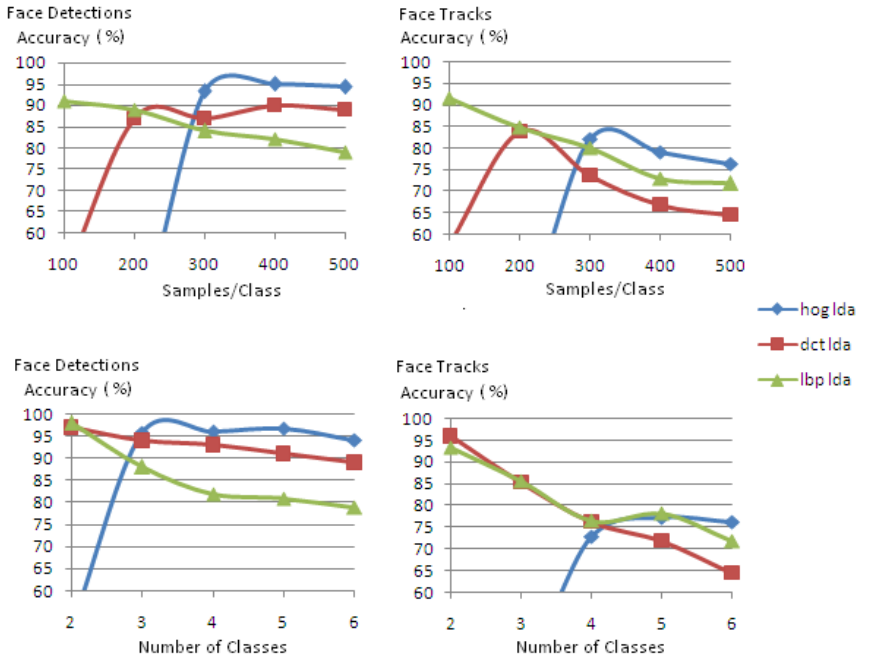

Figure 3.9. Recognition accuracies of LDA methods for Face Detections and Face Tracks Datasets vs. Samples/Class and Number of Classes.

LDA has the fastest testing times of all but the recognition accuracy is lower than the other methods (See Figures 3.8-3.10). Also depending on the feature extraction method used, a large volume of training data is necessary before proper LDA classification can be accomplished. Nearest Neighborhood has the problem of ever-growing database with addition of new samples and classes with higher number of samples may dominate the classification process.

There is a tradeoff between testing times and training times when we consider the usage of SVM with single and Multiple Kernels. If long testing times are acceptable, Single Kernel SVM with DCT coefficients has the highest recognition accuracy. For the online automatic face annotation system where encountered face tacks are to be classified, the number of testing samples per query is not large, as we have determined this number as 10 samples per query track. Training times can be considered more important for an online learning system as the newly encountered samples are sequentially learnt in data chunks, repetitive sessions of long trainings may discourage the user from working with the system.

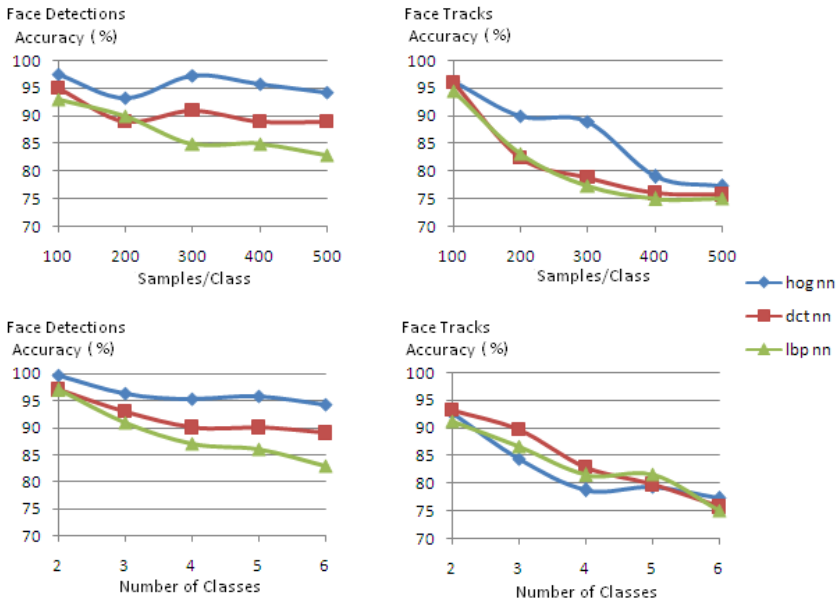

Figure 3.10. Recognition accuracies of NN methods for Face Detections and Face Tracks Datasets vs. Samples/Class and Number of Classes.

It is observed that switching from face detections dataset to face tracks dataset results in a noticeable decrease in recognition accuracies (See Figures 3.8-3.10). This degradation for NN and LDA methods are larger than the degradation for the SVM methods with both single and multiple kernels. The results indicate that SVM methods can adapt to a wide variety of samples better than the LDA and NN methods.

Table 3.1. Performance Summary. Best recognition accuracy for each classification method is presented with their corresponding training and testing times.

	Accuracy (Detections Dataset)	Accuracy (Tracks Dataset)	Training Times	Testing Times
MKSVM	91 % (HOG)	84 % (HOG)	324 secs. (LBP)	3.48 secs. (LBP)
SVM	96% (DCT)	89 % (DCT)	326 secs. (DCT)	7.57 secs. (DCT)
LDA	94 % (HOG)	76 % (HOG)	17.9 secs. (HOG)	0.12 secs. (HOG)
NN	94 % (HOG)	77 % (HOG)	-	5.06 secs. (HOG)

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 Overview

Face recognition methods are tested for an online learning system where face identification is performed for video annotation. A user interface is designed for the system (Figure 4.1) where a minor user interaction is requested to teach the system as the samples of each target face are encountered. The user browses through the video from one shot boundary frame to the next and trains the learner with tracks of face images extracted at each shot boundary frames. Once the number of collected samples exceeds the selected threshold, the system starts making classification and predicted class labels are presented to the user. The training and classification steps occur simultaneously after this point. For online learning purposes, extensions to the offline learning algorithms have been developed in order to achieve effective learning which involves updating the learner model sequentially instead of redoing the whole learning process from scratch.

In the rest of the chapter, the sequential variants of the considered classification algorithms will be explained then the user interface and learning methodology will be presented. Finally the results in terms of recognition accuracy will be presented and discussions be made.

4.2 Implementation of the Sequential Classification Methods

Extensions to the batch learning algorithms are discussed in this section. Sequential variants of LDA and SVM algorithms have been implemented along with the NN algorithm for the online video annotation application.

4.2.1 Nearest Neighbourhood

The sequential version of the Nearest Neighbourhood classifier is identical to the offline version discussed in detail in Section 3.4.1. Tracks of faces are simply added to the database with their labels each time a new track of face is learnt. During classification query tracks are compared with the database samples to find the most similar pair of samples to produce a label.

4.2.2 Chunk Incremental LDA

In the training phase, an incremental method known as Chunk Incremental LDA is used to create a dynamic fisherspace model which gets updated whenever a new chunk of training data with L samples ($L \geq 1$) and a given class label is received.

This method combines the fisherspace models of the existing data and the new chunk of training data. To find the parameters for the new eigenspace, within-class scatter matrix (Sw) and between-class scatter matrix (Sb) are to be updated.

$$\bar{\boldsymbol{x}}' = \frac{N\bar{\boldsymbol{x}} + L\bar{\boldsymbol{y}}}{N + L} \tag{4.1}$$

The overall mean is updated according to Equation 4.1 where $\bar{\boldsymbol{x}}$ is the mean of the existing data of N samples and $\bar{\boldsymbol{y}}$ is the mean of new coming L samples.

If the new samples belong to class *c*, then the class-mean for class *c* is updated according to Equation 4.2.

$$\bar{\boldsymbol{x}}'_{\boldsymbol{c}} = \begin{cases} \frac{1}{(n_c + L)}(n_c\bar{\boldsymbol{x}}_{\boldsymbol{c}} + L\bar{\boldsymbol{y}}), & if\ c\ is\ an\ existing\ class \\ \bar{\boldsymbol{y}}, & if\ c\ is\ a\ new\ class \end{cases} \tag{4.2}$$

where n_c is the number of existing samples belonging to class c.

