DETECTING AND TRACKING HUMAN FACES IN COMPRESSED DOMAIN FOR CONTENT BASED VIDEO INDEXING

WANG JUN

NATIONAL UNIVERSITY OF SINGAPORE

2002

DETECTING AND TRACKING HUMAN FACES IN COMPRESSED DOMAIN FOR CONTENT BASED VIDEO INDEXING

WANG JUN

(B.E. (Electrical Engg.), SEU China)

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2002

ACKNOWLEDGEMENT

I have been at NUS for almost two years. Initially, I had several difficulties to deal with: getting familiar with a new school and a new country, means of doing my research here, etc.. Fortunately, my family, friends and colleagues made things easy.

I have to thank my advisor, Mohan S Kankanhalli for leading me to the area of multimedia, and computer vision research. He taught me not only the knowledge of science, but also how to do a good research. His focused attitude towards research and work influenced me always. Thanks to my "the second boss" Philippe Mulhem for his encouragement, valuable advice and his humor, our Friday meetings went easy.

Thanks especially to R.K, my collaborator and coauthor in most of the work presented in Chapter 4, for teaching how to write more formal English and for introducing me to delicious Indian foods. Thanks to Hadi and Weiqi for constant advice and guidance. Thanks to Yunlong for his code which was my start point and for his valuable suggestions. Thanks for Lixin Fan for the conversations we had that give me good insights into the work in my hands. Thanks all friends and other group members: Qiuying, Ji yi, Jek, Stephane, Chen lei, Chitra , Yi Lan, Peng Bin and others…

And finally, thanks to my parents, my brother and my girlfriend. Their constant love, encouragement and support never let me feel lonely.

SUMMARY

With recent advances in broadband networks, image/video compression standards (JPEG/MPEG) and consumer electronics (including amateur digital video cameras), video data now ranges from simple home videos to sophisticated movie productions. The huge amounts of video data generated everyday make it imperative to index the data in a way that enables fast content-based search and retrieval. This has resulted in active research into developing efficient video indexing techniques. Visual features resulting from recorded human activities are more likely to be of value for content description and indexing in the video sequences (news, movies, home video, and etc.), than anything else. So human faces often constitute the most important content in the video sequences. Therefore to be able to recognize faces and index them would be a crucial feature of any indexing system. As the first step to face identification, which is a part of the DIVA (Digital Image and Video Album, http://diva.comp.nus.edu.sg:8080) project, the work presented in this thesis aims to find human faces in video for the purpose of the content based indexing.

In order to use all the information which video brings us, we proposed two-step method to find face sequences in compressed domain. There are three components (U,V, and Y DCT coefficients) in the MPEG compressed I frames video. As the first step to utilizing spatial information completely, two things are done. First, a skin color filter, based on compressed chrominance information, is used to find out potential skin regions. Second,

a 4x4 block-sized window is used to scan the skin regions in the MPEG I frame to extract Y-DCT features. A neural network is trained using those DCT features to classify the face pattern. Since DCT coefficients have compact information about the I frame, using DCT coefficients as features can reduce the complexity of the neural network, and in addition increase the processing speed. Compared to other face detection approaches in the compressed domain, our algorithm achieves comparable performance.

In the second step, by using temporal correlation information presented in video, we proposed a sensor fusion approach to fuse the outputs of a face detector and two object trackers (developed by another member of the DIVA group). We consider the face detector and object trackers to be noisy sensors. Kalman filter is used in the sensor fusion architectures. The sensor fusion approach is adopted in order to build a system that is more accurate than any of the individual sensors. Four sensor fusion architectures have been tried out to compare the effectiveness of fusion frameworks in tracking. The tracking results show that the fused results are optimal and better than those of any individual sensor as well as their mean. Also the hierarchical two-stage state vector fuser appears to be the best fusion architecture.

The main contributions of this thesis are as follows. In the face detection part, a compressed domain neural network based face detection approach is proposed by directly using DCT coefficients as features. In the face tracking part, a hierarchical sensor fusion framework based on measurement fusion and state vector fusion of Kalman filers is built to fuse three individual sensors coming from two object trackers and one face detector.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

With recent advances in very large scale integration (VLSI) technology and digital signal processing technologies (image/video compression standards), audiovisual data ranging from TV programs, simple home videos to movies are being stored and transmitted in the digital format. Due to the developments in digital broadcasting technologies, broadband networks, and especially digital consumer electronics (including digital video cameras), these digital audiovisual data have been influencing most families all over the world. Because of the digital format, random access and even non-linear editing of video has become practical. However manual indexing of the huge amounts of digital audiovisual data generated everyday is time consuming. The alternative, feature based indexing may not give a satisfactory retrieval result. It is therefore imperative to automatically index the visual data based contents that enables fast content-based search and retrieval. This has resulted in active research into developing efficient content based video indexing techniques.

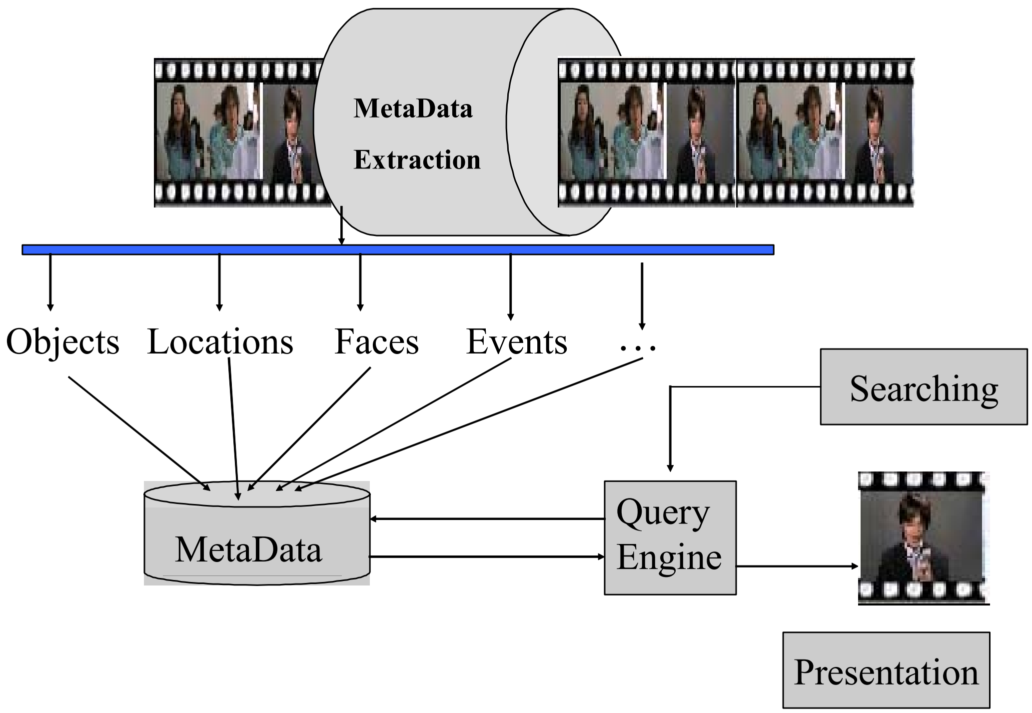


Fig 1.1 Digital Video Album System Diagram
The system has two components: the indexing and the retrieval components. In the indexing component, contents in the video such as objects, human faces and even events are indexed by finding the relations between the low level features (color, shape, motion vector, and texture) and the semantic features. In the retrieval component, the query engine manages the interaction with the users, giving users the more semantic query results based on the metadata created by the indexing component.

Ever since the introduction of Sony's DV format in 1995 and the subsequent adoption of the IEEE 1394 interface almost universally has made the digital camcorder almost as easy to attach to a PC system as a mouse. Suddenly enthusiasts have accessed to a technology that allows them to produce source material in a digital format whose quality far exceeds that of the analog consumer formats available and turn it into professional-looking home videos at their desktop PC. This has resulted in home videos to become one of the most importance video resources in people's lives. For efficient management of the massive amounts digital home video data, the DVA (Digital Video Album) project, part of DIVA (Digital Image and Video Album, http://diva.comp.nus.edu.sg:8080) project, is aiming to develop techniques for content -based indexing and retrieval of digital video. As shown in figure 1.1, the DVA System mainly has two components: the indexing and the retrieval components. In the indexing component, content in the video such as objects, human faces and even events are indexed by finding the relations between the low level features (color, shape, motion vector, and texture) and the semantic features. The retrieval component manages the interaction with the users, giving users the more semantic query results based on the metadata created by the indexing component.

In this chapter, the problem of finding human faces in videos and its importance to the content based video indexing are discussed in section 1.1. In section 1.2 and 1.3, the methods for finding human faces in videos are presented. Section 1.4 gives the outline of the thesis.

1.1 Finding human faces in videos and its importance

In most videos, visual features resulting from recorded human activities are more likely to be of value for content description and indexing in the video sequences (news, movies, home video, and etc.) than anything else. Especially in home videos, the main activities in the video concern family's lives such as birthday party, wedding ceremony, travel and etc. and in news video, the main activities in the video are about the events and people. Most possible questions users would ask for these videos may be as following:

- Find the video showing my daughter *Lisa* in the swimming pool.
- Present the news clips that show S*addam Hussein* speaking.
- Summarize the video clips to show only the news anchor

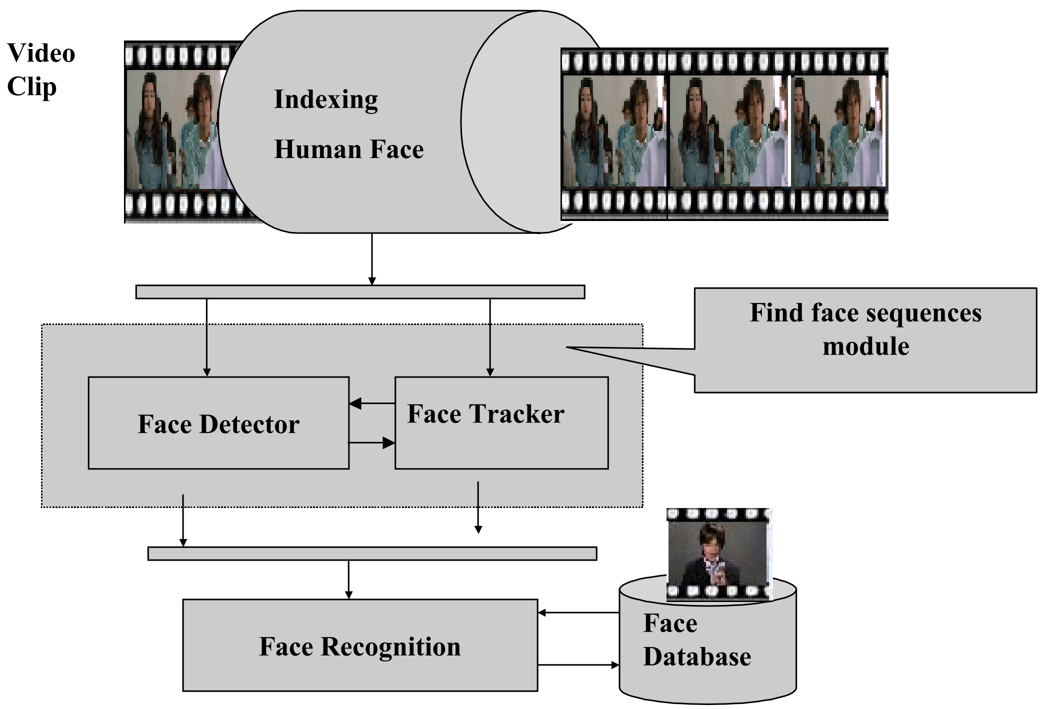


Figure 1.2 Face identification system for videos
Modules, including face detection in image, face tracker, and face recognition, are needed for this purpose.

To be able to answer and implement above functionalities (content based video presentation, content based video query, and content based video summarization), the location and recognition of the humans in video would be critical. Human faces therefore constitute the most important content in the video sequences. If we can automatically or at least semi-automatically recognize and locate faces in videos, above human activity driven functionalities can be easily implemented.

However letting the computer automatically locate and recognize faces in a video, the so-called face identification problem, is not an easy task at all. Normally many modules (as shown in Figure 1.2) including face detection in image, face tracker, and face recognition are needed for this purpose. The definition of these modules is shown below:

1. Face Detection: Location of a face(s) in a complex scene (frame)

2. Face Tracker: Continuous location of the face(s) in an image sequence.

3. Face Recognition: Comparison of the face to a database of known faces and distinguishing individuals located by face detector or face tracker.

Depending upon the application, the methods of these modules may not be same due to the different domain knowledge presented and different requirements. Currently, most of face trackers are designed for the purposes of human computer interaction (HCI) and videoconferences. They therefore have the assumption that only one face in the fixed scene and real time (online) processing is quite often required. However, in our specific domain (indexing of home videos), that assumption is violated. Usually, a multi-face tracker is required but real time (online) processing is not necessarily needed (however efficiency of algorithm still needs to be considered), Therefore the methods (background segmentation methods and face location methods and etc.) used in the field of HCI are not suitable and the results are not robust enough (requirement of real time processing often leads to weak face models) for the purpose of video indexing.

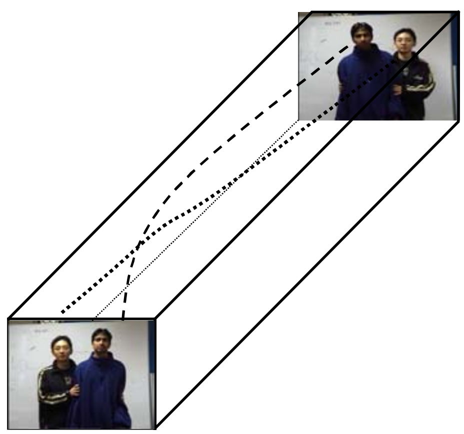


Figure 1.3 Face sequences in videos
Two human faces exist in the video. The movements of the two faces are indicated by two trajectories using dotted lines.

In order to clearly state our specific problems in our specific domain, we give the definition of the face sequences beforehand:

- *face sequences:* location of a face(s) in a video along three dimensions (two spatial positions and one temporal position) within one shot (as shown in Figure 1.3).

So according to this definition, finding face sequences in video includes two parts: focusing on establishing the face model in the spatial dimensions (frame) and detecting a face(s) in a frame leads to face detection, and focusing on temporal correlation leads to face tracking. By this definition, we can see that face detection and face tracker really model the face dynamics in the three dimensions of the video.

Recognizing face sequences and consequently indexing them is a crucial feature in the indexing part of the DVA project. As a prerequisite for face identification, however, face sequences (three dimensions of face position) have to be found. Being part of the DVA project, the work present here is mainly to find face sequences (it means location of a face(s)) in the video as the initial point of human activity indexing. From here, we can see that face detection and face tracker cannot really be separated.

The challenges associated with this research (finding face sequences in home video) come from two aspects (face itself and video domain) as shown as follows:

- Pose and facial expression variation: Face can appear in different poses (frontal, profile, 45 degree, and even upside down and rotated), and facial expressions cannot be same always. These make appearances of faces vary drastically.

- Illumination variation: The brightness and location of the light source in particular can radically change the appearance of a face. This dramatically increases the variation within a face class.

- Presence or absence of structural components: Facial features such as beards, mustaches, and glasses may or may not not be present and there is a great deal of variability among these components including shape, color, and size.

- Occlusion: Other faces or objects may occlude faces. This makes facial features lost in some frames.

- Amateur video: Home videos are shot by amateurs. Loss of focus and shaking may exist.

- Low quality: compared to images and broadcasting video, home video has a lower pixture quality.

- Huge amounts of data in video need to be processed: video data is large even in the compressed format (MPEG1/2/4).

From the above, we can see that video data poses tremendous challenges to this research. However, temporal correlation from video also gives us more information about the faces. In order to utilize all the information which video contains, we propose a two-step method to find face sequences in the compressed domain. In first step, by fully using spatial information in video, we develop a neural network based face detector using DCT coefficients. In the second step, by utilizing temporal correlation information in video, we propose a sensor fusion approach to connect the face detector and the two object trackers (developed by another member of the DIVA group). Note that our algorithms are completed implemented in compressed domain for the purpose of efficiency, motivated by the fact that lot of video data is stored in the compressed format.

The following two sections will describe these two steps and show how each of above challenges can be addressed.

1.2 Face detection in the I frame

We use a neural network as classifier to classify patterns into faces and non-faces using DCT coefficients as features in compressed domain.

In the face detector a statistical skin region filter is initially used to filter out skin color regions in I frames. The neural network based face detector classifies these regions as faces and non-faces using the luminance DCT information of the output of the skin region filter.

The neural network approach in compressed domain is as follows: for a given MPEG video, each candidate skin color region in I-frame is scanned in steps of two pixels by a sliding window of 4×4 blocks passing through it. From that sliding window, DCT coefficient features of Y components are extracted (The bottom-left and bottom-right blocks are ignored, since the information in these two blocks usually comes from the background.) These features are put into one neural network (126 input units, 20 hidden units, and 1 output unit) to classify whether the pattern inside the sliding window is face pattern or not after normalization. To detect faces larger than the size of this sliding window, the window is passed over successively on images downscaled by a fixed factor. More than one thousand face images have been used for training purpose. These images include different facial expressions and facial features such as beard, mustache or glasses as well. It can therefore help to detect faces with different facial expressions and features. Some of DCT coefficients (most of whose variations come from background) are removed when training and detecting. It could help to detect face in different illuminated scene.

1.3 Sensor fusion approach to fuse face detector and object tracker

To tackle finding face sequences in video, face detection methods are often limited by their inability to detect faces in various poses since we found that most of the faces in home video are non-frontal, even though they are temporally related with front-view faces. Also face detectors currently seem to detect faces only for individual frames, without taking in to account their temporal correlation in case of video when tracking of faces needs to be performed. In another aspect, face trackers are often limited by the need to segment the face to be tracked. Besides, in the absence of a good matching mechanism, face trackers may accumulate errors.

Due to the complexity of the videos and variations of the human faces, the face detector and face tracker may not work too well independently. The two can therefore be merged to mutually overcome limitations specific to either of them. In particular, a corrective feedback from good face detection results (for instance) can prevent error accumulation. Conversely, a face detector's performance can be improved with the help of good tracking support. However, finding a method that can combine two components (face detector and object tracker) to obtain a joint estimate is non-trivial. It seems logical to take the sensor fusion approach to fuse the measurements. Kalman filtering provides a good mechanism to fuse noisy measurements from sensors by minimizing the mean square error (MSE). In this thesis, we present a novel sensor fusion based face tracking approach in the compressed domain. A Kalman filter based two-stages fusion framework is proposed to fuse three individual sensors, that come from two object trackers (one color based object track and one motion vector based object tracker) and one face detector (a compressed domain neural network based face detector). Experiments have been done to prove that the proposed method is effective.

1.4 Outline of the thesis

The remainder of the thesis is organized as follows:

Chapter 2 gives literature review about face detection both in pixel domain and compressed domain, and face tracker.

Chapter 3 describes the compressed domain neural network based face detection approach, which is limited to upright, frontal faces. Detailed implementation is given and the approach is evaluated over both image and video test sets.

Chapter 4 presents a Kalman filter based sensor fusion method to fuse the face detector presented in chapter 3, and object tracker together. Four fusion architectures are given and results are compared and evaluated.

Chapter 5 describes some applications in which other researchers in our group have used our research for content based indexing and retrieval system, summarizes the contributions of the thesis and points out directions for future work.

Chapter 2

Literature Review

Human face perception is currently an active research area in the computer vision community. It could be broadly used for image/video indexing, human computer interaction (HCI), surveillances, and robotics. There are many closely related problems in human face perception. Following what is presented in [1], we give the definition of these problems as follows.

- *Face Localization*: to localize the position of a single face in an image; the detection task is simplified by the assumption that an input image contains only one face. Some methods have been proposed based on this assumption and therefore only can be used in constrained conditions such as for HCI or videoconference.

- *Face Tracking:* to locate the position of a face in an image sequence in real time. This deals with the face localization problem in captured video in real time. Only one face with the fixed scene is usually assumed.

- *Face feature detection (face alignment):* to locate the face features such as eyes, nose, nostrils, eyebrow, mouth, lips, ears, etc.

- *Face authentication:* to verify the claim of the identity of an individual in an input image.

- *Facial expression recognition:* to detect the expression (sad, happy, smile, angry, and etc) of a face

- *Face Detection:* to determine whether or not there are any faces in the image, and if present, return the image location and extent of each face.

- *Face Recognition:* to recognize the individuals in an image.

The definition may be slightly different according to the research domain and the application target. For instance, as mentioned in Chapter 1, the emphasis in this thesis is quite different. It requires tracking for multiple faces with changed scenes and does not impose the real time processing requirement (however efficiency of algorithms is still strived for). These differences cause our approaches to be different from the other face tracker described in literatures. A detailed discussion will be presented in the chapter 4.

Among the afore-mentioned problems, detection and tracking of human faces is a prerequisite for face recognition and /or facial expression analysis. Our research in this thesis mainly concerns these two problems as a whole. In this chapter, face detection and face tracker algorithms are reviewed both in the pixel and compressed domain in section 2.1 and 2.2 respectively. Section 2.3 gives the discussion.

2.1 Face detection

There are more than 150 reported approaches to face detection so far [1]. This problem attracts researchers by not only its underlying applications, but also its challenges to the underlying pattern recognition and learning techniques. From the pattern recognition point of view, face detection is a two-class classification problem in which an image region is classified as being a face region or a nonface region. Classes with high dimensional feature space and variations, coming from scale, location, orientation, pose, illumination, facial expression and etc., makes it hard to model the distribution of the face efficiently.

Most of the face detection approaches have been implemented in the pixel domain. However, when face detection is applied in compressed image or video (JPEG/MPEG), some of face detection approaches have been proposed to extract features in compressed image/video due to following two facts:

1. DCT coefficients, as a feature, are attractive for pattern recognition since DCT based compression reduces spatial redundancy and compactly captures the compact information about the patterns;
2. It would be efficient if face detection can be implemented entirely in the compressed domain, without performing the inverse DCT followed by feature extraction, for thousands of compressed videos.

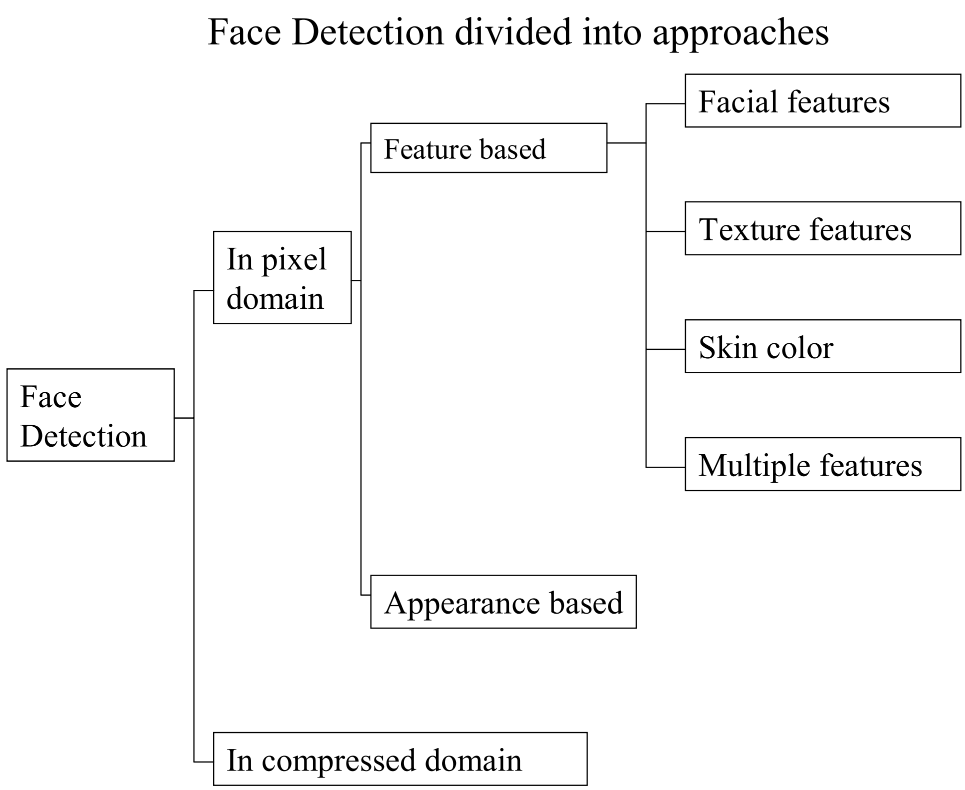


Figure 2.1 The classification of the face detection approach

In a later section, we provide a survey and discussion of the different subareas both in pixel domain and compressed domain shown in Figure 2.1.

2.1.1 Face detection in the pixel domain

Roughly face detection algorithms in pixel domain can be classified into two groups:

1. Feature based approaches: A face pattern is considered to be a set of facial features such as eyes corners, mouth, nose with positions and size within an oval shaped area. The presence of a face in is concluded from the integration of several detection results [2][3][6]. The advantage of these feature-based approaches is that the patterns of the components (eye corners, nose, chin etc.) might vary less under pose changes, orientation and viewpoint changes than the patterns belonging to the face as a whole [4][5]. However, it is hard to choose a set of facial features and model their geometrical configuration in such feature-component-based approaches.

2. Appearance-based approaches: A face can also be considered to be a single pattern and features are extracted from the entire face region. Methods following this philosophy range from Gaussian Mixture distribution model [7], neural networks [8], principal components analysis [9] to SVMs [10]. This second group of algorithms, which treat the face as a single unit, though more complex and slower, has proven to be more effective.

2.1.1.1 Feature based approaches

As mentioned above, it is assumed that facial features such as eyebrows, eyes, nose, mouth, and hair-line are invariant or less variant. Normally these facial features are firstly detected by edge detectors. Based on the detected features, a statistical model or domain knowledge is used to describe their relationships and consequently verify the existence of a face. These features can be categorized into following classes [1]:

2.1.1.1.1 Facial Features

Brunelli and Poggio [6] develop a new algorithm, based on Kanade's first facial feature extraction methods [20] to compute a set of geometrical features (facial features such as nose width and length, mouth position and chin shape). With prior knowledge that face has bilateral symmetry, one can extract and compute facial features by horizontal and vertical edge dominance maps.

Based on the fact that the overall shape of the face or the whole head is very similar to an ellipse, Wang and Tan [3] proposed a method to extract face contour. In this approach, input image is first enhanced by means of histogram equalization, followed by edge detection based on a multiple-scale filter. The extracted edges are then linked using a method based on energy function. The face contour is finally extracted using the direction information of the linked edges. 84.96% detection rate is reported in this paper.

2.1.1.1.2 Texture

Intensity distribution of face can be treated as texture which can be used to separate them from background. Augusteijn [11] used second-order statistical features (SGLD) on subimage of 16x16 pixels to classify face like textures (hair and skin textures). A cascade correlation neural network and a Kohonen self-organizing feature map are used for supervised classification and clustering respectively. Subsequently, Dai and Nakano [12] also used SGLD model for face detection purpose. Color information is also incorporated with texture to improve the performance. The advantage of this approach is that it can detect faces not only upright.

2.1.1.1.3 Skin color

Skin Color is another feature of human faces. Using skin-color as a feature for face detection has several advantages: 1. it is much faster than using other facial features; 2. Under certain lighting conditions, color is orientation invariant so that face pose do not have too much influence. Yang [13] demonstrate that:

- In the color space, skin color is clustered.
- Skin color differences among people mainly rely on intensity instead of chrominance.
- Under a certain lighting condition, a skin color distribution can be characterized by a multivariate normal distribution in the normalized color space.

Usually, a Gaussian density function based skin color model is build for this purpose. The simple way is to build a univariate Gaussian distribution, and its parameters are often estimated using the maximum likelihood [14][15]. Based on the observation that skin color of different ethnic groups more likely form multimodal distribution, a mixture of

Gaussian skin color is built. The parameters in this model are usually estimated using an EM algorithm [16].

However, using skin color for face detection has several problems.1. In different lighting conditions, skin color model may not be adaptive, even though in a certain lighting condition, the skin color model performs well; 2. Different cameras (the attributes of CCD) may not produce the same skin color even for the same person in the same lighting environment.

2.1.1.1.4 Multiple Features

Using above features only for face detection may not have robust results. Recently, many methods combine several facial features to give more reliable results.

Saber [2] proposed an algorithm for detecting human faces and facial features as well, by combining skin color, assuming an elliptical shape of face and symmetry based cost functions. In their approach, first, a supervised pixel-based color classifier is employed to mark all pixels that are within a pre-specified distance of “ skin color”. Skin regions are extracted by Gibbs random field model-based filters. Then an elliptical model is fit to each disjoint skin region. Finally asymmetry-based cost functions search the center of the eyes, tip of nose, and center of mouth within ellipses whose aspect ratio is similar to that of a face.

Fan and Sung presented a method in their paper [19] to combine color, shape, and texture information to detect face and consequently register facial features. In the first stage, a Gaussian skin color model is used to identify possible face locations under varying pose. Secondly, image patterns are compared with a varying pose face model, build by combined feature-texture similarity measure (FTSM) in terms of shape and texture differences. By aligning a prototype face with the unknown pose faces, one can also register the facial features (eyes, nose and mouth). Experiments in video sequences are reported in this paper.

2.1.1.2 Appearance-based approaches

In appearance-based methods, a face is considered to be a single pattern and features are extracted from the entire face region. An image feature (usually intensity) is extracted to construct as random variable X. X is a high dimensional vector in high dimensional feature space. Given the fact that face pattern is clustered in this high dimensional feature space and the probabilistic distribution of face pattern is unknown, assumption of Gaussian distribution or mixed Gaussian distribution leads to Bayesian classification or *Maximum Likelihood* (ML) approaches. Learning face appearances from samples have attracted much attention recently and have proved excellent results. Depending on the different statistic and neural models, appearance-based approaches are divided as follows:

2.1.1.2.1 Eigenface

Eigenface approach has been proposed by Sirovich [22], Turk and Pentland [23] and Toole [24]. Eigenfaces are the first few eigenvectors of the covariance matrix of the set of face images. These principal components of the distribution of faces mainly describe the variations of the face distributions. One can use these principal components to construct lower dimensional space called face space. An image can be classified as face image or nonface image by measuring the distance from the face space.

2.1.1.2.2 Fisherface

The eigenface [23] method uses principal components analysis (PCA) for dimension reduction, yields projection directions that maximize the total scatter across all classes. Compared to eigenface, the fisherface algorithm [35] is based on Fisher's Linear Discriminant and produces well-separated classes in a low-dimensional subspace.

Belhumeur [35] has developed a face recognition algorithm, which is insensitive to large variation in lighting direction and facial expression using Fisher's Linear Discriminant Function (FLD). Each pixel in an image can be considered as a coordinate in a high-dimensional space. They observed that the images of a particular face, under varying illumination but fixed pose, lie in a 3D linear subspace of the high dimensional image space if the face is a Lambertian surface without shadows. However, since faces are not truly Lambertian surfaces and do indeed produce self-shadows, images will deviate from this linear subspace. Rather than explicitly modeling this deviation, they linearly project the image into a subspace in a manner, which discounts those regions of the face with large deviation. Their projection method is based on Fisher's Linear Discriminant and produces well separated classes in a low-dimensional subspace, even under severe variation in lighting and facial expressions. Their extensive experimental results demonstrate that the Fisherface method has error rates that are lower than those of the Eigenface technique for tests on the Harvard and Yale Face Databases (http:// cvc.yale.edu/projects/yalefaces/yalefaces.html).

2.1.1.2.3 Mixture of Gaussian distribution

Sung and Poggio [7] developed a mixture of Gaussian distribution model for detecting face in gray image. Initially, a training set of faces (scaled to 19x19) is collected and normalized using histogram equalization. To eliminate problems due to variability in the background, a mask is applied to each of the patterns. Because of the mask, the dimension of the feature is 283 pixels (shown in figure 2.2).

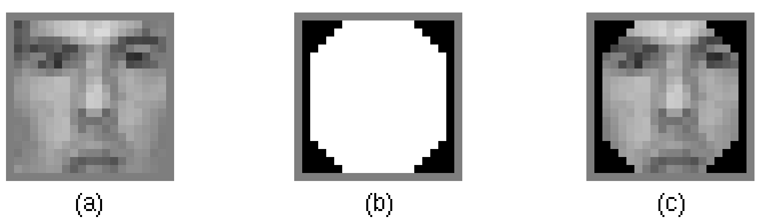


Figure 2.2 Training samples and a mask [7]
(Courtesy of K.-K Sung and T.Poggio)

The training set of faces is modeled as a set of 6 elliptical Gaussian clusters in a high dimensional feature space, using K-mean clustering algorithm.