If new samples belong to an existing class, the within-class and between-class scatter matrices are updated according to Equation. 4.3 & 4.4.

$$\boldsymbol{Sb}' = \sum_{C=1}^{\boldsymbol{M}} n_c'(\overline{\boldsymbol{x}}'_{\boldsymbol{c}} - \overline{\boldsymbol{x}}')(\overline{\boldsymbol{x}}'_{\boldsymbol{c}} - \overline{\boldsymbol{x}}')^{\boldsymbol{T}} \tag{4.3}$$

$$\boldsymbol{Sw}' = \boldsymbol{Sw} + \frac{\boldsymbol{n_c L^2}}{(\boldsymbol{n_c + L})^2}(\overline{\boldsymbol{y}} - \overline{\boldsymbol{x}}_{\boldsymbol{c}})(\overline{\boldsymbol{y}} - \overline{\boldsymbol{x}}_{\boldsymbol{c}})^{\boldsymbol{T}} + \frac{\boldsymbol{n_c}^2}{(\boldsymbol{n_c + L})^2}\sum_{\boldsymbol{j=1}}^{\boldsymbol{L}}(\boldsymbol{y_j} - \overline{\boldsymbol{x}}_{\boldsymbol{c}})(\boldsymbol{y_j} - \overline{\boldsymbol{x}}_{\boldsymbol{c}})^{\boldsymbol{T}}$$
$$+ \frac{\boldsymbol{L(L+2n_c)}}{(\boldsymbol{n_c + L})^2}\sum_{\boldsymbol{j=1}}^{\boldsymbol{L}}(\boldsymbol{y_j} - \overline{\boldsymbol{y}})(\boldsymbol{y_j} - \overline{\boldsymbol{y}})^{\boldsymbol{T}} \tag{4.4}$$

If the samples belong to a new class, then the within-class and between-class scatter matrices are updated according to eqn. 4.5 & 4.6,

$$\boldsymbol{Sb}' = \sum_{C=1}^{\boldsymbol{M+1}} n_c'(\overline{\boldsymbol{x}}'_{\boldsymbol{c}} - \overline{\boldsymbol{x}}')(\overline{\boldsymbol{x}}'_{\boldsymbol{c}} - \overline{\boldsymbol{x}}')^{\boldsymbol{T}} \tag{4.5}$$

$$\boldsymbol{Sw}' = \boldsymbol{Sw} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{L}}(\boldsymbol{y_j} - \overline{\boldsymbol{y}})(\boldsymbol{y_j} - \overline{\boldsymbol{y}})^{\boldsymbol{T}} \tag{4.6}$$

Where M is the number of classes, n_c' is the updated number of samples belonging to class c, $\overline{\boldsymbol{x}}'$ is the updated global mean value, $\overline{\boldsymbol{x}}'_{\boldsymbol{c}}$ is the updated class-mean for class c and y_j is one of L new samples.

Once the updated within-class scatter matrix and between-class scatter matrices are computed, the projection direction is again found by calculating the eigenvectors of (3.6). As we stated earlier, we retain the eigenvectors with the highest eigenvalues that capture the 95% of the total energy. So the dimension of the projected space may change throughout the learning course after each new training phase. Number of the training tracks used for each class is recorded. All the samples used for the construction of the eigenspace is projected on the fisher-eigenvectors. Projected samples for each class are clustered using K-Means algoritm where K is chosen as the number of the training tracks

for the corresponding class. Therefore the projected space is represented by $K_C = \{K_1, K_2, \dots, K_M\}$ cluster centers for M classes. This clustering scheme is applied as a precaution against the possible domination of crowded classes over classes which have a fewer number of samples. But to maintain balance, higher number of representative cluster centers are donated for classes which have more tracks.

In the classification phase, samples in the query track are projected and their similarity with the cluster centers of projected training data are measured according to the method of Nearest Neighborhood with normalized correlation metric. (See Equations 3.2 and 3.3) Classification time for each query track is constant and independent of the size of the database.

4.2.3 Sequential SVM

For *N* different people in the database, namely *N* classes, *N* set of SVM's are trained for "1 vs. the Rest" classification. Thus, the training process is very heavy. But the classification time for each query sample is constant and independent of the size of the database as in the case of LDA. However, since the training phase requires time, SVM is not appropriate for real-time applications. Instead sequential SVM (SSVM) is used in our application.

There is more than one method to utilize Sequential SVM as presented in [12, 40]. The one preferred in this work is to retrain the previously found Support Vectors along with the newly added training data to give the final classifier for the final corpus of data. This method is known as "the fixed partition" method [12].

The training session is initiated upon collecting samples from two distinct classes and one SVM model is created for 'Class 1 vs. Class 2'. This initial training stage is identical to the batch-SVM learning algorithm. After the initiation, whenever the user feeds a new track of faces to the learner, depending on the input class label, a sequential learning algorithm is called. If the new data belongs to a new class, an SVM-model is trained for 'Class 3 vs. The Rest' classification from scratch by using the Support Vectors returned from the initial SVM model of 'Class 1 vs. Class 2'. These support vectors are labeled with class labels of '0' and added together with the new samples which are labeled with class labels of '1'. Also SVM model for 'Class 1 vs. Class 2' is replaced with two SVM models, namely 'Class 1 vs. The Rest' and 'Class 2 vs. The Rest' by adding 'Class 3'

data with class labels of ‘0’, together with the initially found Support Vectors and retraining.

The sequential learning is performed in the same way for addition of other new classes or new samples for the existing classes. Using only the Support Vectors, the training times are reduced drastically compared to the case of retraining each SVM from scratch whenever a new group of data arrives.

4.3 User Interface

Any video can be loaded via this interface and there is a button which starts the shot boundary detection. The shots are detected automatically. The first frame of the shot is presented to the user on the interface and the face detection is run on it. The detected faces are marked with a rectangle on the screen (Figure 4.1). In order to make annotations for the training session, the user is allowed to either choose a name from the pool of available names or add a new label for the detected face. The user chooses a label for the detected face by clicking on the name and one of the detected faces. If an available face in the scene is missed by the face detector, the user can place a rectangular box and resize it in order to mark the undetected face. A tracker is instantly run upon clicking on the face to collect as many sample face images of the same person as possible. The tracking continues from one shot boundary frame to the next or until the tracking is stopped due to other causes like the face exiting the scene etc. The behavior of the tracker is shown on a separate window (Figure 4.2) after each track command. If the user is not satisfied with the current track, it can be discarded. Otherwise it is added to the database.

After collecting sufficient number of data using the face tracker, the recognition procedure can be invoked by clicking a button on the interface. For Nearest Neighbourhood recognition is performed after two distinct characters are introduced to the system by the user. The addition of face samples to the database continues until 500 samples are collected for each target class. On the other hand, the minimum number of samples needed for LDA is equal to the number of target classes plus the feature vector dimensionality (256 for LBP, 480 for DCT and 1260 for HOG) and sequential training of the model is performed until approximately 500 samples/class is reached.

For single and multiple kernel SVM methods, recognition is performed after 150 samples/class becomes available. This number is chosen experimentally, considering a trade-off between long training times and good representation skill for the class. Face tracks are populated in a queue for each class separately and whenever 150 samples become available for each class, the first 150 of the samples per class in the queue are fed to the learner and sequential SVM training is done together with the existing support vectors for each class. The remaining samples in the queue are kept for the next phase of training which starts whenever 150 samples become available in the queue for each class again.

Whenever a face is encountered, a face recognition algorithm is executed to make the classification according to the data collected so far. Each newly detected face is also tracked by the face tracker automatically for a number of frames (chosen as 10) and each sample of this short track is classified according to our face recognizer. The user may acknowledge these suggested labels or correct them with the actual names by again clicking on the faces and selecting names from the list. And the system is updated with the tracks of the correctly labeled faces.

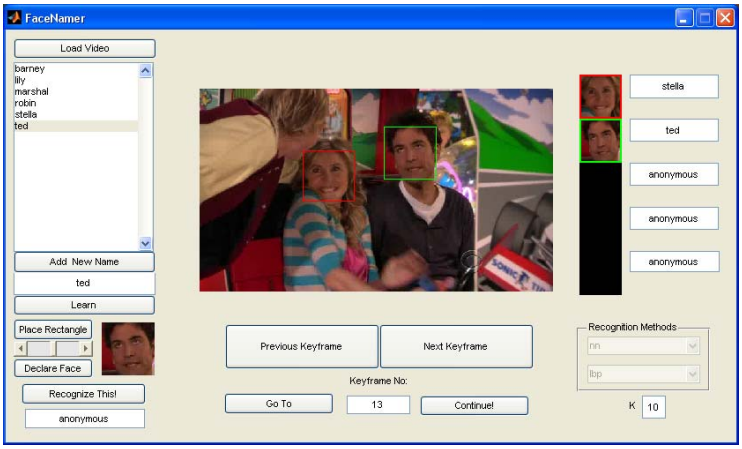

Figure 4.1. The Main Graphical User Interface

Figure 4.2. Demonstration of the Face Tracking along with the tracked face.