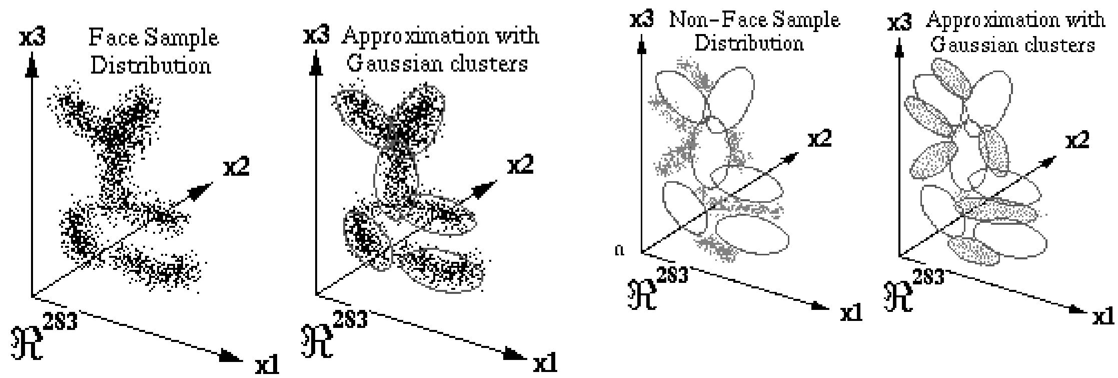


Figure 2.3 Six face and six non face clusters in high dimensional space [7] (Courtesy of K.-K. Sung and T. Poggio)

To refine the boundaries of the clusters, a set of nonfaces that closely resemble faces are used to specify another set of 6 Gaussian clusters. This leads to the combined face/nonface model as shown in figure 2.3.

Sung and Poggio used these two sets of clusters to fully specify what is and what is not a face. To classify a given pattern, they compute the distance from the input pattern to the centroids of each of the clusters. Rather than using a crude distance measure like the Euclidean distance, they use a probability based 2-value Mahalanobis-like distance. The main idea here is that, for each cluster, the first value measures the distance within a lower dimensional subspace of the 75 (approx.) largest eigenvectors; this distance captures the large variations in the patterns. The second value is the Euclidean distance from the pattern to its projection on the lower-dimensional subspace; this accounts for pattern differences in the smaller eigenvectors (Figure 2.4).

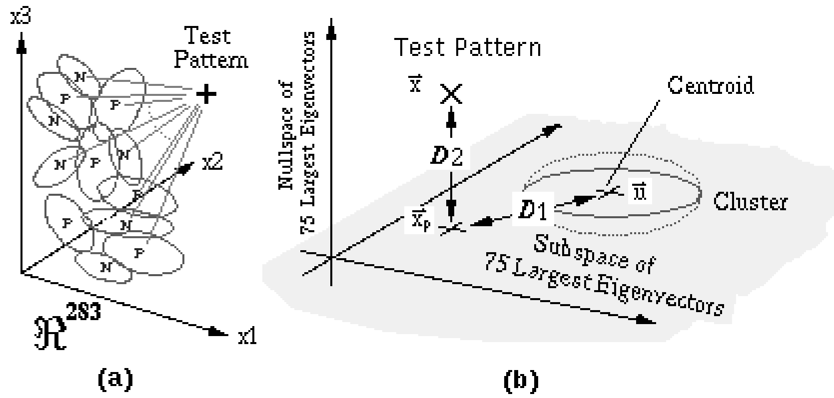


Figure 2.4 [7] (a) A set of 12 distances between test pattern and the model's 12 cluster centroids. (b) Each distance measurement between the test pattern and a cluster centroid is a two –value distance metric.
(Courtesy of K.-K. Sung and T. Poggio)

The 2 distance measures for each of the 12 clusters are input into a neural network classifier that learns to differentiate between the faces and nonfaces.

When the system is given an image in which to find the faces, it takes each sub-window of the image, rescales them to 19x19, applies the preprocessing steps, and computes the distance measures. The distance measures are fed into the trained neural network, leading to a classification of each window as a face or non-Face.

2.1.1.2.4 Neural networks

Rowley [8] uses a multi-layer neural network as classifier. In training part, neural network is trained by face and non-face patterns from face/non-face patterns (the intensity of image), and the same bootstrap method as Sung's[7] is used to obtain enough non-face patterns. In the classification part, each intensity value of pixel in image region input into the trained neural network to classify whether it is a face pattern or not and a decision-making module is used to render the final decision from multiple detection results. Rowley reported their system is less computationally expensive than Sung and Poggio's system, and even have higher detection rates based on a test set of 24 images containing144 faces.

2.1.1.2.5 Support vector machines

Support Vector Machines (SVMs) can be considered as a new method to train polynomial function, neural networks, or radial basis function classifiers. Instead of training classifier based on of minimizing the training error, ie., empirical risk, SVMs minimize an upper bound on the expected generalization error. Osuna [10] used SVMs to deal with face detection problem. By finding the support vectors using training method, SVM aims to find out the boundaries between face and nonface classes [10].

Kwong and Gong [25] extend SVMs to model the appearance of human faces, which undergo nonlinear change across multiple views. By analysis of the nature of face pose distribution, their algorithm can perform both multi-view face detection and pose estimation.

2.1.1.2.6 Boosting

Recently, unitizing a cascade of simple classifiers [47], some researchers try to speed up the face detection processing. Given the fact that face and non-face patterns in real images are skewed, using some extremely fast classifiers with a very low false negative rate and not good false positive rate, as pre-filter can remove some non-face regions rapidly. For this purpose, Viola [47] used Adaboost to train a cascade of simply classifiers. A sequence of simple classifiers is applied to every example. Large amounts of simple features, coming from Haar basis functions, are selected by Adaboost. These selected features will be used in different stages in a cascade of classifiers. The initial classifier eliminates a large number of negative examples with very little processing. Subsequent stages eliminate additional negatives but require additional computation. However, few negative examples remain after several stages. The main contribution of this paper is the adaptation of Adaboost for the task of feature selection and classifier learning. This paper reported that their face detection system can process 15 frames per second, achieves over 90% detection, and a false positive rate of 1 in 1,000,000.

Li [48] used this boosting idea to build a multi-view face detection. Instead of using AdaBoost, they try to use FloatBoost algorithm to avoid the monotonicity assumption in AdaBoost, that when adding a new feature to the current set, that value of the performance criterion does not decrease. By appling FloatBoost, they use a coarse-to-fine, simple-to-complex architecture, which allows system to achieve faster detection processing. In this paper, the speed of 200ms per image of size 320X320 pixels on a Pentium-II CPU of 700 MHz was reported.

2.1.2 Face detection in the compressed domain

Performing analysis in the compressed domain (JPEG/MPEG) reduces the amount of effort involved in decompression. Moreover, DCT coefficients, as features, are attractive for pattern recognition since DCT based compression reduces spatial redundancy and gives compact information about patterns.

Wang and Chang [26] combine chrominance, shape, and DCT frequency information to achieve high-speed face-detection without decoding of the compressed video sequence. Luo and Eleftheriadis[27] perform face detection using Sung's [7] Gaussian mixture model in the compressed domain. Chua, Zhao and Kankanhalli [33] propose a face detection method that uses the gradient energy representation extracted directly from the compressed MPEG video data. To tackle face recognition problem, [17] and [21] use DCT coefficients to build HMM's that operate entirely in the compressed domain.

Among these approaches for face detection in the compressed domain, Luo's method [27] applying Sung's Gaussian mixture distribution-based face model [7] in the compressed domain for face detection achieved better results. They design face detection as a 1-dimensional vector classification problem. In the DCT domain, feature vectors are created directly from (block based) DCT parameters. Using DCT value instead of gray value of image as data of high dimension vector to create the clustered Gaussian distributions, which model the face and nonface distribution in high dimensional space. After that, system calculates distances between input test image (getting vector from DCT value) and center of clustered Gaussian distributions. System classifies whether it is face or not by those distances.

However, in the compressed domain, a major problem to overcome is that the image frames are divided into 8 by 8 blocks before DCT transform. Therefore, any detection work based on DCT parameters have be done at the locations of blocks rather than pixels. That is the blocks reduce the spatial resolution of the system by 8,which makes it hard to detect small faces without fully decoding the image from the block based DCT parameters to pixels. This problem is called *block quantization (alignment) problem.* In order to solve this problem, Luo [27] uses 16 out of 64 possible spatial alignment positions for each training face to generate extra training samples. However, this method gives extra variations, which not belong to the face variations, to the face class and consequently gives low detection rate.

2.2 Face tracking

In many areas such as object oriented image coding, security control, expression recognition and intelligent man-machine interaction, there are several face tracking approach in literature, but they often make assumptions which prevent then from performing well in practical videos (home video, movies), especially for the multi-face scenarios.

2.2.1 Face tracking in pixel domain

Some of the researchers use skin color as main feature to track a human face in video[31][18]. Spors and Rabenstein [31] proposed a real time face tracker using skin color and PCA based eye localization. In first stage, skin color segmentation is performed using statistical models for human skin color and in order to improve the skin color segmentation result, a foreground/background segmentation and an adaptive background update scheme were added. In the second stage, the segmented skin color region is further used to compute the position and side of the dominant facial region, utilizing a PCA based localization method.

Many researchers are trying to make better face trackers by use of more than one sources of measurement, in order to improve robustness. In [32] Rehg, Murphy and Fieguth use a Bayesian network to fuse sensor data available from sensors for skin colour, texture, voice etc. The results of this fusion are used to identify the presence of speaker and to track the speaker. The paper uses virtually all the sensors sources that can be used, making it a complete system. But the goal is to identify the speaker and not track it through a long sequence of frames. So substantial 'tracking' results are not presented. Also it is not certain if the fusion of the sensor data could have been performed better using a different method.

However, strong domain assumptions (fixed scene, one human face assumptions) and absence of strong face model make these methods have typical failures with unconstrained scenes in practical video.

2.2.2 Face tracking in compressed domain

For purpose of efficiency, the *FaceTrack* system, described in [34] by Wang ,Stone and Chang, does face tracking in compressed domain. Faces are detected in the I frames using skin colour, shape and energy features. The face(s) detected are tracked along the sequence of frames using motion vector information and Kalman filter prediction with four different motion models. The system mainly relies on the face detector output, and does not use tracking for resolving multiple tracks. Ambiguities in case of closely placed faces in a multi-face scenario are resolved by multiple hypothesis approach. Due to lack of strong model for face spatial distribution, this algorithm may not work well in some videos.

2.3 Discussion

From the review, we notice that many successful pixel domain face detection methods can be easily adapted to work in the compressed domain by using DCT coefficients as features. Even Gaussian distribution can be built on the DCT coefficients since the DCT is an orthonormal transform, and, *Euclidean distance* and *Mahalanobis distance* can still be employed after the transform [27]. We also notice that most of current face detection methods are designed for gray images, not for compressed videos. There are few face detection approaches available for the compressed videos. However, they are not robust enough for the purposes of indexing. Therefore, developing a compressed domain face detection algorithm could be possible direction.

We compare the face track and face detection methods in literature in table 2.1 with respect to the tasks of finding face sequences in videos. Some algorithms for front-view pose do quite well such as [7][8][25][47][48] in gray images. However, to tackle the problem of finding face sequences in video, face detection methods are often limited by their inability to detect faces in various poses (most of the faces in home video are non-frontal), due to the complexity of the videos and variations of the human faces. In table 2.1 we can see first, there are few works talking about face detection in compressed videos by analysis of temporal attributes, especially for indexing purposes. Secondly, strong domain assumptions (fixed scene, one human face assumptions) and absence of strong face model for spatial information make the face tracker methods (HCI and video conference field) have typical failures with unconstrained contexts in real-world videos. In other words, for content-based video indexing system, lack of analyzing the temporal attribute of videos in current face detection and strong domain assumptions and lack of analyzing the spatial attribute of videos in current face tracker make finding faces in video difficult.

Therefore, it is imperative to find a robust algorithm, which can model three-dimensional distribution (two spatial distributions and one temporal distribution) of faces in videos. As mentioned in chapter 1, a fusion approach to connect face detector and object tracker could be possible solution.

<table>
<tr><td colspan="7">Finding face sequences in video</td></tr>
<tr><td>Research</td><td colspan="3">Face detection</td><td colspan="3">Face track</td></tr>
<tr><td colspan="7">Domain</td></tr>
<tr><td>Domain</td><td>Pixel</td><td colspan="2">Compressed</td><td colspan="2">Pixel</td><td>Compressed</td></tr>
<tr><td>Processing Data</td><td>Image</td><td colspan="2">Video</td><td colspan="2">Video</td><td>Video</td></tr>
<tr><td>Target application</td><td>Image</td><td colspan="2">Compressed video (indexing)</td><td colspan="2">Videoconference. HCI, and etc.</td><td>Compressed video (indexing)</td></tr>
<tr><td>Domain assumptions</td><td>Do not have</td><td colspan="2">Do not have</td><td colspan="2">One face in the fixed scene</td><td>One face in the fixed scene</td></tr>
<tr><td colspan="7">Information Used</td></tr>
<tr><td>Spatial info.</td><td>Used</td><td>Used (But not robust)</td><td>Not used</td><td>Not used</td><td colspan="2">Not used</td></tr>
<tr><td>Temporal information</td><td>Not used</td><td>Not Used</td><td>Used</td><td>Used</td><td colspan="2">Used</td></tr>
<tr><td colspan="7">References</td></tr>
<tr><td></td><td>[2][3][7]
[8][9][10]
[47][48]</td><td>[27][33](Not robust)</td><td>Only [26]</td><td>[31][32]</td><td colspan="2">[34]</td></tr>
</table>

Table 2.1. Comparison of face detection and face track methods with respect to the tasks of finding human faces in videos.

Chapter 3

Face Detection in Compressed Domain

This chapter mainly discusses face detection in the compressed domain. The main focus of this chapter is on the neural network based method used in pixel domain by [8], which has been adapted into compressed domain, by using DCT coefficients of Y components (luminance) directly (instead of gray intensity in pixel domain). In order to achieve this, a DCT coefficients feature extraction scheme is proposed.

DCT coefficients, as features, are attractive for pattern recognition since DCT based compression reduces spatial redundancy and provide compact information about patterns. Moreover, it would be efficient if face detection can be implemented entirely in the compressed domain, without performing the inverse DCT followed by feature extraction, for thousands of compressed videos. The features used for this purpose are the DCT coefficients of Y components (luminance), and Cr and Cb components (chrominance) available from the compressed data of I-frames in MPEG videos. Since DCT coefficients capture frame information concisely, use of DCT features reduces the complexity of the neural network used in the algorithm. In addition, it increases the computational efficiency. The data is used in two stages: in the first stage, a skin color filter based on Cr and Cb DCT coefficients is used to locate skin regions. In the second stage, a 4×4 blocks sized window is used to scan the skin regions in the compressed domain frame to extract the Y-DCT features. A neural network then is trained using these DCT coefficient features to classify patterns as faces or non-faces.

3.1 FACE DETECTION USING DCT COEFFICIENTS OF Y-COMPONETS IN THE COMPRESSED DOMAIN

3.1.1 Face detection procedures

In the uncompressed or pixel domain, many successful face detection methods share the following common algorithm [27](as shown in figure 3.1), which works on grayscale images. A fixed size rectangle window (Sung [7] uses masked 19 pixel by 19 pixel window, Rowley [8] uses 20 pixel by 20 pixel window) is used to scan the whole image, extracting luminance features at each point. These features are used to build face models that encode the distribution of the face pattern. At each point the pattern extracted is compared with a previously trained face model (Sung [7] uses Gaussian Mixture Distribution Model, Rowley [8] uses Neural Networks, and Osuna [10] uses SVMs) to decide if it belongs to a face or not. Moreover, to detect faces at different scales, the image is needed to be repeatedly downscaled by a fixed factor (until the image size is equal to or bigger than the scanning window size) for extracting the features.

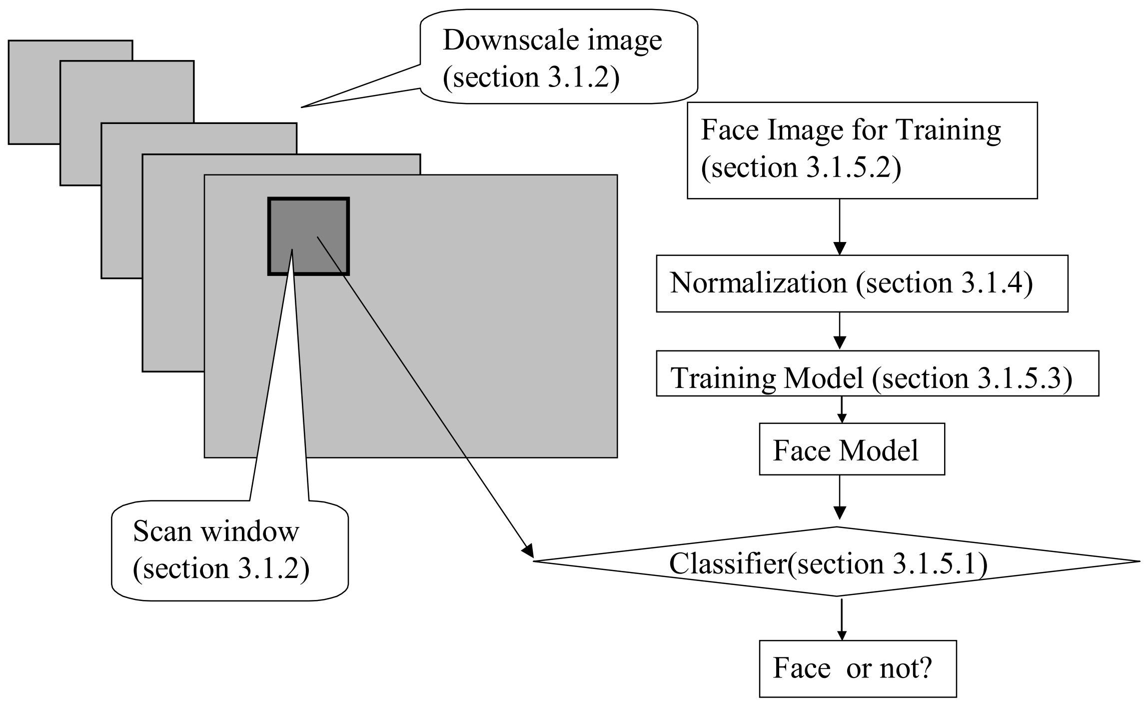


Figure 3.1 A common face detection procedure in a image

A fixed size rectangle window is used to scan the whole image, extracting luminance features at each point. These features are used to build face models that encode the distribution of the face pattern. At each point the pattern extracted is compared with a previously trained face model to decide if it belongs to a face or not. Moreover, to detect faces at different scales, the image needs to be repeatedly downscaled by a fixed factor (until the image size is equal to or bigger than the scanning window size) for extracting the features.

We adapt this procedure in the compressed domain. Each part in the procedure will be discussed in the corresponding section labeled in the graph.

Since the above method has proven to be highly successful in the pixel domain, we adapt it to the compressed domain.

3.1.2 The DCT transform

MPEG compression standard uses the block-based discrete cosine transform (DCT). Basically, in every I frame, the frame is sampled using non-overlapping blocks of the size 8 ×8 pixels, that are transformed utilizing the 2D DCT. The coefficients of the transformed block are quantized and then coded by a Huffman entropy encoder.

In the rest of this chapter, compressed DCT domain would imply JPEG image and MPEG I frames that have been partially decoded (i.e. entropy decoded and de-quantized) so that the DCT coefficients are available in 8 by 8 block structures. In the following we discuss grayscale face detection by luminance components (Y) in the compressed DCT domain. The usage of chrominance components is introduced in section 3.2 later.

In compressed domain, the DCT coefficients as features have the following attributes [27]:

- DCT is an orthonormal transform; both the *Euclidean distance* and the *Mahalanobis distance* are unchanged after the transform, only if all frequencies are used. Therefore, statistical models or other pattern recognition methods may not be influenced by converting the problem to the DCT domain if we ignore consider the blocks and quantization errors incurred.

- DCT domain is more attractive than the pixel domain for pattern classification problems in that the DCT transform reduces the correlation among individual components and compresses the feature energy to the low frequency parameters. In other words, DCT coefficients have compact information about the pattern of interest. Moreover, it is much easier to choose feature components directly from DCT parameters than from pixel values in compressed video. Therefore it is practical for us to directly use DCT coefficients as features to classify patterns (human faces).

In order to demonstrate that human faces still cluster in the high dimensional DCT coefficient feature space, we determine eigenfaces based on DCT coefficients. The extracted DCT coefficient feature vector has 126 elements for a 32 pixels by 32 pixels window (a 4 by 4 block) in the compressed domain (the feature extraction method will be discussed in the section 3.1.3, as shown in figure 3.3). Let the training vector extracted from each compressed domain face be Γ_1, Γ_2, Γ_3, … , Γ_M. The average of the set is defined as:

$$\Psi = 1/M\sum_{n=1}^{M}\Gamma_n \qquad (1)$$

The difference of each training image from the average image is defined by the vector $\Phi_i=\Gamma_i-\Psi$. The calculating of eigen vectors is performed on the set of vector Φ to get the 126 orthonormal vectors K_m, and their associated eigenvalues λ_k. The bigger the eigenvalues λ_k, the larger the variations in the corresponding orthonormal vector K_m direction. We use the elements of vector K_m as DCT coefficients to again construct image, which is known as an eigenface. Figure 3.2 lists the first 10, 20th and 100th eigenfaces. From those images, we can see that first few eigenvectors looks like human faces and eigenvectors become less face-like as the eigenvalues become smaller. It means that variations of human faces can be represented in lower DCT coefficient feature space. Linear combinations of first few eigenvectors roughly span the space of human face. In other words, the attribute of human face clustering in lower dimensional space in pixel domain does not change when using the DCT coefficients to represent them.

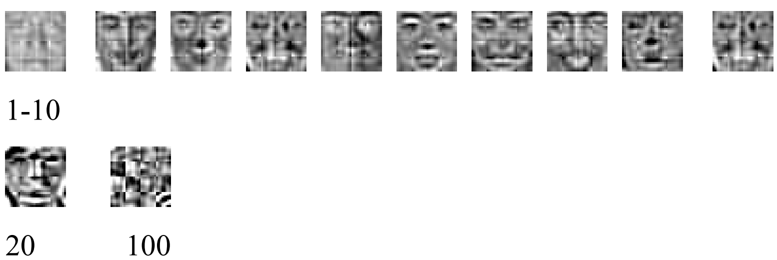


Figure 3.2 1st-10th , 20th , 100th eigenfaces created using DCT coefficients as features

.

First few eigenvectors looks like human faces and eigenvectors become less face-like as eigenvalues become smaller.

Therefore, a statistical model or other pattern recognition methods may be not be influenced by converting the problem to the DCT domain if we do not consider the blocking and quantization errors incurred.

However there are two problems that need to be tackled before applying pixel domain pattern recognition methods to the compressed domain [27]:

- *Block quantization (aligning) problem*

A major problem to overcome is that the image frames are divided into 8 by 8 blocks before the DCT transform is applied. Therefore, any detection work, based on DCT parameters, has to be done at the locations of blocks rather than pixels. That is the blocks reduce the spatial resolution of the system by 8,which makes it hard to detect small faces without fully decoding the image from the block based DCT parameters to pixels.

Luo and Eleftheriadis[27] tackle this problem by including additional face patterns arising from different block alignment positions as positive training examples. But this method induces too many variations, not necessarily belonging to inter-class variations, to the positive training samples. This in turn makes the high frequency DCT coefficients unreliable for both face model training and face detection functions.

Fortunately, there exist fast algorithms to calculate reconstructed DCT coefficients for overlapping blocks [30][36][37]. These methods help to calculate DCT coefficients for scan window steps of even upto one pixel. However a good tradeoff between speed and accuracy is obtained by reconstructing the coefficients for scan window steps of two pixels.

Down-sampling (Down-scaling) problem

In the pixel domain, it is easy to down-sample in arbitrary ratios. But in the compressed domain, it is a little more expensive to carry out resolution transforms for any arbitrary ratio.

In order to detect faces of different sizes, compressed domain image downscaling is used before extracting DCT coefficients. Several fast downscaling algorithms that operate directly in the compressed domain exist in [36][37][38]. We apply the algorithm in [38] which can down-sample image and video by a fractional factor of 1.25 in the DCT domain. This algorithm permits us to derive a wide range of scaling factors by cascading several scaling processes, such as 1.25 1.56 1.95 3.05 3.81.

3.1.3.Feature extraction

In compress domain, frame is divided into 8 by 8 pixels sized block to store DCT coefficients. As shown in figure 3.3, we adopt 4 blocks by 4 blocks sized rectangle (32 pixels by 32 pixels) as scan window to extract DCT coefficient features of Y components, i.e. These coefficients form a feature matrix.

It is not necessary to extract all the DCT coefficients from all sixteen blocks from that scan window The bottom-left and bottom-right blocks (shown in dark in figure 3.3) are ignored, since the most texture information in these two blocks comes from the background. Due to the fact that the feature energy congregate in the low frequency components, the low frequency DCT coefficients retain enough encoded information to make inter-class distinctions (i.e. distinguishing a face from a non-face region).

The first ten low frequency coefficients in each block of sixty-four coefficients are chosen. Moreover, as far as learning is concerned, the goal of feature selection should be to select features that are less sensitive to intra-class differences (i.e. differences within faces) but significantly sensitive to inter-class variations. Since we found that the DC values encode the variations mostly resulting from illumination and camera properties, rather than inter-class differences, we choose to ignore the DC values of each block in the scan window. In all, in the scan window, fourteen blocks are chosen out of the sixteen blocks and in each block, nine low frequency coefficients are chosen, i.e. the feature matrix has 126 coefficients (9 coefficients/block * (16-2) blocks) for a 32 pixels by 32 pixels scan window (a 4 by 4 block). In comparison with the pixel domain methods, where the number of features for classification is large (20 pixels by 20 pixels face pattern needs 400 features), the number of features used in the compressed domain is much smaller. This feature matrix (Figure 3.3) is simply converted to a 1D vector for classification using a neural network.

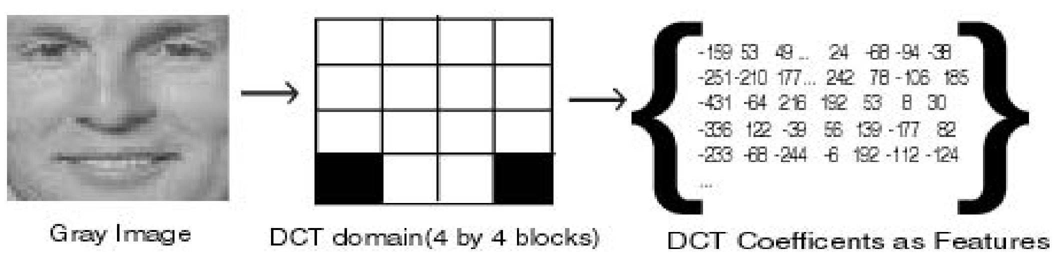


Figure 3.3 DCT coefficient extraction

The bottom-left and bottom-right blocks (shown in dark) are ignored, since the most texture information in these two blocks comes from the background. It is not necessary to extract all the DCT coefficients from all sixteen blocks. Therefore, the first ten low frequency coefficients in each block of sixty-four coefficients except for DC values are chosen i.e. 9 coefficients per block. In all, the feature matrix has 126 elements for a 32 pixels by 32 pixels scan window (a 4 by 4 block). In comparison with the pixel domain methods, where the number of features for classification is large (20 pixels by 20 pixels face pattern needs 400 features), the number of features used in the compressed domain is much smaller. This feature matrix is simply converted to a 1D vector for classification using a neural network.

3.1.4 Normalization

DCT coefficients in different locations of each block have different orders of magnitude (The lower the frequency, the bigger the magnitude). Therefore, we need to estimate the upper bound and the lower bound of DCT coefficients and use them to convert the coefficients into [0,1]. This helps preventing the large valued feature from dominating the detection process.

Suppose $x_1,$ x_1, x_2, …, x_n are the DCT coefficients retained from the feature extraction stage and $x_1^{(j)}$,$x_2^{(j)}$,…,$x_n^{(j)}$; j = 1,2,…,p are the corresponding DCT coefficients retained from the training examples where n is the number of DCT coefficient features retained (currently, we use 126 DCT coefficient features extracted from 4×4 block sized square window) and p is the number of training samples. The upper bounds (U_i) and lower bounds (L_i) can be estimated by

$$U_i = \alpha * \max\{1, x_i^{(1)}, \ldots, x_i^{(p)}\}, \; I = 1,2,\ldots,n; \tag{2}$$

and

$$L_i = \alpha * \min\{-1, x_i^{(1)}, \ldots, x_i^{(p)}\}, \; I = 1,2,\ldots,n; \tag{3}$$

Where α >=1 is a factor to extend the bounds (here we set α = 1). Then the normalized vectors

$Z_1^{(j)}$,$Z_2^{(j)}$,…,$Z_n^{(j)}$; j = 1,2,…,p

can be determined by

$$z_i^{(j)} = 2\frac{x_i^{(j)} - L_i}{U_i - L_i} - 1, \; i = 1,2,\ldots, n \tag{4}$$

3.1.5 Neural network-based classifier

3.1.5.1 Structures

Artificial neural networks, commonly referred to as “neural networks”, is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. By trying to simulate the work of the neurons in the brain, the work of neural networks is entirely different way from the conventional digital computer.

The basic computational element (neuron) is often called a node or unit. It receives input from external sources or some other units. Each input has an associated weight w, which serves as parameters of neuron, which can be modified so as to model synaptic learning. The unit computes some function f of the weighted sum of its inputs:

$$y = f(\sum_j w_j x_j) \tag{5}$$

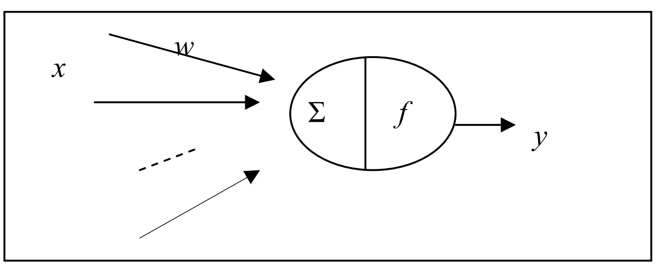


Figure3.4 Neuron

Its output, in turn, can serve as input to other units. The weighted sum $w_j x_j$ is called the net input to unit j, often written net_j. The function f is the unit's activation function. In the simplest case, f is the identity function, and the unit's output is just its net input. This is called a linear unit.

The most used neural networks is fully connected three layer feed-forward neural networks that employ a sigmoid activation function, also called a squash function. A neural network is built using neurons (as shown in figure 3.4), which are the basic information treating units. We use a neural network as classifier to classify patterns into faces and non-faces.

So now for a given MPEG I video, each I-frame is scanned in steps of two pixels by a sliding window of 4×4 blocks passing through it. From that sliding window, DCT coefficient features are extracted using the scheme in section 3.1.3. These features are normalized (section 3.1.4) and put into one trained neural network (126 input units, 20 hidden units, and 1 output unit) to classify whether the pattern inside the sliding window is a face pattern or not. The structure is shown in figure 3.5. To detect faces larger than the size of this sliding window, the window is passed over successively on images downscaled by a factor of 1.25 as in [38].