4.4 Tests and Results for Video Annotation

4.4.1 Performance Criteria

In order to evaluate the precision of the video annotation tool, the episodes of TV series "How I Met Your Mother" have been used. Six of the most prominent characters are selected as targets whose faces are to be recognized by the system (See Figure 4.3). We have used Precision/Recall graphs to evaluate the overall recognition success where Precision and Recall are defined according to the Equation 4.7 & 4.8.

$$Recall = \frac{Number\ of\ Classified\ Face\ Tracks}{Number\ of\ All\ Face\ Tracks} \tag{4.7}$$

$$Precision = \frac{Number\ of\ Correctly\ Classified\ Face\ Tracks}{Number\ of\ All\ Classified\ Face\ Tracks} \tag{4.8}$$

Figure 4.3. Six main characters of the TV series "How I Met Your Mother" that are selected as the target classes for our face identification system.

Recall parameter is varied according to a "refusal-to-classify" criterion. Whenever a query face track is classified, a similarity measure is also given with the suggested label. We reject classifications with similarity measures below a threshold. Refusal-to-classify method aims to reject intruders such as people who are not of interest or false face detections which does not contain a face. We expect to receive low similarity measures for these kinds of intruders so that they may be eliminated without receiving labels. Similarity measure for a query sample is chosen as the normalized correlation value according to Equation 3.3. if NN or LDA is used. For SVM classification, the distance to the decision hyperplane is selected as the similarity score where the query sample is assigned to the class which gives the greatest distance value. Here the distance values may be negative if a given query sample is classified as a false sample by all SVM models. Then the class which produces the greatest negative value (smallest absolute value) as the distance is chosen.

Face detections which do not produce successful face tracks are discarded and are omitted in the Precision/Recall evaluations. Those face detections, which may or may not include a face, that are too small or blurry may not produce successful face tracks as our Lucas-Kanade face tracker involves extraction of corners which represent the facial feature points to be tracked. If the detected face is too small or too blurry, then corners may not be extracted and face tracking fails. (See Figure 4.4)

Figure 4.4. An example detected face which does not produce a face track.

In the following Precision/Recall calculations for each method, classification results for successful face tracks that are extracted from the first 200 shot boundary scenes are evaluated and two Precision vs. Recall graphs are plotted for two different evaluation scenarios: “closed set identification” and “overall identification”.

“Overall Identification” graph considers all successful face tracks during precision and recall calculations, taking into account the tracks of false face detections and non-target faces whereas the “Closed Set Identification” plots consider only the precision scores attained after discarding the face tracks which do not belong to one of the target faces. As the name implies the “Closed Set Identification” is a face recognition application within the exclusive group of the target faces.

4.4.2 Discussions on the Performances of Online Learner Methods

"Closed set identification" scores are expected to give higher precision scores than the "overall identification" precision scores as models are constructed considering the target faces alone, without taking intruders into account. Feature vectors of target faces are expected to lie in closer formations that may be separated from each other using some classification methods as models are fit to discriminate one target class from another. On the other hand, intruders include faces of people which are not of interest or false face detection windows which do not contain faces at all. Feature vectors of these intruder samples are scattered all around the feature space and cannot be simply modeled due to the huge diversity involved. In SVM learning with "1 vs. the rest" method, samples of different faces are discriminated from each other. But this discrimination is solely based on face images and discrimination of target faces from intruders may not be optimally performed. As a result a fall in precision scores is observed (See Figures 4.5-4.16) especially for SVM methods (See Figures 4.11-4.16) when closed set and overall identification results are compared.

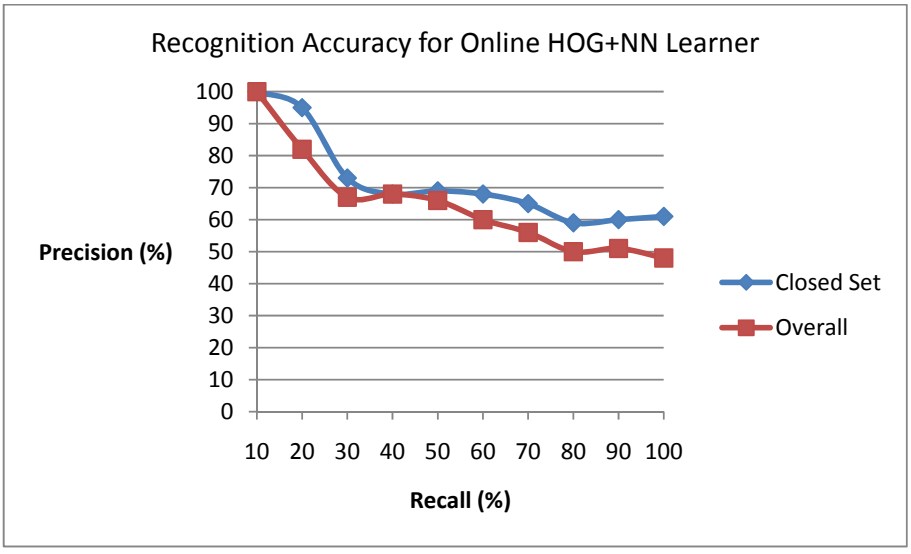

Figure 4.5. Precision vs. Recall Graph for Online HOG+NN learner.

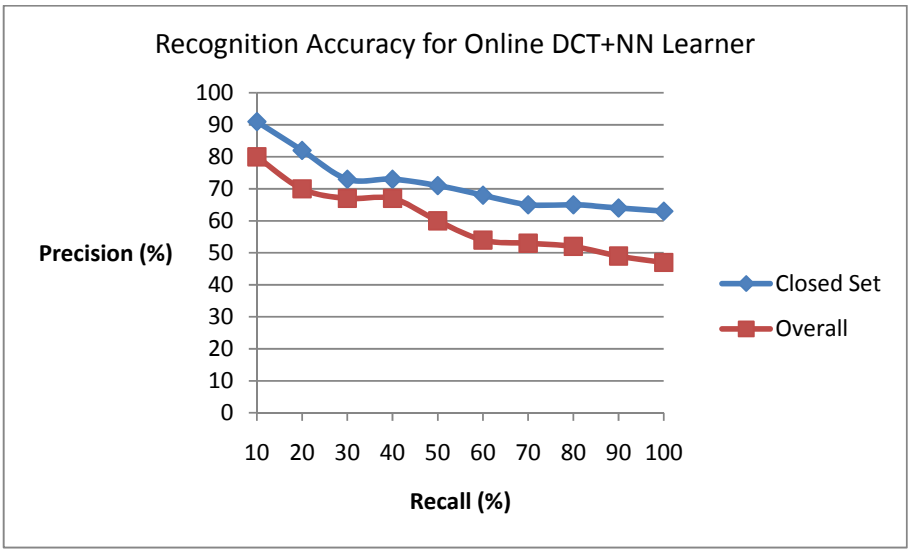

Figure 4.6. Precision vs. Recall Graph for Online DCT+NN learner.

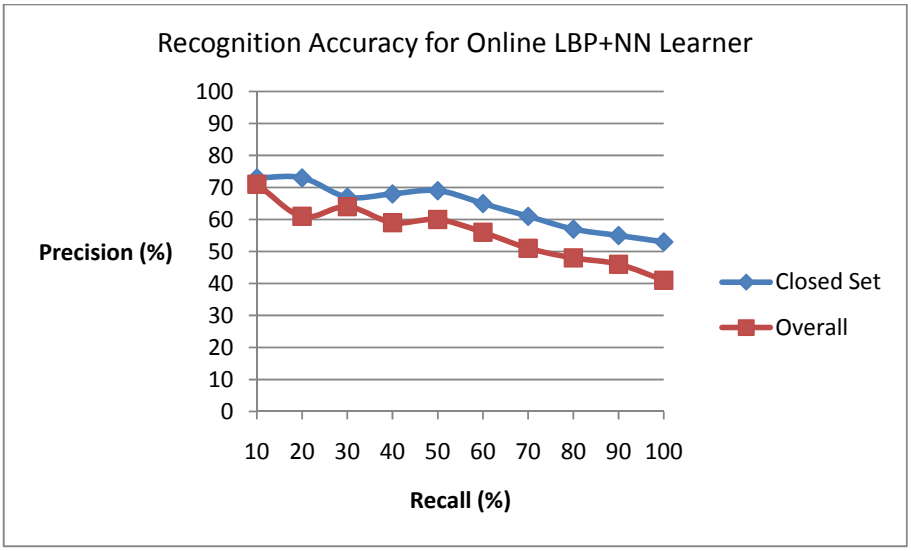

Figure 4.7. Precision vs. Recall Graph for Online LBP+NN learner.

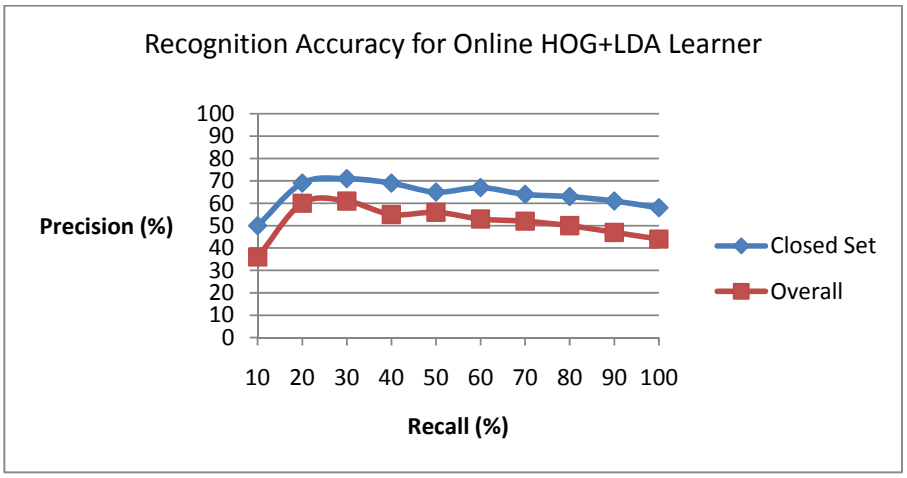

Figure 4.8. Precision vs. Recall Graph for Online HOG+LDA learner.

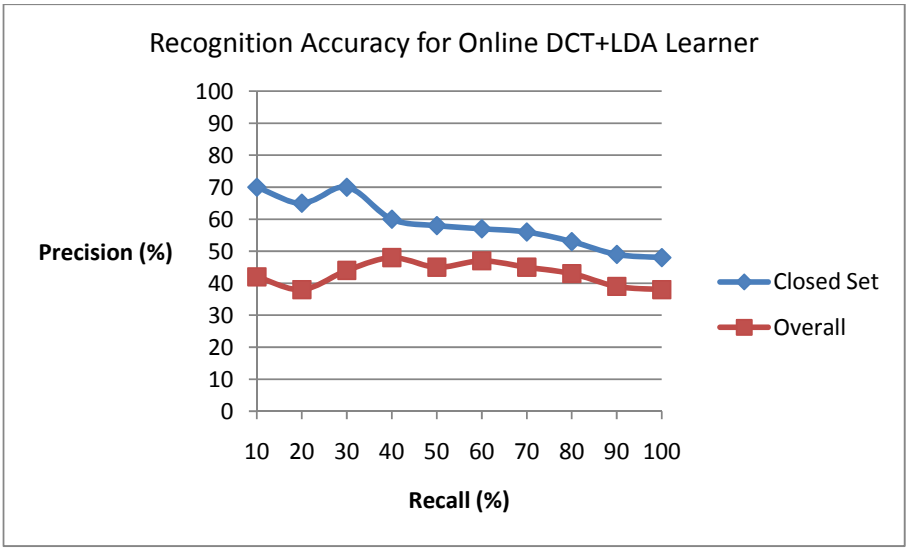

Figure 4.9. Precision vs. Recall Graph for Online DCT+LDA learner.

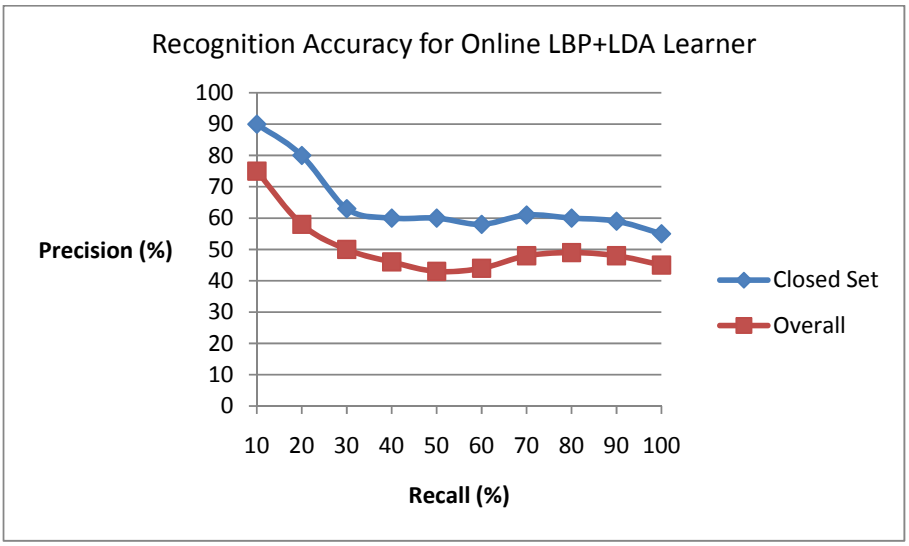

Figure 4.10. Precision vs. Recall Graph for Online LBP+LDA learner.

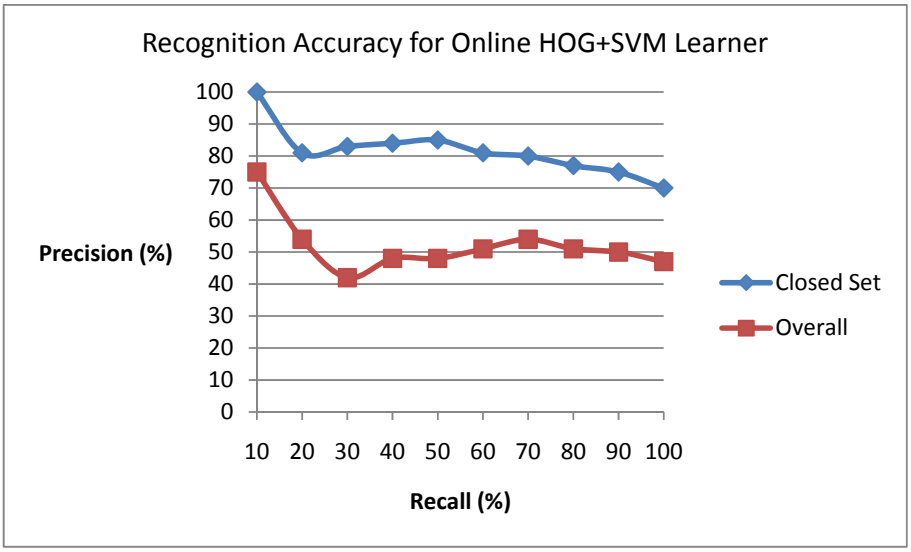

Figure 4.11. Precision vs. Recall Graph for Online HOG+SVM learner.

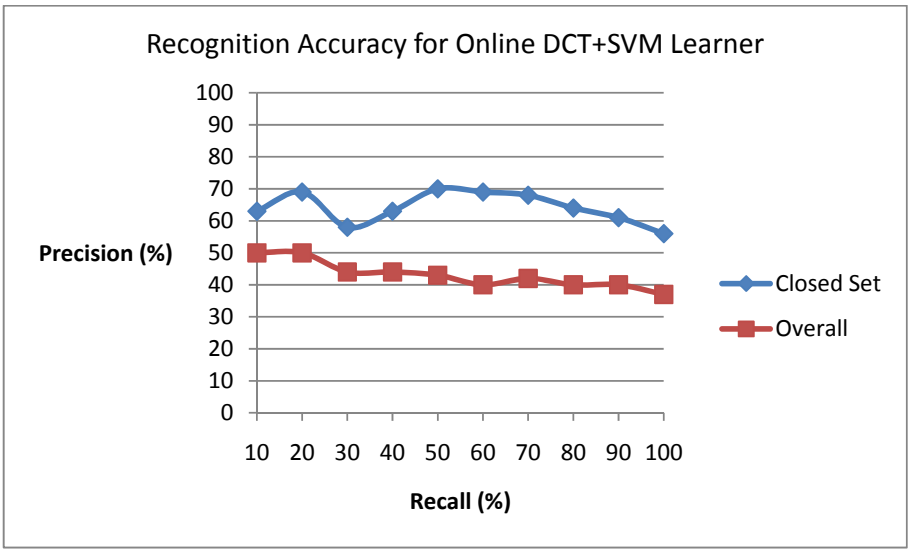

Figure 4.12. Precision vs. Recall Graph for Online DCT+SVM learner.