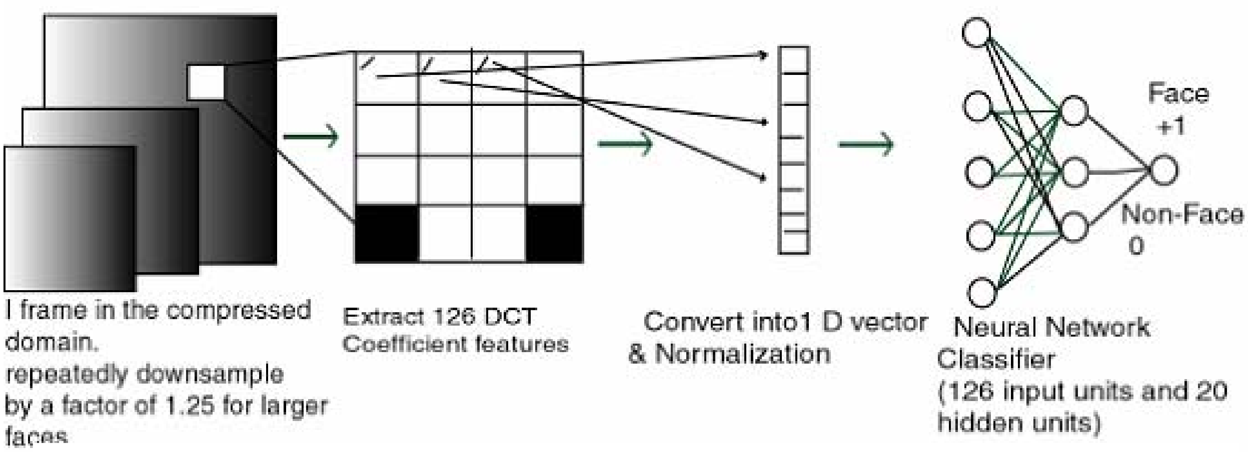


Figure 3.5. System for the proposed neural network based face detection in the compressed domain

DCT coefficient features are extracted using the scheme in section 3.1.3. These features are put into one trained neural network (126 input units, 20 hidden units, and 1 output unit) to classify whether the pattern inside the sliding window is a face pattern or not after normalization.

3.1.5.2 Training Data Preparation

Before we use a neural network as classifier, the neural network has to be trained to tune its parameters. In order to detect faces, we have to provide the neural network both positive and negative training samples. We use the front-view face type to create positive training samples. Figure 3.6 shows a sample face and a sample of non-face for training purpose. All the training samples are scaled to 32 by 32 pixels size before processing the feature extraction procedure.

Figure 3.6 Positive and negative samples (scaled to 32by32 pixels)

For the positive training samples, as the initial training sets, we have collected 1088 face samples. In order to improve robustness of face detection process, some of the positive training samples are taken at closer distances. For the sake of increasing training samples, we follow Sung [7]'s method of synthesizing positive samples by slightly rotating and mirroring images.

For the negative training samples, clearly, it is hard to define good representatives of the so-called non-face space, since non-face space is huge. Given the infinitely huge non-face space, it is difficult to collect all the samples, which are scattered in all the area of non-face space. Since training of neural network requires many negative training samples, we expand the non-face training sample set by applying the bootstrap algorithm during training. It allows adding non-face images to the training database that are of importance with regards to improving the neural networks performance at a given time during the training period. Additional non-face images can be obtained by scanning images containing no faces, by automatically clipping false detections and by inserting these into the current training set. Then, we use 2000 non-face samples, as expected, the networks false detection rate diminished greatly in our experiments when applying the bootstrap method.

Since the training sets available with us were pixel domain images, they had to be first converted to obtain compressed coefficient features. In order to get the same accurate DCT coefficient as in MPEG, we convert those raw data into DCT coefficients by DCT transformation, quantization and de-quantization (shown in figure 3.7) using the default MPEG quantization table.

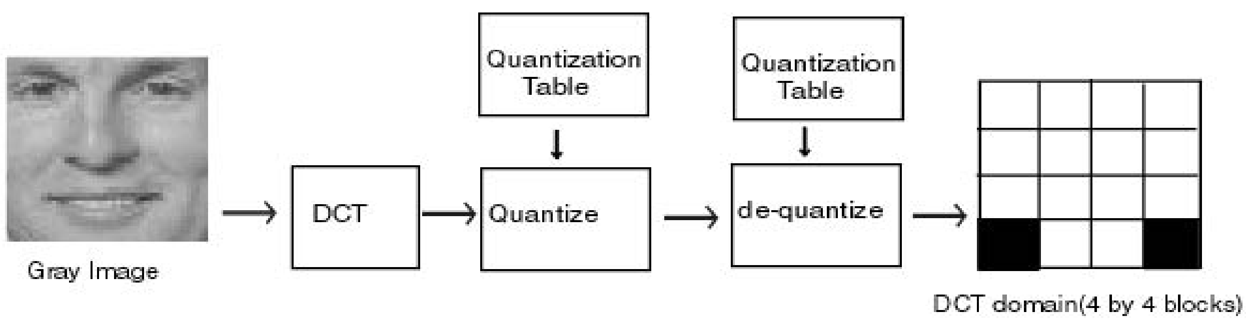


Figure 3.7 Training data preparation

In order to get the same accurate DCT coefficient as in MPEG, we convert the raw data into DCT coefficients by DCT transformation, quantization and de-quantization.

3.1.5.3 Neural network training

3.1.5.3.1 Training procedure

There are three types of datasets: training dataset, validation dataset, and test dataset. Training datasets is used to tune the neural network by the back propagation algorithm, validation dataset is used to decide when to stop training the neural network by monitoring the error over the validation dataset and test dataset is to get the extra negative samples using the bootstrap method.

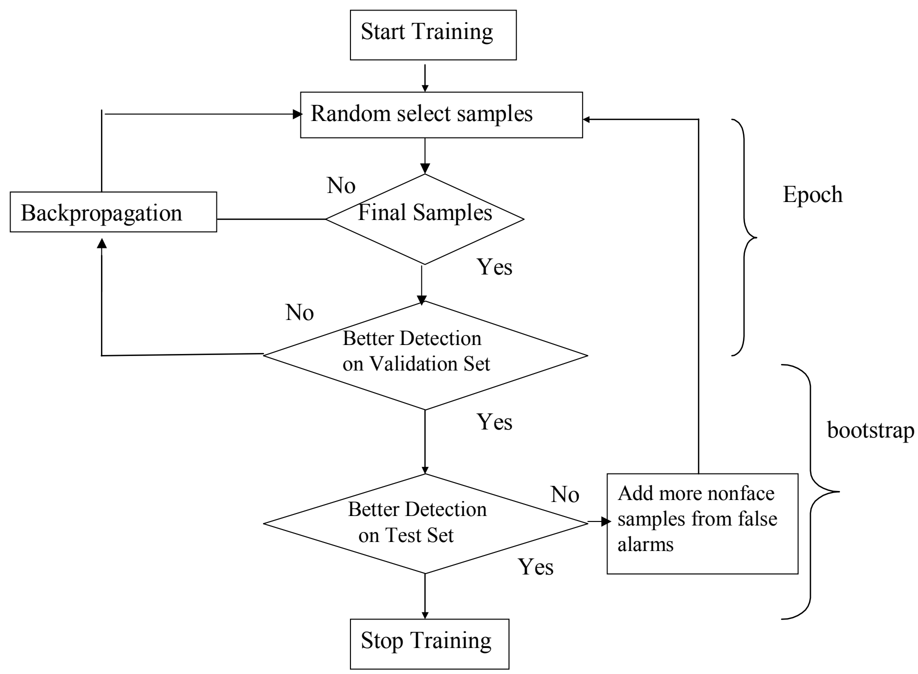


Figure 3.8 Neural network training procedures

First, randomly selected positive or negative training sample from training data sets is put into the neural network and consequently the weights of neural network are updated by the back-propagation algorithm (see section 3.1.5.3.2). After all the training samples in training dataset are used up, validation dataset is tried out to decide whether to stop this turn of bootstrap or not by monitoring the training error. If a good performance with validation dataset is obtained, then the neural network starts the next bootstrap after getting extra negative training samples by finding false alarms from the test dataset. Otherwise training continues in this manner.

There are three most useful training protocols: stochastic, batch and on-line. In stochastic training, training samples are chosen randomly from the training set, and the network weights are updated for each sample presentation. This method is called stochastic because the training data can be considered to be a random variable. Note that network weights are updated when each training sample is put into the neural network. In batch training, all training samples are presented to the network before learning takes place. In online training, each training sample is presented once and only once. This is for the case when training samples have to be obtained online instead of offline. Here we use stochastic training since it is fast and it performs well.

The neural network training procedure is shown in figure 3.8. First, randomly selected positive or negative training sample from training data sets is put into the neural network and consequently the weights of neural network is updated by the back-propagation algorithm(see section 3.1.5.3.2). After all the training samples in training dataset are used up, the validation dataset is tried out to decide whether to stop this turn of bootstrap or not by monitoring the training error. If a good performance at validation dataset is obtained, then the neural network starts the next bootstrap after getting extra negative training samples by finding false alarms from the test dataset. Otherwise training continues in this loop. Our experiments show that the neural network can achieve about 92% classification rate on validation dataset after over 7-8 bootstraps.

The change of training error while keep training are shown in figure 3.9. The changes of classification rate, false alarm rate and miss rate are shown in figure 3.10. From the figures, we can see that initial all the data have oscillation, but after hundreds of epochs, they converge well.

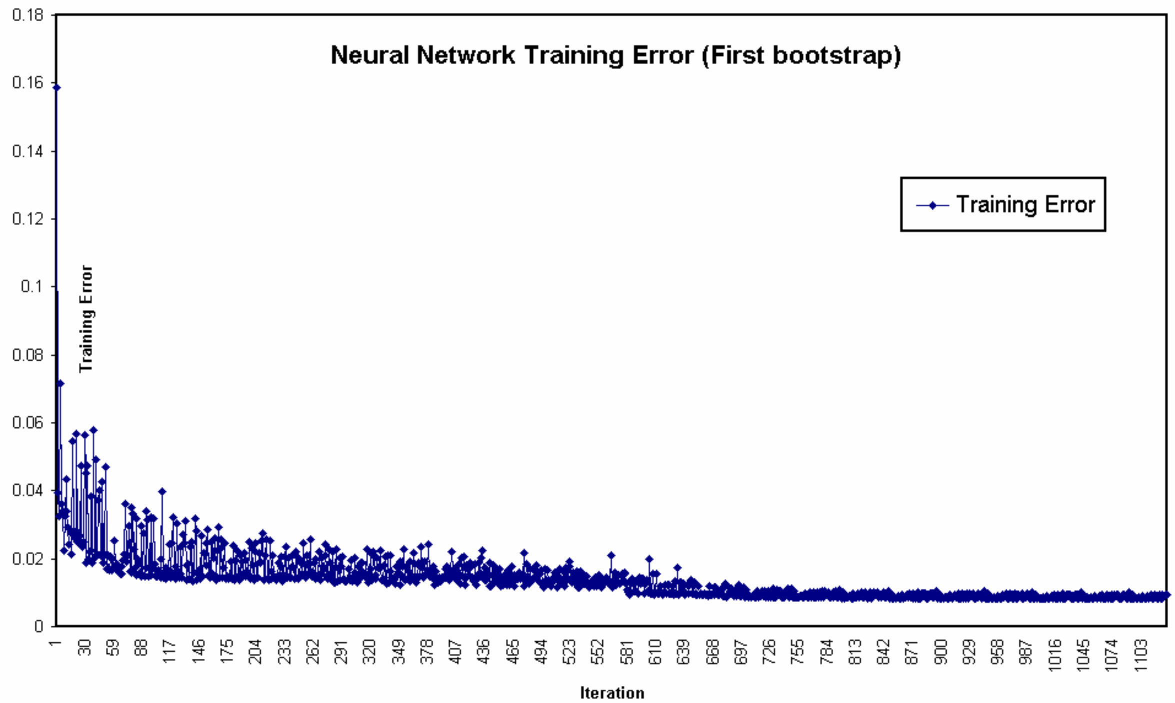


Figure 3.9 Training errors for back propagation algorithm during training on validation set.

Initial the training error has oscillation, but after hundreds of epochs, it converges well.

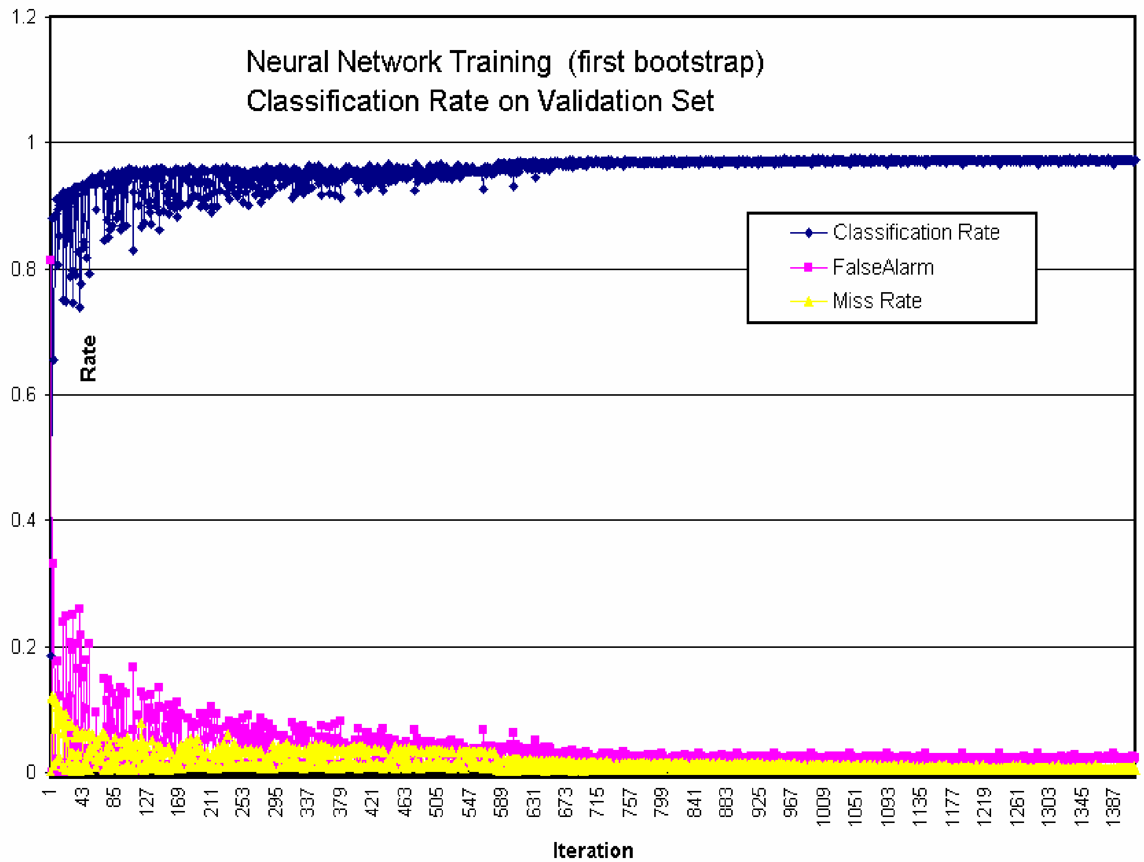


Figure 3.10 Classification rate, missing rate and false alarm rate for back propagation algorithm during training.

Initial these rates have oscillation, but after hundreds of epochs, they converge well.

3.1.5.3.2 Back propagation learning

A back propagation weight tuning method is used to training the three layer fully connected neural network (126 input units, 20 hidden units, and 1 output unit). Among the supervised learning algorithms, the back propagation algorithm has emerged as the most widely used and successful algorithm for training multilayer feed forward networks. The back propagation learning includes two phases: the forward phase and the backward phase. In the forward phase the input signals propagate through the network layer, eventually producing some response at the output of the network. The actual response produced is compared with a target response (from the label of the training data), generating error signals that are the propagated in a backward direction through the network. In the backward phase of operation, the free parameters of the networks are updated in order to minimize the sum of squared errors [46].

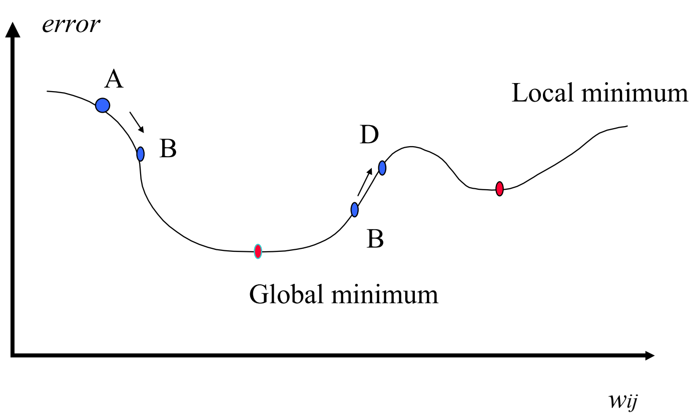


Figure 3.11 Local minima and convergence of neural network
w_{ij} is the weight that needs to be adjusted while training *error* means the error between the training example target values and the network outputs. If we start training from point B, the gradient descent will become trapped in the local minima. However, if starting from point A, the global minimum can be obtained.