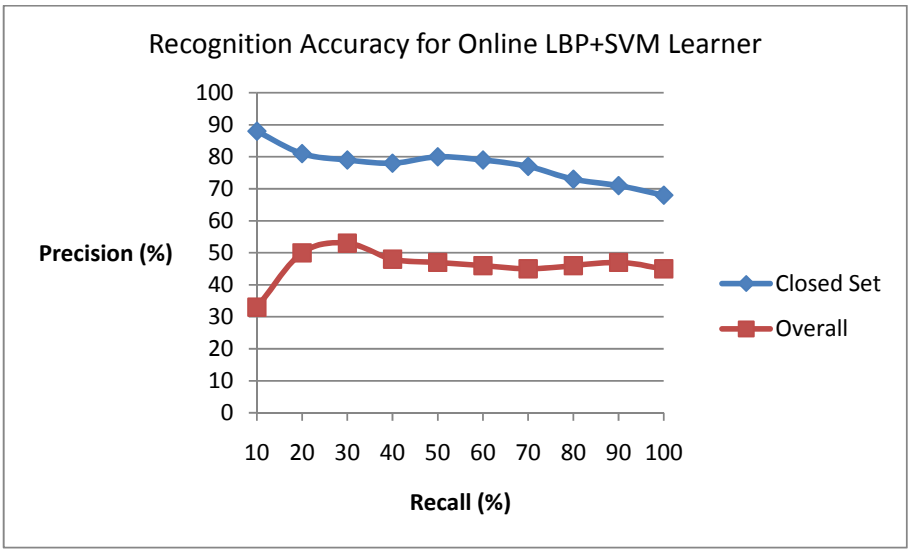

Figure 4.13. Precision vs. Recall Graph for Online LBP+SVM learner.

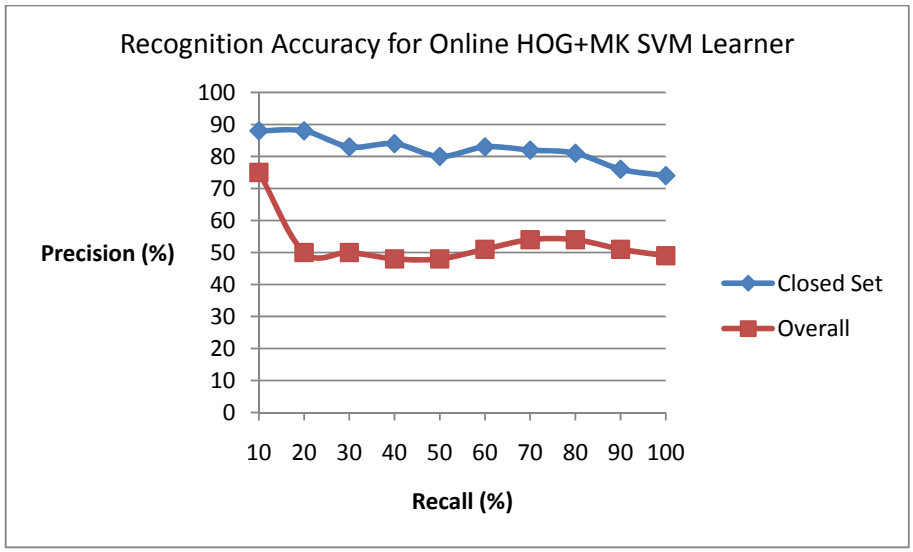

Figure 4.14. Precision vs. Recall Graph for Online HOG+MK SVM learner.

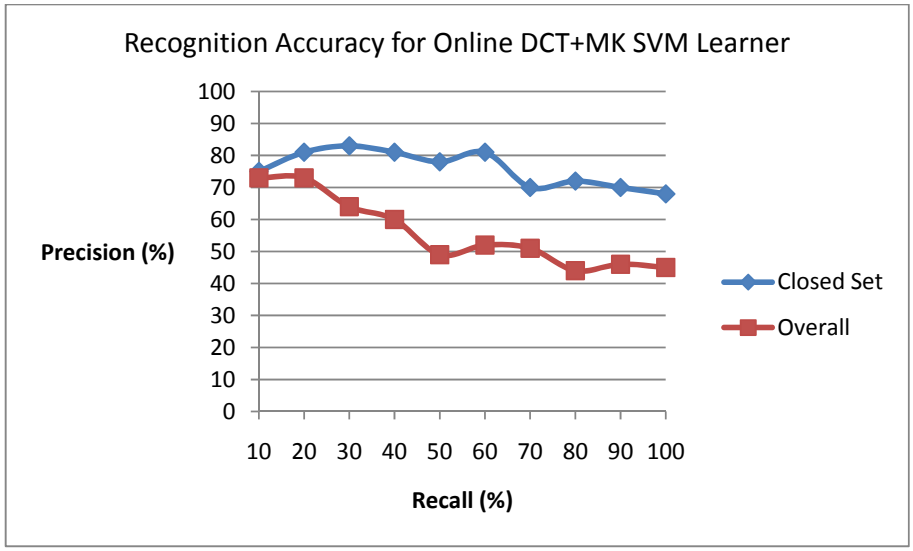

Figure 4.15. Precision vs. Recall Graph for Online DCT+MK SVM learner.

Ideally we expect a monotonic and sharp decrease with increasing recall values. The test results (See Figures 4.5-4.16) show an overall trend of decrease although monotonic

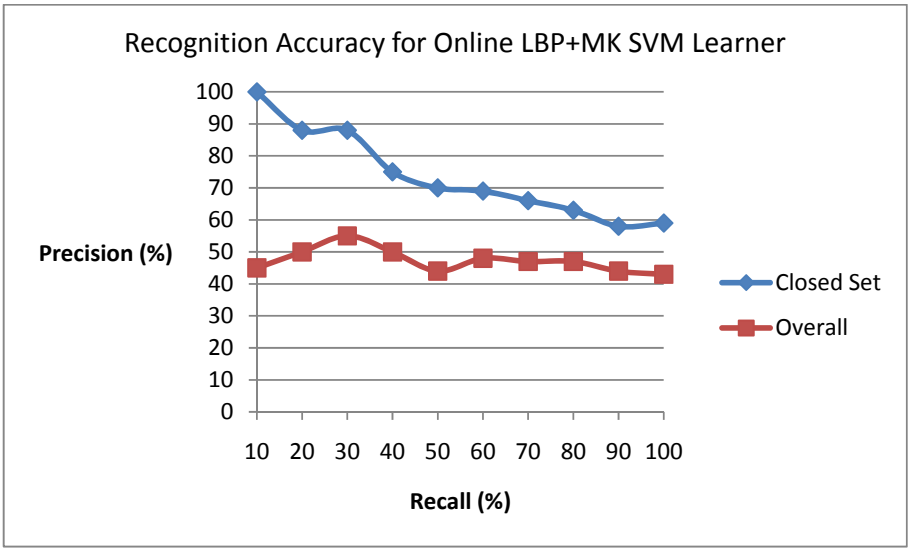

Figure 4.16. Precision vs. Recall Graph for Online LBP+MK SVM learner.

behavior ceases at some points. Wrong face classifications may still occur with high similarity scores, resulting in a fall of precisions at low recall values.

For the "overall classification" precisions, Nearest Neighbourhood methods (See Figures 4.5-4.7) outperform SVM (See Figures 4.11-4.16) and LDA (See Figures 4.8-4.10) methods. This result shows that the target face samples and intruder samples cannot be easily discriminated as the intruder samples get scattered around the feature space. This entanglement of intruder and target samples makes it difficult to fit a model for the feature space.

For the "closed set identification" precisions, SVM methods perform the best (See Figures 4.11-4.16). Discarding the intruder samples boosts up the precision values considerably for SVM methods. As it is stated before, this result is expected since SVM is a discriminative model and instead of generating a model for each class, the aim is to accomplish linear classification in terms of a separating hyperplane which considers the

target class and its competitors. Thus an intruder sample is not taken into account for the generation of the model. Hence rejection of intruders may not be performed optimally. This results in a major decrease in precisions when application scenario is switched from "closed set identification" to "overall identification". (See Figures 4.11-4.16)

Precision scores obtained from Multiple Kernel SVM methods (See Figures 4.14 - 4.16) are comparable to those obtained from single Kernel SVM methods (See Figures 4.11-4.13). However the burden of the extra computations that are necessary to train the Multiple Kernel SVM models does not provide comparable improvement in precisions hence using an SVM method is favorable over the use of Multiple Kernel SVM for our online learning application. Using Multiple Kernel Learning algorithms may be preferred in order to select coefficients for the base kernels, instead of assigning equal weights for each.

Among all features used, HOG features are the most successful in describing the discriminative facial features although its extraction is computationally heavier than the others. Also the number of feature vector dimensions is higher than the others. Greater dimensionality may cause degradation in classification performance for the Nearest Neighbourhood classifier, which is a phenomenon known as "the curse of dimensionality", where the information in the feature vector is dispersed in so many dimensions. During nearest neighbor computations, irrelevant features in the many dimensions may suppress and overshadow the relevant features. But our test results show that 1260 dimensioned feature vectors of HOG features works robustly for Nearest Neighbour classification (See Figure 4.5). In Linear Discriminant Analysis, construction

times of scatter matrices (See Equations 3.4 and 3.5) increase with the number of dimensions in the feature vector. Hence higher the number of feature dimensions, longer it takes to construct the eigenspace model. For the SVM methods, training times are increased with the increased number of feature vector dimensions but test results show that HOG features compensate for the longer training times with high precision values. (See Figures 4.11 and 4.14)

Although the face track covers the eyes and mouth regions, due to the variation of poses, i.e., rotation of the head, still the aspect ratio of the face may change due to out-of-plane rotations, and when the image is resized to a standard size for processing, corresponding

DCT blocks of two different images may actually cover slightly or totally different facial regions. On the other hand this is expected to be as less of a problem for LBP since LBP is invariant to monotonic changes of gray levels as such in the case of resizing the image. HOG features use overlapping blocks to describe face which increases robustness against localization errors of facial features.

For the “closed set identification” scenario HOG features used with single or multiple kernel SVM has the best precision/recall graphs (See Figures 4.11 and 4.14). But when the scenario is switched to “overall identification” where the annotation tool is used to automatically classify all encountered faces, the best precision/recall graphs are obtained for HOG features used along with Nearest Neighbourhood classifier (See Figures 4.5).

Domination of crowded class samples is a problem with the nearest neighbourhood method but in our application we have limited the number of samples/class to a fixed value in order to deal with this problem. The threshold is selected as 500 samples/class. Nearest neighbor classification is a fast method compared to the other methods. Although the database grows larger and larger with addition of each new track, the query face track has constant length (selected as 10 samples), hence classification times increase linearly only with the number of training samples. Nearest Neighbourhood with HOG features performs the best overall score for our online face identification system (See Figure 4.5). It is observed that fitting discriminant functions for target classes in our “overall identification” application cannot give adequate results (See Figures 4.8-4.16).

If we focus our attention only on the “closed set identification”, discriminant models like SVM achieves high precision values (See Figures 4.11-4.16). HOG features with Multiple Kernel SVM perform the best (See Figures 4.14). Single Kernel SVM with HOG features follows its Multiple Kernel counterpart with comparable precision scores (See Figure 4.11).

4.4.3 Comparison of the Online and Offline Methods

In order to compare the recognition accuracies for online and offline methods, offline “precision vs. recall” graphs have also been plotted. In the online learning scenario, classification with HOG features provided the best results. Therefore we have compared the offline HOG+SVM, HOG+LDA and HOG+NN methods with their online counterparts via the Precision vs. Recall metric.

For the offline methods, first 200 shot boundary scenes of the same “How I Met Your Mother” episode are used. For the NN and LDA sessions, 400 samples/class have been accumulated for 6 classes. After that point no more training is done and the testing is performed on the rest of the scenes. For the offline SVM method, 250 samples/class have been accumulated for 6 classes and similarly after a single training session, the rest of the scenes are used for testing alone.

For the SVM learning, switching from the offline learner (See Figure 4.17) to the online version (See Figure 4.11) results in a decrease in the “overall identification” scenario whereas the performance in the “closed set identification” scenario stays similar. The decrease in the precision scores are expected as the precision scores of the online method includes the classification results in the early stages of the learning session whereas in the offline session, the model is trained once when all classes have a higher number of representative samples. In the offline training, the best decision hyperplane is determined according to a large surplus of training data. However in the online training, the decision hyperplane is constructed using partial information hence the recognition precision is expected to be lower than the offline version. Approximately 30 % of the training samples are returned as the support vectors from the trained SVM model. When the number of training samples is large, sequential SVM training provides considerably shorter training times. With addition of each new face track which is typically 20-200 samples long, retraining thousands of samples would take as much as three times longer training time compared to the sequential SVM training with the expense of a fall in the recognition precision.

For the LDA and NN learning, online and offline learners provide comparable precision scores (See Figures 4.5, 4.8, 4.18 and 4.19). But it should be noted that the similar precision scores have been recorded over a much larger set of query tracks for the online version, as the recognition starts at the initial stages of the training session whereas in the offline stage, only training is performed until the number of training samples exceeds a value. In our tests approximately 150 scenes are used to collect training data for the LDA and NN methods and the remaining 50 scenes are used for testing. On the other hand for the online learning session classification hence the precision calculations are initiated whenever two distinct classes receive samples for the NN case and whenever the total number of samples exceeds the number of feature vector dimensions for the LDA case.

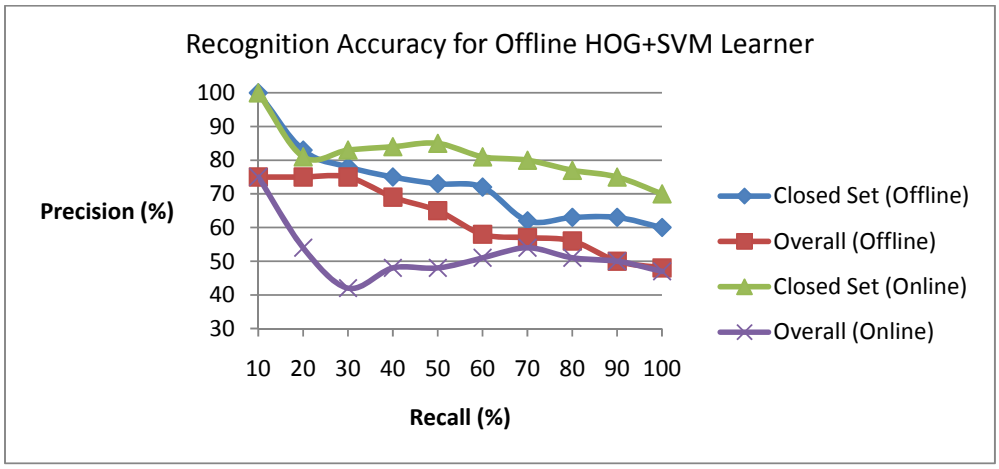

Figure 4.17. Precision vs. Recall Graph for Online/Offline HOG+SVM learners.

Online training is a useful tool for updating the model with new representative data or even adding totally new classes to the scheme. A model trained over a training surplus i.e., an episode of a TV series, may be easily extended to be used over a wider extent i.e., another episode of the TV series by incorporating the new information contained in the new environment.

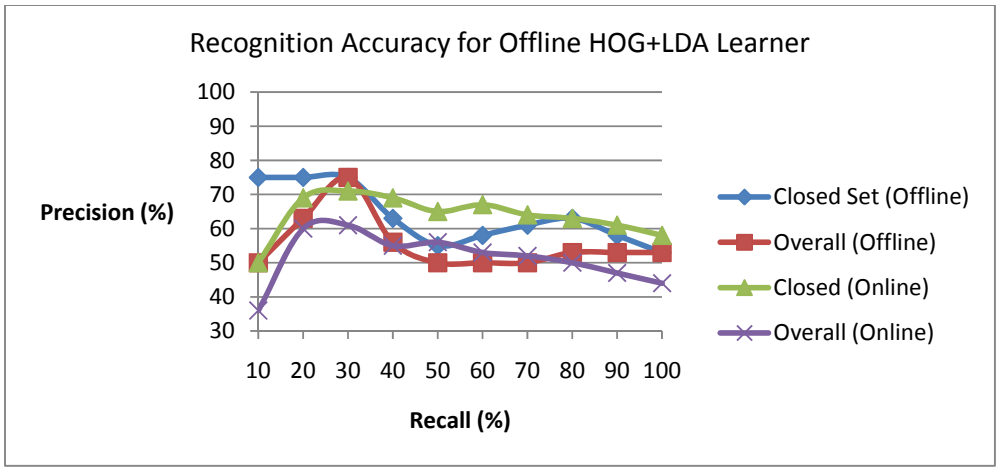

Figure 4.18. Precision vs. Recall Graph for Online/Offline HOG+LDA learners.