3.1.5.3.3 Local minima and convergence

As mentioned above, the back propagation algorithm provides a gradient descent search through the space of possible network weights, iteratively reducing the error E between the training example target values and the network outputs. Due to the fact that the error surface for neural network may contain many different local minima, gradient descent can become trapped in any of these different local minima. As a result, neural network trained by back propagation algorithm is only guaranteed to converge toward some local minimum in E and not necessarily to the global minimum error [49], as shown in figure 3.11.

In order to tackle this problem, we use two techniques [49]:

1. A momentum term can be added into the weight-update rules. To see the effect of this momentum term, consider that the gradient descent search trajectory is analogous to that of a ball rolling down the error surface. The effect of momentum is to keep the ball rolling in the same direction from one iteration to the next. This can sometimes have the effect of keeping the ball rolling through small local minima in the error surface, or along flat regions in the surface where the ball would stop if there were no momentum.

2. When training neural network, we use the same training samples, but initialize each network with different random weights. If the different training efforts lead to different local minima, the network with the best performance over a separate validation dataset can be selected.

3.1.5.3.4 Number of hidden units

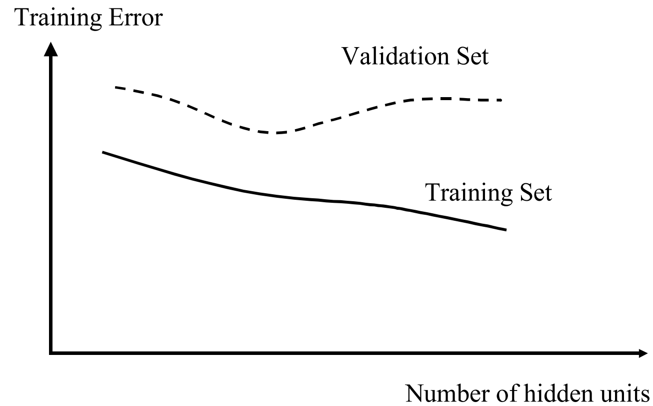


Figure 3.12 Errors on training and validation sets with respect to the different number of hidden units

The training and validating error on a two-category classification problem for networks differ solely in their number of hidden units. For a large number of hidden units, the training error can become small because such networks have high expressive power and become tuned to the particular training set. Nevertheless, in this regime, the validation error is unacceptably high. At the other extreme of too few hidden unites, the neural network does not have enough free parameters to fit the training data well, and again the validating error is high. We need to seek some intermediate number of hidden units that will give low validation error.

The number of input units and output units is decided by the dimensionality of the input vectors and the number of classes, respectively. However, the hidden units need to be defined when designing the neural network. The number of hidden units governs the number of parameters in the neural network (expressive power of the net) to be tuned, and thus the complexity of the classifier (neural network) or the complexity of the decision boundary. Having a larger number of hidden units does not necessarily result in good classification results. Sometimes, larger number of hidden units can make neural network tend to fit all the training samples, even giving worse results, especially when there is some noise in the training set. Too few hidden units tend to have good generalization ability. However in some cases, they cannot discriminate between the training samples. Without further information there is no foolproof method for setting the number of hidden units before training in neural network.

Figure 3.12 shows the training and validating error on a two-category classification problem for networks that differ solely in their number of hidden units. For the large number of hidden units, the training error can become small because such networks have high expressive power and become tuned to the particular training set. Nevertheless, in this regime, the validation error is unacceptably high. At the other extreme of too few hidden units, the neural network does not have enough free parameters to fit the training data well, and again the validating error is high. We need to seek some intermediate number of hidden units that will give low validation error.

The number of hidden units determines the total number of weights in the neural network, which we consider informally as the number of degrees of freedom. Normally, number of hidden units depends on the complexity of the problem or number of the training samples. We here adjust the complexity of the network (number of hidden units) in response to the training data, starting with a “large” number of hidden units and prune the hidden units and choosing the best one on validation set.

3.1.5.3.5 Generalization

The aim of training neural network is to let the trained neural network have a good performance on unknown dataset instead of training dataset. However, if we train neural network according to the training error on the training dataset, neural network may not perform well on new samples, since back propagation algorithm is susceptible to over-fitting the training dataset at the cost of decreasing generalization accuracy over other unseen examples [49]. This is especially so when the parameters in neural network (controlled by the number of hidden unites) is far more than enough, compared to the number of training samples, the neural network tends to fit all of the available training samples.

As shown in figure 3.13, the training error over the training set decreases monotonically as the number of gradient descent iterations grows, while the error over the validation set first decreases, and then increases even as the error over the training examples continues to decrease. The error over the validation set actually measures the generalization accuracy of the neural network. In order to prevent over training, ideally, one should stop training at the point for which error over validation dataset is a minimum. This can be achieved in practice by training the network until the error over validation dataset begins to rise.

In order to prevent the neural network from concentrating too much on certain features, the sequence of the face and non-face training sample is randomly arranged.

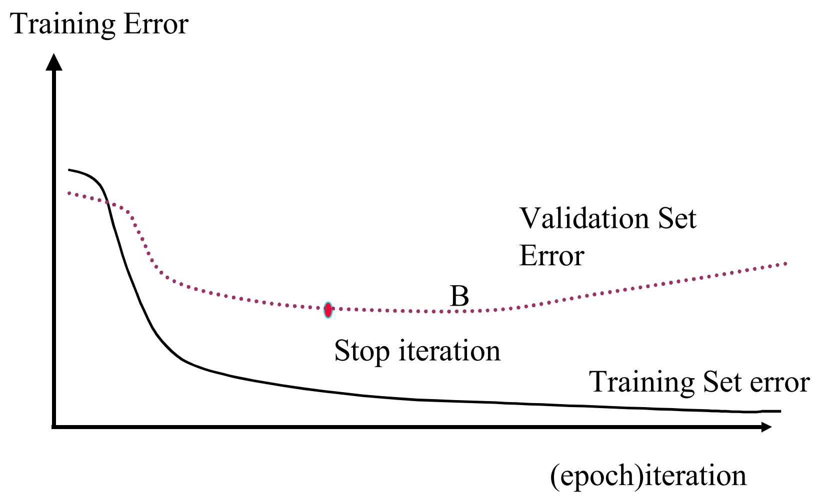


Figure 3.13 Stop Criteria for neural network training

The training error over the training set decreases monotonically as the number of gradient descent iterations grows, while the error over the validation set first decreases, and then increases even as the error over the training examples continued to decrease. The error over the validation set actually measures the generalization accuracy of the neural network. In order to prevent over training, ideally, one should stop training at the point for which error over validation dataset is a minimum. This can be achieved in practice by training the network until the error over validation dataset begins to rise.

3.1.6Experiments

The above algorithm has been implemented in C++. In order to evaluate the performance of our algorithm, we have used some grayscale images from the face detection test sets of CMU (at http://www.cs.cmu.edu/~har/faces.html). This test database consists of 3 subsets of gray images with total of 130 images and 507 faces. Since not all gray images are suitable in our cases (explained later), experiments are performed on 43 gray images belonging to this target. To simulate the compressed domain situation in MPEG, we convert these grayscale images before using them for training purposes (Figure 3.7).

To the best of our knowledge, this is the first time the CMU database has been used to evaluate face detection algorithms in the compressed domain ([27] has used CMU database, but they didn't give the quantitative performance on it). Before comparing with the performance of the pixel domain approaches, the following factors in the compressed domain should be taken into consideration:

	Our Method in Compressed Domain	**Neural network based method [8] in Pixel Domain**
Total Faces	**176**	**507**
Detected Faces	**122**	
Missed Faces	**54**	
False Alarm	**61**	**31**
Detection Rate	**69.3%**	**86%**

Table 3.1. Face detection results on the gray images from the CMU database

1. The conversion from the gray images raw data to the compressed domain DCT coefficients causes some information loss as well as errors.

2. DCT blocks recalculation and I frame downsampling induce some errors. They become more obvious when the factors become larger.

3.The pre-processing algorithms are different. In the pixel domain, non-uniform light conditions are compensated by linear fitting [8] or histogram equalization while, in the compressed domain, we just remove the features (DC), whose variations mainly come from light condition.

4. The scale ratio in the multi-scale search is different. Most of the approaches in the pixel domain use the factor between 1.1 and 1.2 while we use 1.25 instead in the compressed domain.

5. In the pixel domain, a sliding window is shifted pixel by pixel over each image while we shift the sliding window 2 pixels by 2 pixels for the purpose of efficiency.

6. The size of the sliding window is different. We use 32×32 pixels sized square window while they use smaller sized square window (19×19 in [7] and 20×20 in [8]). Since in the most cases, the resolution of the I frames in the compressed video is lower than that of images, finding faces bigger than 32×32 is enough for most of compressed video applications. Because of this, we don't consider faces smaller than 32×32 and only count the face bigger than 32×32 when we use CMU test sets, in this case.

A neural network based face detection system proposed by Rowley [8] achieves a detection rate ranging from 76.5% to 92.5% depending on the heuristic methods and arbitrating among neural networks on the CMU database. Since some of the face size in the CMU gray image test sets are smaller than 32×32, we pick up a subset of some of gray images (43 images), in which the faces are all bigger than 32×32. The detection rate for frontal-view face is almost 70% on 43 gray images from the CMU database as listed in table 3.1 (The final output is a combination of the different scale levels). Figure 3.13 shows some test results on the CMU database (To show the real detection ability, the results shown here are directly obtained from the neural network without removal of overlaps and multi-level combination). These preliminary results prove that our algorithm is reliable, given the condition in the compressed domain. The way to reduce the false alarms and improve the face detection rate of our system is to arbitrate among multiple networks [8] and increase the initial face training samples. Since we are doing this face detection for compressed video, the results obtained here are encouraging enough for us to continue research in this direction.

3.2 Color-based filter combining

3.2.1 Candidate region selection

In previous section, we have discussed face detection method based on the luminance components. However, it is not necessary to search the entire frame area. As we know, each MPEG I frame is divided into 16×16 pixel sized macroblocks and each macroblock is composed of four 8×8 pixel sized luminance blocks and two 8×8 pixel sized chrominance blocks (we assume the color format of MPEG I frame is 4:2:0). A skin color filter , which uses chrominance compoents, can be used to locate skin color regions in each frame, from which the luminance components can be picked up for classification by the neural network. This makes the face detector faster and more robust.

3.2.2 Skin color model

In 4:2:0 format, each macroblock has one *Cb* block and *Cr* block. The DC values of *Cb* blocks and *Cr* blocks represent the average *Cb* and *Cr* values of macroblock respectively. So the DC values of each *Cb* and *Cr* block are used to represent the average chrominance of the corresponding macroblock in a frame. In intra-coded MPEG I frame, the DC values of chrominance blocks are directly available in the compressed domain. In inter-coded P and B frame, the DC values of inter-code block can be simply reconstructed by using the DC values of the reference frame(s) [26].

In normalized color space obtained by dividing the red and green components by intensity, a skin-color distribution forms a tight cluster under constant light conditions irrespective of the ethnic group [13]. An approach similar to [38], is taken to the build skin color model using a bivariate Gaussian distribution model N(μ,Σ) (μ is mean and Σ is covariant matrix)that models the two normalized variables *r* and *b* (*r* is DC coefficients of *Cr* and *b* is DC coefficients of *Cb*), which form the vector *x* in the conditional probability equation (it means the probability of *x* belongs to skin color, given the skin color model *N*):

$$p(x \mid N) = (2\pi)^{-1} \mid \Sigma \mid^{-1/2} \exp\{-d(x) \ / 2\} \tag{6}$$

where d(x) is *Mahalanobis distance* given by

$$d(x) = (x - \mu)^t \Sigma^{-1} (x - \mu) \quad (7)$$

The larger the distance *d(x)*, the lower the probability that the block belongs to the skin color class. To decide the skin color block, we set a threshold H. The block whose value of is *d(x)* smaller than H belongs to skin color regions of the frame. It is difficult to find a universal optimal threshold H for various different video resources, since skin colors vary in different lighting conditions and environments. However, given that we use skin color information to find out the candidate regions for next face detection step, we just choose one loose threshold *H* = *7*.

3.2.3 Experiments

3.2.3.1 Skin color detector

To evaluate the performance of the skin color detector in different light and environment conditions, we use different video clips for testing ranging from news to the lab environment. Figure 3.14 shows the GUI for skin color detector.

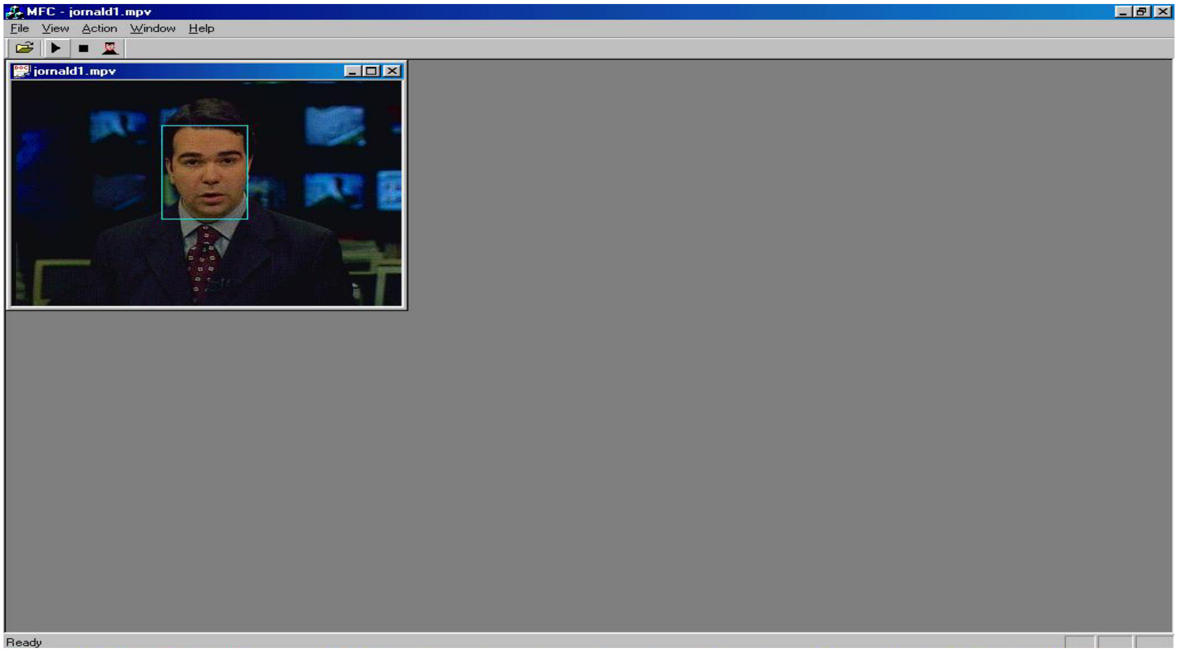


Figure 3.14 GUI for the skin color detector

The skin region (marked by green rectangle) can be found by skin color detector

Table 3.2 lists results of the test on skin color detector.

Threshold=7

	Mpeg7 test (news)	Lab Env.	Local News	Others	Totals
Skin Colors	106	3	17	13	139
Detected	90	2	16	11	119
Missed	16	1	1	2	20
False alarm	84	4	13	5	106
Detection rate	85%	67%	94%	85%	86%
Detection accuracy	52%	33%	55%	69%	53%

Table 3.2: Performance of Skin Color Detector in Video Clips (MPEG1 and MPEG2)

Detection rate: the ratio of correctly detected skin color *v.s*. total skin colors that should be detected

Detection accuracy: the ratio of correctly detected skin color *v.s.* the total detected skin colors

From our experiments, we notice that:

1. Fire, soil, stone, woods and other yellow and red things, which are similar to skin color, are falsely detected sometimes. These are the main cause of the false-detect errors.
2. Missed skin color mainly has two reasons:
   a. Background color is similar to skin color. It cause the detector consider the whole background as skin color. The Skin inside background is therefore missed
   b. Because of lighting conditions and resolution of video clips

When we design skin color model and set parameters, there is trade off between detection rate and detection accuracy. Table 3.4 shows different thresholds for the Gaussian distance that influences the detection rate and detection accuracy.