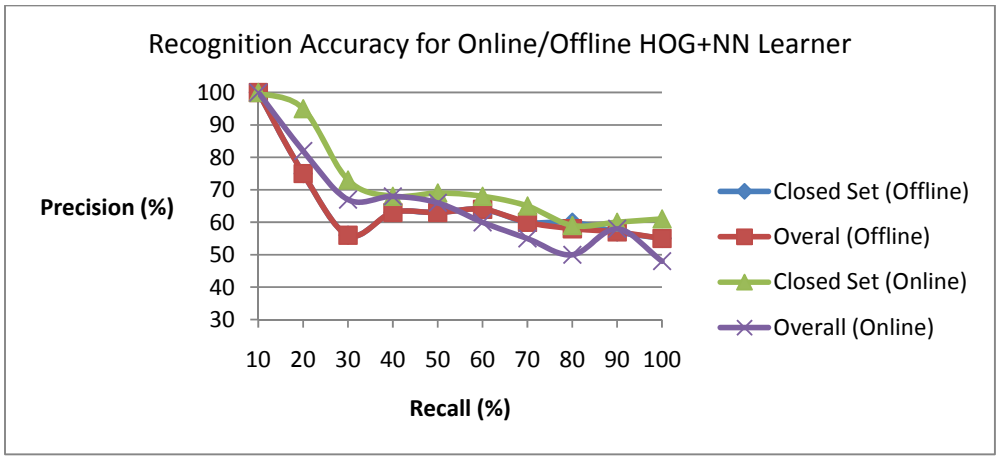

Figure 4.19. Precision vs. Recall Graph for Online/Offline HOG+NN learners.

In order to evaluate the online learning performance in terms of precision and execution times, a final test is performed. Precision scores and training times are plotted along the learning session. We have evaluated the HOG + SVM learner for this test. Video is partitioned into intervals and at each time step, the data collected so far is used to perform training for the offline version. For the online version, the training is performed sequentially using the trained SVM model along with the training data collected in the previous interval. Testing is performed using all of the tracks in the rest of the video.

For example, when percentage of the video is used for training is 10 %, all face tracks which reside in the first 10 % of the video is used for training a model for both online and offline versions and the training times are recorded. Then the precision scores are found by testing all of the face tracks which reside in the remaining part of the video. Intervals which correspond to sections with 10 % of the video are considered. We collecting face tracks up to the point where we have considered first 20 % of the video, the training is performed again. For the online version, the data collected between 10 % and 20 % of the video is trained sequentially using the previously trained SVM model. But for the offline version, all of the data in the first 20 % of the video is used to retrain the SVM model from scratch.

Online and Offline precision scores are comparable according to Figure 4.20. For the online case training is done sequentially at each time step, the training times have a constant characteristic (See Figure 4.21). On the other hand for the offline learner, all of the previously collected training data has to be learnt which results in linearly increasing training times as the user advances through the time steps.

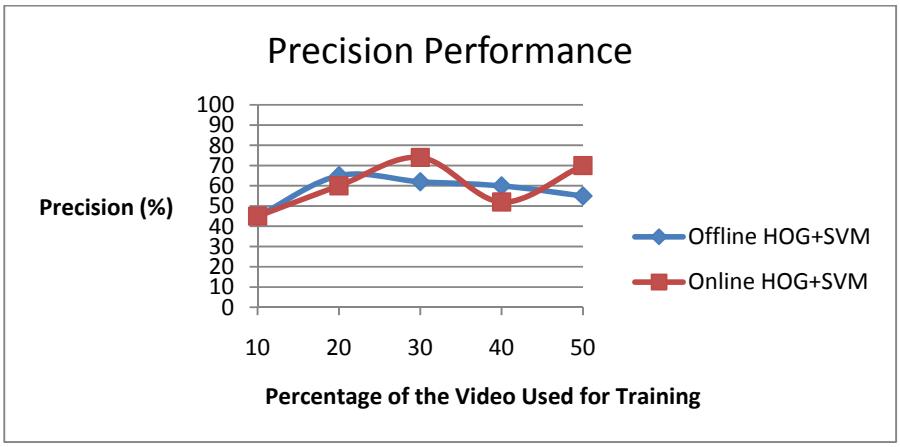

Figure 4.20. Precision Performance for Online and Offline HOG + SVM learners.

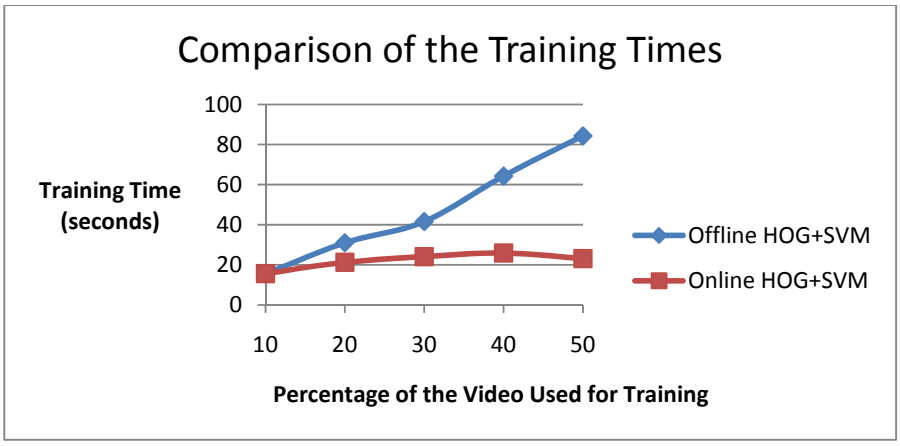

Figure 4.21. Training times for the SVM models.

CHAPTER 5

CONCLUSION

In this thesis, an online face identification system is developed and suitable face recognition algorithms have been evaluated. Extensive tests have been done in order to compare different facial feature extraction and classification algorithms in terms of their recognition accuracies and execution times. Execution times are also crucial since our aim is to design an online learning system where constructed models are updated with each addition of new data instead of reconstructing the models over and over. Recognition accuracies of the considered algorithms have been evaluated by implementing an interface for the face identification system. Here videos are partitioned into meaningful segments using a shot boundary detection algorithm. Face detection and tracking algorithms have been implemented to collect tracks of face images. We have chosen Viola-Jones face detector [1] which is a very popular, fast and robust face detector with Lucas-Kanade Optical Flow feature tracker [21] as the best performing face tracker for our case. The success of our system depends on not only the overall tracking of the "face region" but also keeping the eyes and mouth well-located within the face while rejecting the background clutter as much as possible. That's why a facial feature tracker is used. The face recognition under a variety of environmental conditions is already a challenging task so we have limited our attention to near-frontal poses. Hence our face tracker is a facial feature tracker which returns near frontal faces. But this limited our training and testing samples as faces are encountered throughout videos under a variety of poses like profile faces. In our method there is no heavy post-processing steps in order to improve the quality of the extracted face tracks, since computationally expensive efforts prohibits a real time application scenario.

Local Binary Patterns, which is originally a texture descriptor, shows a good recognition precision for the human face recognition process where faces are not constrained to be

controlled samples. Local Binary Patterns is a good alternative to Discrete Cosine Transform Features being invariant to resizing of images for processing, having nearly half the size of feature dimensions which decreases the computational cost and reduces the amount of memory necessary to store the feature vectors for the Nearest Neighborhood classification. But the test results show that Histogram of Oriented Gradients is the best performing facial feature extraction method although the high dimensionality of the feature vectors poses computational difficulties such as longer training times and higher memory requirements.

According to our test results, Nearest Neighborhood classification method has a higher recognition rate than the other two learning algorithms. This is due to the fact that, recognition is performed for a collection of poses with high diversity which makes it difficult to fit a model for the distribution of the feature vectors. Nearest Neighborhood on the other hand does not fit a model for the data, hence different poses, different facial expressions and other visual challenges in the database do not yield as much a problem as they do for the case of LDA and SVM. The best performing combination includes the Histogram of Oriented Gradients used with Nearest Neighborhood for the overall face identification scenario. Nearest Neighbourhood is also a handy classification method as new data can be added to the training surplus directly, whereas for SVM methods, sequential training causes some decrease in the recognition precisions due to the splitting of the training data during different training phases. Also the training and testing times for Nearest Neighbourhood method are extremely short. If the number of training samples per class is kept balanced, domination of crowded class vectors no longer pose a threat to the recognition.