	Threshold=7	Threshold=5
Skin Colors	29	29
Detected	23	19
Missed	6	10
False alarm	19	17
Detection rate	80%	66%
Detection accuracy	55%	53%

Table 3.4: Influence of Threshold (one news clip)

From table 3.4, we can see, when we make constraint condition tight (threshold =5), the false alarms go down. But we lose out on the detection rate. We use the skin color detector as filter. It means that we need higher detection rate and the higher detection accuracy is not required (can be improved by the subsequent neural network based face detector). Therefore, we choose a loose threshold for this purpose.

3.2.3.1 Combining skin color filter and neural network face detector

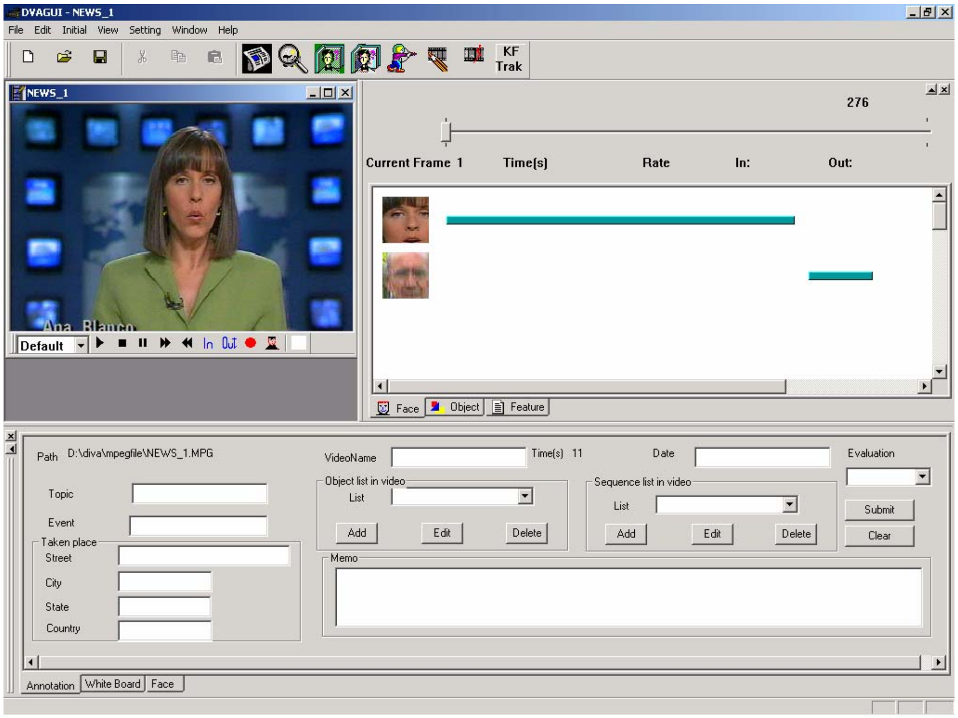


Figure 3.15. GUI for the face detection experiments in DVA system

Video can be played in the left window. Face detection results are shown in the right window. The green bars indicate the duration of the human faces in the video sequence.

We use combined the skin color filter and the neural network face detector in our content-based video indexing application. In order to find out the face pattern extraction ability of this face detector, we extract face patterns in each MPEG I frame and measure the overlaps between every two pair of neighborhood I frames to create face pattern timelines for individuals in video as shown in figure 3.15. We have shown some screen shots of face detection in video ranging from advertisements, news, movies, and workplace environments in figure 3.17, 3.18 and 3.19.

The experiment results show that our face detection system is reliable for video, except in places along the timeline where the face is non-frontal (this problem can be overcome in the future by incorporating a tracking mechanism for faces).

The results also show that by combining skin color filter the system can avoid most of the false alarms and can also speed up the face detection process. Our experiments show that the system can achieve the speed of 127 ms per image of size 252X288 pixels on a Pentium-III CPU of 1GHz.

Note that we cannot present quantitative results for the standard CMU face database because they only have gray scale images. We are currently building our own color video test sets to conduct further experiments for face detection in video.

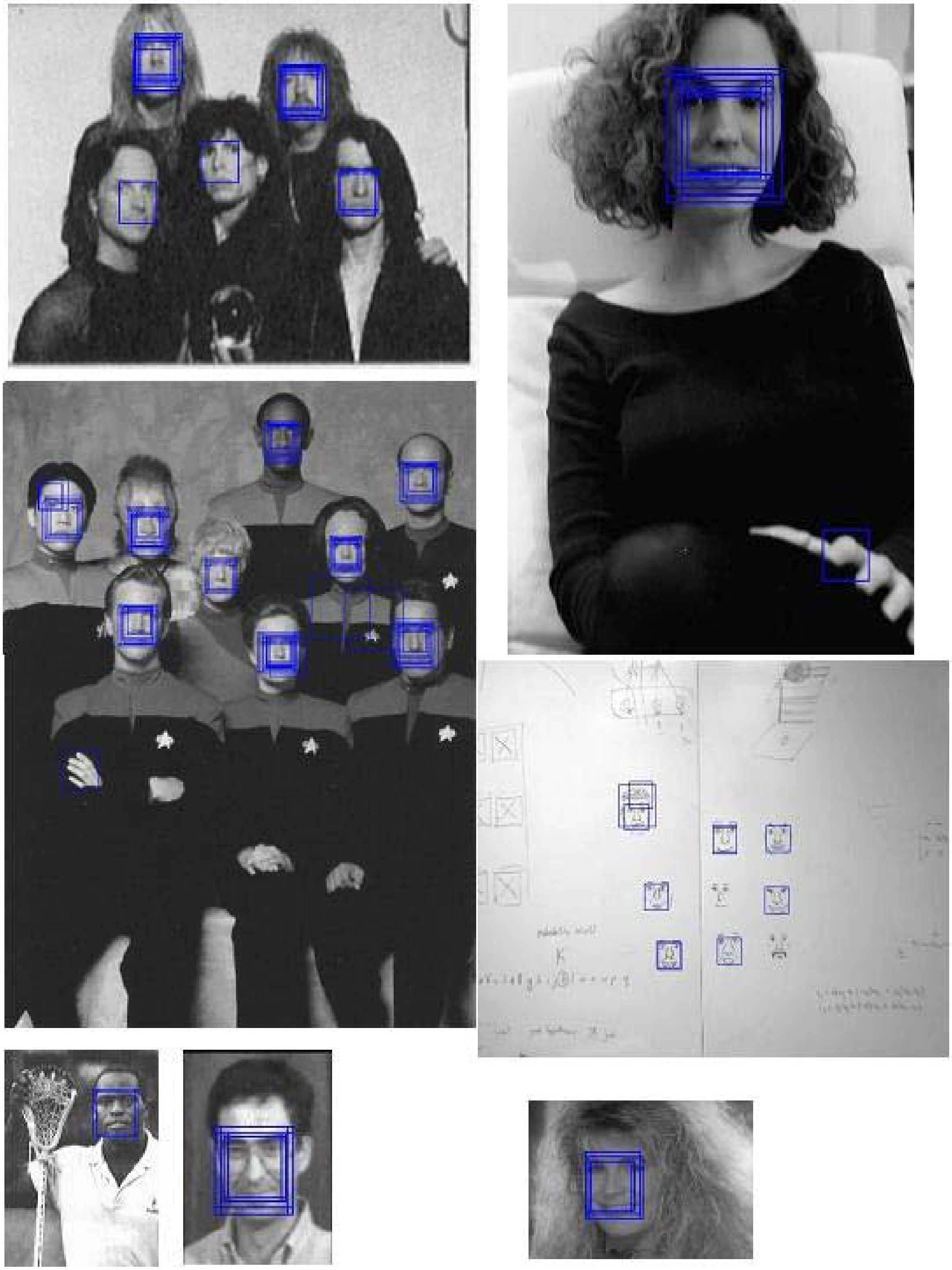
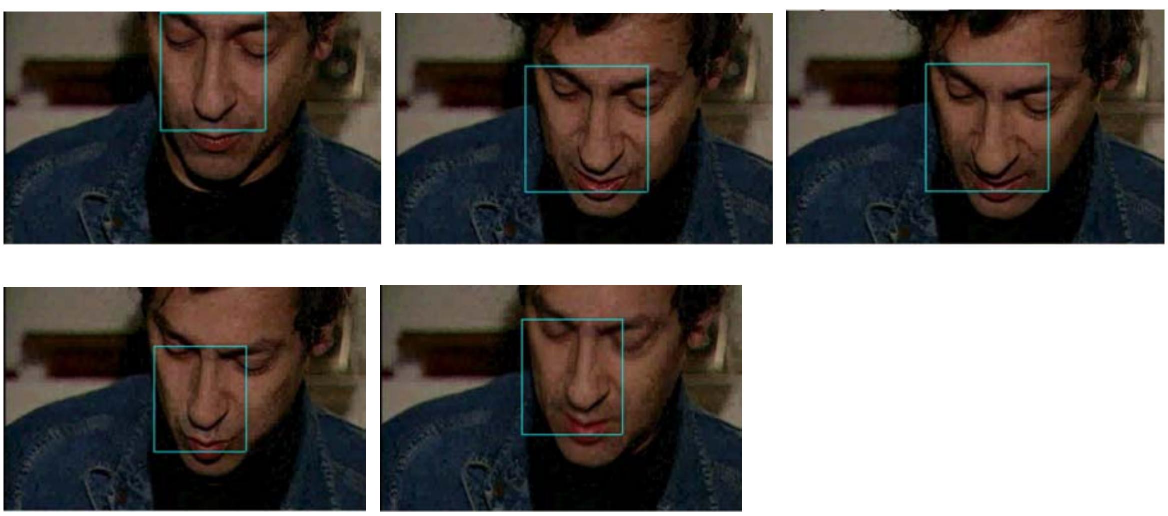


Figure 3.16. Some test results on CMU database.
To show the real detection ability, the results shown here are directly obtained from Neural Network without removal of overlaps and multi-level combination. In order to simulate the compressed domain scenarios, these images in CMU database needs to be converted into the compressed domain (see section 3.1.5.2 for detail methods) before performing our algorithm on them.

Advertisement clip (MPEG 1 format picture size 352x240).An error detection in the first frame

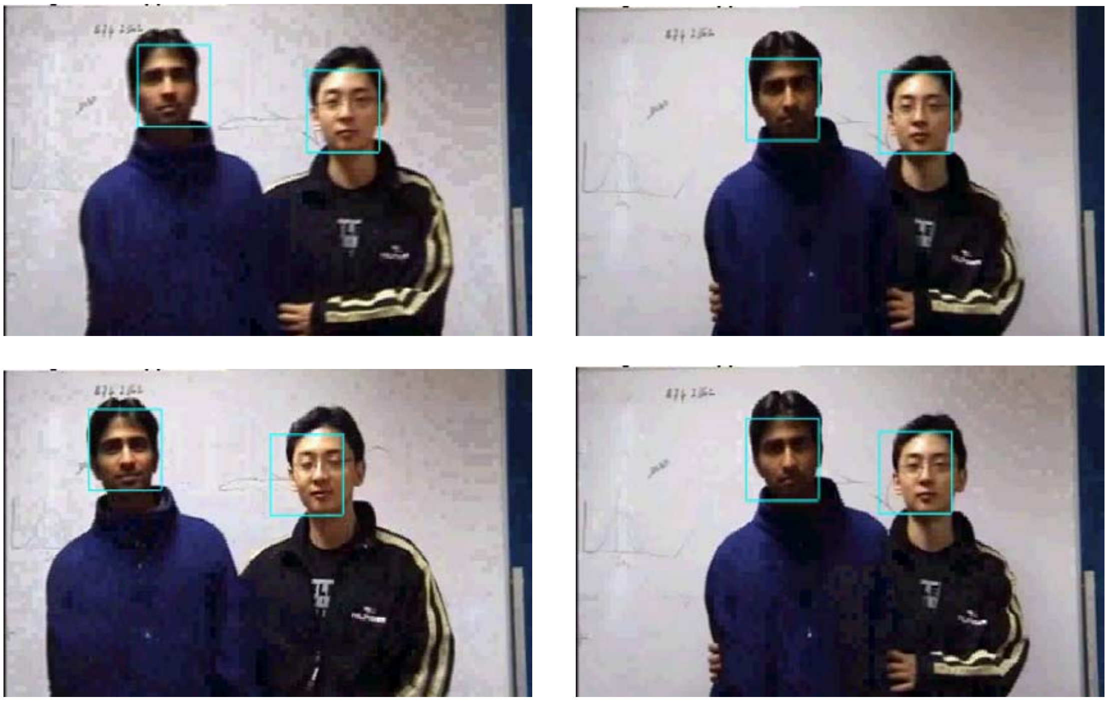
News video clip (MPEG 1 format picture size 252x288) Small or non-frontal view faces may not be detected.



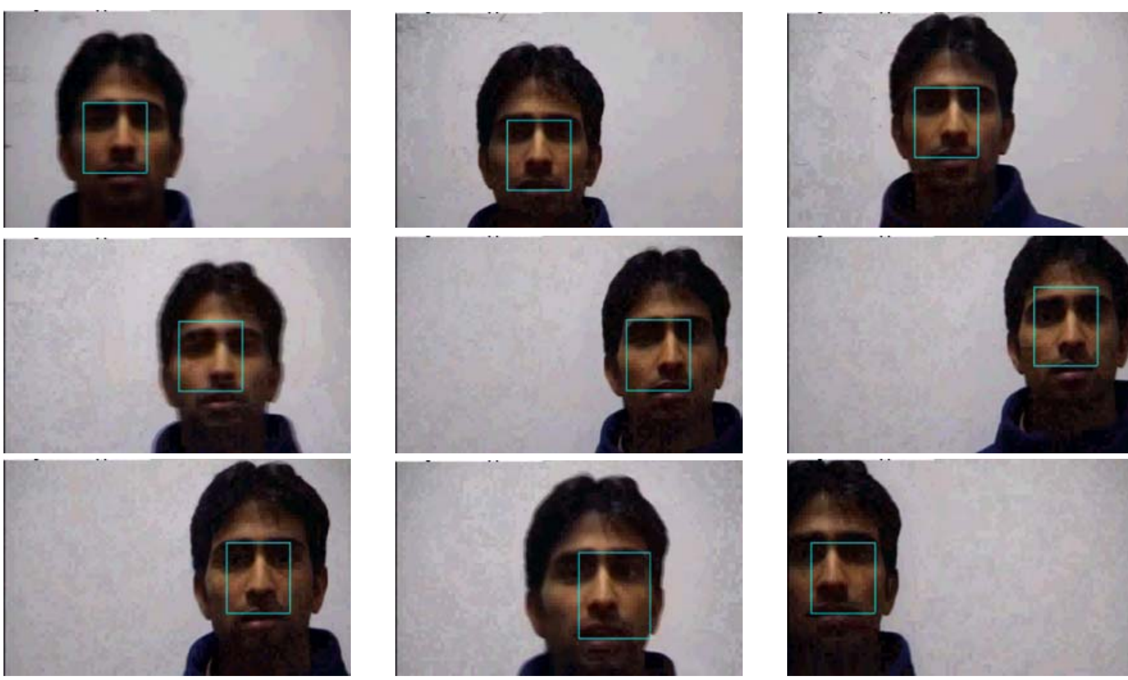
MPEG 7 Test Set (MPEG 1 format, picture size 352x288)

MPEG 7 Test Set Channel 5 news Singapore MPEG 7 Test Set

Figure 3.17 Screen shots of face detection results from the news and advertisement clips



Two persons in the lab environment (MPEG 1 format picture size 352x288)



One person in the lab environment (MPEG 1 format picture size 352x288)

Figure 3.18 screen shots for face detection results from Lab environment

MPEG 1 format picture size 353x240. In this group, false and missed detections occur.

MPEG 1 format picture size 640x512 .One false detection occurs in this group

Figure 3.19 screen shots for face detection from some movie clips

Chapter 4

A Sensor Fusion Based Face Tracker in Compressed Domain

This chapter mainly describes a sensor fusion based face tracking system in compressed domain for content based video indexing. The main contribution of this chapter is in proposing Kalman filter based two-stage fusion architectures to fuse a face detector and two object trackers to get optimal fused results.