The results are expected to show much higher precision values if precision is calculated over a closed set for identification by considering only the suggested labels for the target faces. But counting all face tracks, which include intruders such as other faces or false face detections, reduces the overall accuracy. High precision values are expected for low recall values and vice versa but it has been shown that intruders may also receive relatively high similarity measures and intruder samples may not be eliminated by simply setting a threshold to reduce recall.

Possible extensions for this work may include the improvement of the face tracker. Kalman filtering [24] may be used in order to improve the estimation of the future facial

feature positions. Another improvement can be the handling of non-frontal faces. A profile or half profile face detector may be used and collected face images may be clustered according to the pose of the face images. For the multiple Kernel SVM methods, Multiple Kernel Learning [41] algorithm can be applied in order to select coefficients for the base kernels. Selecting equal weights for each base kernel does not provide satisfactory improvements compared to the single kernel SVM, considering the extra computational burden involved.

REFERENCES

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

[2] L. Wolf, T. Hassner, Y. Taigman,"Descriptor Based Methods in the Wild", Real Life Images Workshop at the European Conference on Computer Vision (ECCV), October 2008.

[3] H.K.Ekenel, R. Stiefelhagen "Local Appearance Based Face Recognition Using Discrete Cosine Transform",in Proceedings of the 13th European Signal Processing Conference (EUSIPCO 2005), Sep. 2005.

[4] T. Li, S. Zhu, M. Ogihara, "Using Discriminant Analysis for Multi-class Classification", Proceedings of the Third IEEE International Conference on Data Mining, pp. 589- 592, 2003.

[5] Ognen Arandjelovic, Andrew Zisserman, "Automatic Face Recognition for Film Character Retrieval in Feature-Length Films", IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) vol. 1, pp.860-867, 2005.

[6] Bradski, G.R., "Computer Vision Face Tracking for Use in a perceptual user interface", Intel Technology Journal, 2nd Quarter, 1998.

[7] Cheng Y., "Mean Shift, Mode Seeking and Clustering.", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, pp. 790-799, 1995.

[8] J. Sivic, M.Everingham, A. Zisserman, "Person Spotting: video shot retrieval for face sets", in Proceedings of the 4th Conference on Image and Video Retrieval, pp. 226-236, 2005.

[9] M. Fischer, "Automatic identification of persons in TV series" Universität Karlsruhe (TH) M.S. Thesis, 2008.

[10] R. Lienhart, “Comparison of Automatic Shot Boundary Detection Algorithms”, Proc. SPIE, vol. 3656, pp. 290-301, 1998.

[11] C.C. Chang, C.J. Lin, “ LIBSVM : a library for support vector machines”, http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.

[12] C. Domeniconi, D.Gunopulos, “Incremental Support Vector Machine Construction”, Proceedings IEEE International Conference on Data Mining, pp. 589–592, 2001.

[13] L.Chen, B. Hu, “Face annotation for family photo album management”, International Journal of Image and Graphics, vol. 3, No. 1, pp 1-14, 2003.

[14] B. Chupeau, V. Tollu, J. Stauder, I. G. Thomson, “Human face detection for automatic classification and annotation of personal photo collections”, Proceedings of the SPIE Visual Communications and Image Processing, vol. 5150, pp.1848-1856, 2003.

[15] R. M. Jiang, A. H. Sadka, H. Zhou, “Automatic human face detection for content based image annotation”, International Workshop on Content-Based Multimedia Indexing, CBMI 2008, pp. 66-69, 2008.

[16] N. Poh, C-H Chan, J. Kittler, “Face video Competition at ICB2009”, Int’l Conf. on Biometrics (ICB), 2009.

[17] W. Ma, H. J. Zhang, “An indexing and browsing system for home video”, European Signal Processing Conference No.10, pp. 131-134, 2000.

[18] S. Satoh, “Comparative Evaluation of Face Sequence Matching for Content-based Video Access”, Fourth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 163-168, 2000.

[19] J. Stallkamp, H. K. Ekenel, “Video-based Face Recognition on Real-World Data”, IEEE 11th International Conference on Computer Vision, ICCV 2007, pp.1-8, 2007.

[20] J. Shi, C. Tomasi, “Good features to track", Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn., pp 593-600, 1994.

[21] J. Bouguet, “Pyramidal implementation of the Lucas-Kanade feature tracker: description of the algorithm”. Technical report, OpenCV Document, Intel Microprocessor Research Labs, 2000.

[22] M. Turk, A. Pentland, “Eigenfaces for Recognition”. J. Cogn. Neurosci. vol. 3 pp. 72–86, 1991.

[23] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.19, no.7, pp.711-720, 1997.

[24] C. Fagiani, M. Betke, J. Gips, “Evaluation of Tracking Methods for Human-Computer Interaction”, Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision, pp. 121, 2002.

[25] K. Nummiaro, E. Koller-Meier, L. Van Gool, “An adaptive color-based particle filter”, Image and Vision Computing, vol. 21, no.1, pp. 99-110, 2003.

[26] K. Jonsson, J. Kittler, Y.P. Li, J. Matas, "Learning Support Vectors for Face Verification and Recognition,", Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), pp.208- 213, 2000.

[27] E. Aşan, “Video Shot Boundary Detection by Graph Theoretic Approaches”,M.S.Thesis, Electrical and Electronics Engineering, Middle East Technical University, September 2008.

[28] L. Shen, L. Bai, “A review on Gabor wavelets for face recognition”, Pattern Analysis & Applications, Springer London, , vol. 9, no. 2-3, pp. 273-292, October 2006.

[29] F. Jun, N. Dimitrova, V. Philomin, “Online face recognition system for videos based on modified probabilistic neural networks”, International Conference on Image Processing, vol. 3, pp. 2019- 2022, 2004.

[30] MATLAB Central, File Exchange. http://www.mathworks.com/matlabcentral/fileexchange/17960-particle-filter-color_tracker. 5 May 2010.

[31] A. Kudhinsky, C. Pering, M. L. Creech, D. Freeze, B. Serra, J. Gvvizdka, “FofoFile: A Consumer Multimedia Organization and Retrieval System”, Conference on Human Factors in Computing Systems, pp. 496-503, 1999.

[32] W. Zhao, R. Chellappa, P. J. Phillips, A. Rosenfeld, “Face recognition: A literature survey”, ACM Computing Surveys, vol. 35, issue 4, pp. 399–458, December 2003.

[33] H. A. Rowley, S. Baluja, and T. Kanade. “Neural networkbased face detection”. IEEE Trans. on PAMI, vol.20, issue 1, pp.23–38, 1998.

[34] J. Zhu, S. C. H. Hoi, M. R. Lyu, “Face Annotation Using Transductive Kernel Fisher Discriminant”, IEEE Transactions on Multimedia, pp. 86-96, 2008.

[35] B. Raytchev, H. Murase, “Unsupervised face recognition by associative chaining”, Pattern Recognition 36, pp. 245 – 257, 2003.

[36] C. Czirjek, N. O’Connor, S. Marlow and N. Murphy, “Face Detection and Clustering for Video Indexing Applications”, Advanced Concepts for Intelligent Vision Systems, pp. 2-5, 2003.

[37] Y. Tian, W. Liu, R. Xiao, F. Wen, X. Tang, “A Face Annotation Framework with Partial Clustering and Interactive Labeling”, IEEE Conference on Computer Vision and Pattern Recognition, CVPR '07, pp. 1-8, 2007.

[38] M. Everingham, J. Sivic and A. Zisserman “"Hello! My name is... Buffy" – Automatic Naming of Characters in TV Video”, BMVC 2006.

[39] D. Ramanan, S. Baker, S. Kakade, “Leveraging archival video for building face datasets”, IEEE 11th International Conference on ICCV 2007, pp.1-8, 2007.

[40] N. A. Syed, H. Liu, K. K. Sung, “Incremental Learning with Support Vector Machines”, International Joint Conference on Artificial Intelligence (IJCAI), 1999.

[41] M. Varma and D. Ray. “Learning the discriminative power invariance trade-off”, IEEE 11th International Conference on ICCV 2007, pp.1-8, 2007.

[42] N. Dalal and B. Triggs. “Histogram of Oriented Gradients for human detection”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886-893, 2005.

[43] M. Everingham , J. Sivic, A. Zisserman, “Taking the bite out of automated naming of characters in TV video”, Image and Vision Computing, vol. 27, issue 5, pp 545-559, April 2009.

[44] J. Sivic, M. Everingham, A. Zisserman, "“Who are you?” – Learning person specific classifiers from video", IEEE Conference on Computer Vision and Pattern Recognition, pp. 1145-1152, 2009.

[45] R. Gonzalez, R. Woods, “Digital Image Processing”, Prentice Hall, Second ed., 2002.