Owning to the complexity of the videos and variations of the human faces, the face detector and face tracker may not work too well independently. The two can therefore be merged to mutually overcome limitations specific to either of them. In particular, a corrective feedback from good face detection results (for instance) can prevent error accumulation. Conversely, a face detector's performance can be improved with the help of good tracking support. However, finding a method that can combine two components (face detector and object tracker) to obtain a joint estimate is non-trivial. It seems logical to take the sensor fusion approach to fusing the measurements. Kalman filtering provides a good mechanism to fuse noisy measurements from sensors by minimizing the mean square error (MSE). We present a novel sensor fusion based face tracking approach in the compressed domain. A Kalman filter based two-stage fusion framework is proposed to fuse three individual sensors, coming from two object trackers (one color based object track and one motion vector based object tracker from the object tracker presented in [39]) and one face detector (a compressed domain neural network based face detector, presented in chapter 3). The fusion approach is adopted in order to build a system that is more accurate than any of the individual sensors. Four sensor fusion architectures are tried out to compare the effectiveness of fusion frameworks in tracking. The tracking results show that the fused results are definitely better than those of any individual sensor and mean of them as well. Also the hierarchical two-stage state vector fuser appears to be the best fusing option among the four proposed.

4.1 System design

From the review of current face detection and face tracking approach in literature, it can be seen that, for tracking face in the unconstrained context of real-world videos, the face detector may not perform robustly in the absence of utilization of temporal correlation. On the other hand, current face trackers in the absence of robust model of face variations may also fail in tracking faces in real videos. Fusing these two approaches together therefore is a good way to improve face tracking, but is non-trivial.

Our system takes a sensor fusion approach to tracking faces. It does not rely on any single sensor, as it could be inaccurate. On the other hand, we proceed on the premise that multiple sensors (one of face detection and two of tracking) with individual inaccuracies can produce a good overall result with both the mechanism that face detector helps object trackers and object trackers help face detector, and consideration of both temporal correlation indicated by the tracker from the object tracker and face spatial distribution coming from the face detector. In addition, our approach uses a positive feedback loop where the output of the previous tracking result helps the next one. Moreover, the face detection and object tracking components used here are completely implemented in compressed domain.

In order to grasp both the spatial and temporal correlations in videos, there are two object trackers and one face detector used as components for fusion in this work. The two visual measurements from object tracking module are the motion vector based tracker and the color based tracker. One compressed domain neural network based face detector is used for the face detection module. Brief information about these three components will be provided in section 4.1.1. The reason why we use object trackers instead of face trackers is that face detectors already include the functionality of modeling the spatial distribution of the face and it is sufficient for the object trackers to model the temporal distribution of the face dynamics alone.

Kalman filtering is an optimal linear estimator based on an iterative and recursive process and provides a good mechanism to fuse noisy measurements from sensors by minimizing the mean square error (MSE). We prefer to take the sensor fusion ([30]) approach for combining the face detector and the face trackers. By using sensor fusion approach, we treat three individual components as three sensors and fused these three noisy visual measures from sensors to get optimal combined results. By this means, unlike in [32], we use Kalman filter not only to estimate and predict in the video sequences, but also to fuse visual measurements.

Although fusion of three sensors using Kalman filter can be done using several configurations, we do it in two stages and try four different combinations of it (Figure 4.4 (a),(b),(c), and (d)), which will be explained later.

We compare the four two-stage sensor fusion hybrid/hierarchical architectures with partial feedback (figure 4.4(e)). The advantage of this approach is that the following problems can be overcome:

1. Associating the correct track to the right face in a multiple face scenario
2. Correction of tracker by feedback from the face detector.
3. Trackers provide output even when face detector misses detection due to the presence of non-frontal view faces.

There are other two significant feature differences of our system as compared to those of [32] and [34] - one, we use DCT color information in the object tracking module (not only skin color) and two, we also try to account for the change of size of the faces being tracked as a result of camera zoom or object moving closer/further relative to the camera.

4.1.1 System components

There are two main components: face detection (chapter 3) and object tracking ([39]) mentioned earlier. The face detector module (FD) gives one sensor measurement while the object tracker module has two trackers: color based tracker and motion vector based object tracker, and gives two sensor measurements. They are completely implemented in the compressed domain.

4.1.1.1 I frame based face detector

We use the compressed domain neural network based face detector discussed in chapter 3 as the I frame based face detector. For detailed information, please refer to chapter 3.

4.1.1.2 Color based object tracker

We use the work done in [39] for the color based tracker. Here we just provide a brief introduction. For detailed information, please refer to [39].

At the starting I frame, the DC value and first eight AC values (since higher frequency AC values beyond the first eight are small or zero usually) are extracted for both *Cr* and *Cb*. Histograms with sixteen bins (the number was arrived at after experimentation) are made for each of the eight DCT values for both *Cr* and *Cb*. These form the reference histograms. Now at every following I frame the best match for this initial object region is found within a search region around the area tracked by the motion vector tracking (figure 4.1). For this, a window of the size of the object is moved two pixels at a time over the entire search area, and the candidate histograms like the ones made for the reference area are made. Obviously this requires us to reconstruct the DCT coefficients for each candidate area that does not fall on the grid area of 8x8 pixels in which compressed data is available. For reconstruction of DCT values, the algorithm presented in [37] is used. Finding the best match involves finding the minimum value of DiffSum:

$$DiffSum = \sum_{n \text{ in } [1,9]} Wt[n] \left(|HDiffCr| + |HDiffCb| \right) \quad (1)$$

where *HDiffCr* and *HDiffCb* are the histogram bin differences for *Cr* and *Cb* values respectively; *Wt*[*n*] is the set of weights used {.4, .1, .1, .1, .1, .05, .05, .05, .05} for one DC plus eight AC. The weights are chosen in such a way that the DC value, which most prominently conveys colour information is given maximum weightage, followed by lower frequency AC values, which convey coarse texture or shape information ([40]), in decreasing order of importance in the matching algorithm. Though a lower weightage is given to the AC values than the DC value as in eq. 1, it is noteworthy that AC information proves very useful in distinguishing between two objects with the same colour, but different shapes or textures.

Size-change of objects (because of camera zoom or otherwise) is also taken care of by comparing the minimum values of *DiffSum* with the window sizes smaller and larger on all sides by one block width, with that of the starting window size. In case of a zoom-in, the minimum value of *DiffSum* of a larger window is lesser than that for a smaller window, and vice versa, in case of a zoom-out.

4.1.1.3 Motion vector based object tracker

We use the work done in [39] for the motion vector based tracker, of which we just give a brief introduction. For detailed information, please refer to [39].

Tracking is done within the Group of Pictures (GOP) is done using forward motion vectors of P and B pictures. In the compressed domain, the I frame of a GOP is followed by the B frames of the previous GOP which are backward predicted from this I frame. Since the compressed data of an I frame, does not have motion vectors, the backward motion vectors of the last B frame are used to track the object into the I frame of a following GOP. The tracking algorithm is briefly given below.

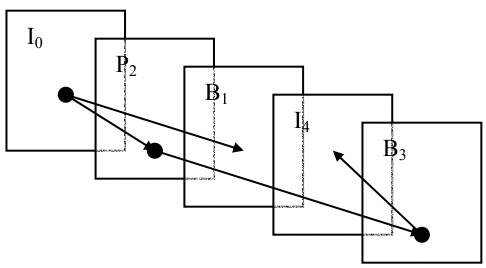


Figure 4.1: Decode order is I_0 P_2 B_1 I_4 B_3; Display order is I_0 B_1 P_2 B_3 I_4

1.User chooses object in the I frame by drawing a rectangle around it.

2. For every predicted frame (P or B), the object region is tracked from the respective reference frame (I or P) as

$$R_p \leftarrow Translate\ (R_p\ ,\ Mode\ (\ MV\ (R_c)\ )\ ) \tag{2}$$

R_p Is the pixel domain rectangle around the object; Rc is the equivalent compressed domain rectangle; $MV(R_c)$ is the set of motion vectors available in R_c ; *Mode( )* give the mode of the motion vectors; *Translate*(R, v) translates the pixel domain rectangular region R of the reference frame by the displacement vector v.

3. To cross the GOP boundary, the image region on the new I frame is obtained again using eq. 2, with the difference that the backward motion vector mode of the last B frame is used, and the B frame pixel domain rectangle itself is treated as the reference rectangle.

4.1.2 Sensor fusion

Sensor fusion is a broad field of applied computer science and electrical engineering. One of the most useful tools of fusing data from measurement sensors is the Kalman filtering approach which has been used in several such applications since its introduction in 1960 by R. E. Kalman.

Kalman filter works as a predictor-corrector mechanism for a system with unknown state variables, and noisy measurements, minimizing the MSE of the combined measurements. More, it can also be used for fusion purpose by measurement fusion or state vector fusion. This is quite appropriate for our case of sensor fusion. We propose the use of a hybrid model of Kalman filter based data fusion with partial feedback (figure 4.4). Before we explain the model itself, a brief introduction to Kalman filtering is in order.

4.1.2.1 Kalman filter

The Kalman filter tries to estimate the state vector $x(k)$ of a discrete-time dynamical system that is governed by the process equation (3)

$$x(k+1) = \Phi(k)x(k) + w(k) \tag{3}$$

with measurements $z(k)$ at time instant k given by

$$z(k) = H(k)x(k) + v(k) \tag{4}$$

where, $x(k)$ is state vector at time k, which consists of all parameters that are estimated by the filter(e.g. position, velocity); $\Phi(k)$ is the state transition matrix, $H(k)$ is the measurement or observation matrix; $w(k)$ and $v(k)$ represent the process and measurement noise respectively; $w(k)$ and $v(k)$ are assumed to be independent of each other, white, and with normal probability distributions N(0, Q) and N(0, R) respectively. Q is the process noise covariance while R is the measurement noise covariance.

By time update, $\hat{x}(k|k-1)$ (predicted state vector) can be obtained

$$E(x(k|k-1)) = \hat{x}(k|k-1) = \Phi\hat{x}(k-1|k-1) \tag{5}$$

$$\begin{aligned} &E(e(k|k-1)e(k|k-1)^T) \\ &= E((x(k|k-1) - \hat{x}(k|k-1))(x(k|k-1) - \hat{x}(k|k-1))^T) \\ &= P(k|k-1) = \Phi P(k-1|k-1)\Phi^T + \Gamma(k)Q\Gamma^T(k) \end{aligned} \tag{6}$$

By measurement update, $\hat{x}(k|k)$ (estimated state-vector) can be obtained:

$$E(x(k|k)) = \hat{x}(k|k) = \hat{x}(k|k-1) + K(z(k) - H\hat{x}(k|k-1)) \tag{7}$$

$$\begin{aligned} &E(e(k|k)e(k|k)^T) = E((x(k|k) - \hat{x}(k|k))(x(k|k) - \hat{x}(k|k))^T) \\ &= P(k|k) = (I - KH)P(k|k-1)(I - KH)^T + KRK^T \end{aligned} \tag{8}$$

where $K = P(k \mid k-1)H^T(HP(k \mid k-1)H + R)^{-1}$ (9)

In our work we use Kalman filters for the purpose of estimation in non-intracoded frames as well as for sensor fusion in I frames, which is different from how it is used in case of [34] where it is employed only for estimation.

4.1.2.2 Sensor fusion using Kalman filters

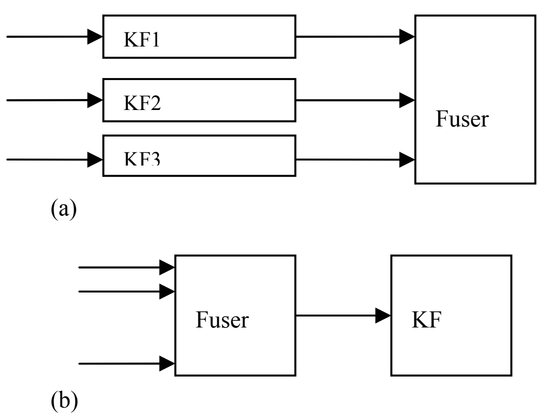


Figure 4.2: Kalman filter based multi-sensor data fusion by (a) State-vector fusion, (b) Measurement fusion

There are two standard methods for data fusion using the Kalman filter (figures 4.2.(a) and 4.2.(b)), namely, state vector (SVF) fusion and measurement fusion (MF). In measurement fusion two or more measurements are combined together according to their weights (if any) and fed into one Kalman filter to get the final estimate. In the state-vector fusion method individual state-vectors are estimated by one

Kalman filter each, and the output of each is fused to get the final estimate. The formalisms of the two fusions will be discussed in the later section. A detailed analysis and comparison can be found in [41].

In our specific domain, we perform hybrid fusion with semi-feedback method as shown in Fig 4.4 (a, b, c and d). We explain the four fusion frameworks next.

4.1.3 The sensor fusion architectures

The measurements of three sensors we used in compressed domain are not always available in each frame (The measurement from the motion vector based object tracker exists in each frame, but the measurements of the face detector and the color based object tracker only exist in I frames). Therefore, as shown in figure 1, in our framework, Kalman filter based sensor fusion approach is used in each I frame to fuse three noisy sensor measurements to get optimal results, and Kalman filter based estimation is applied to smooth the measurement of motion vector based object tracker in non-intracoded frames.

There are three components that need to be fused in the I frames in our system, namely the face detector, motion vector based object tracker and color based object tracker. We propose a two-stage fusion architecture: first, motion vector based object and color based object tracker are fused to get robust tracking result; secondly, the fused tracking result and face detector result are fused to get the final tracking result. By utilizing measurement and state vector fusion method of Kalman filter, we employ four different frameworks to do face tracking, two of them being hierarchical ones and other two being hybrid ones, each having two stages, as shown in figure 4.4. The hierarchical framework employing state vector fusion at both the levels is referred to as SSF (Figure. 4.4(a)); the hierarchical fusion architecture employing measurement fusion at both levels is called MMF (Figure. 4.4(b)); the hybrid architecture using measurement fusion in the first stage for fusion and state vector fusion in the second stage is called MSF (Figure. 4.4 (c)); and finally the hybrid architecture employing state vector fusion at the first stage and measurement fusion for the second stage is referred to as SMF(Figure. 4.4 (d)).

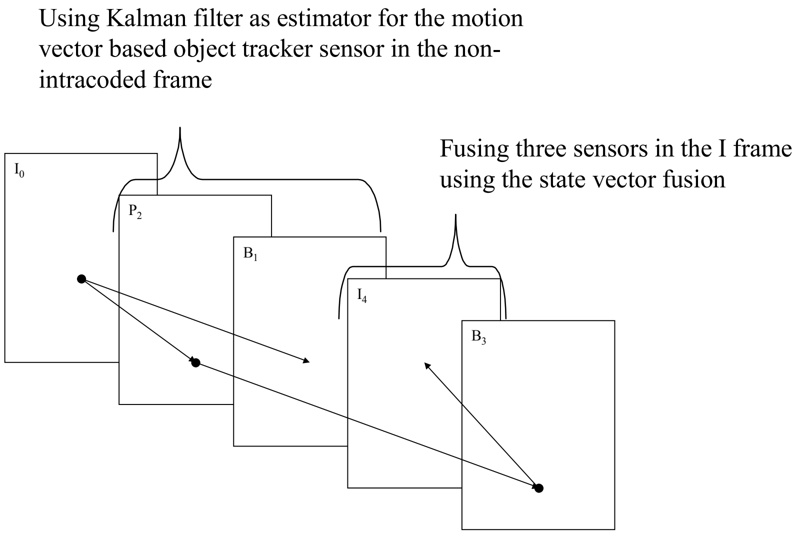


Figure 4.3. Face tracking framework in the compressed video

By using this two-stage fusion architecture, there are two advantages.

1. In the multi-face scenario: due to the lack of temporal information of faces, the face detector has difficulty to know which of detected face in the incoming I frame belongs to which of the current tracker. However, in this architecture, with the help of the first stage of object tracker sensor fusion result, it is possible to assign each face detected (by the face detector) in the incoming I frame to the one being tracked by the tracking sensors (shown in the figure 4.4 as face allocator, FA). This is a problem that can not be resolved by the face detector itself. The second stage then generates the best estimate from all the sensors.

2. The final fused results can also be fed back to the tracking sensors as the correction mechanism. It helps to stop the error accumulations of object trackers. Since it just feeds back to the object tracker only, and not the face detector, we call this partial feedback.

In the next section, we explain the detailed configurations of the four architectures.

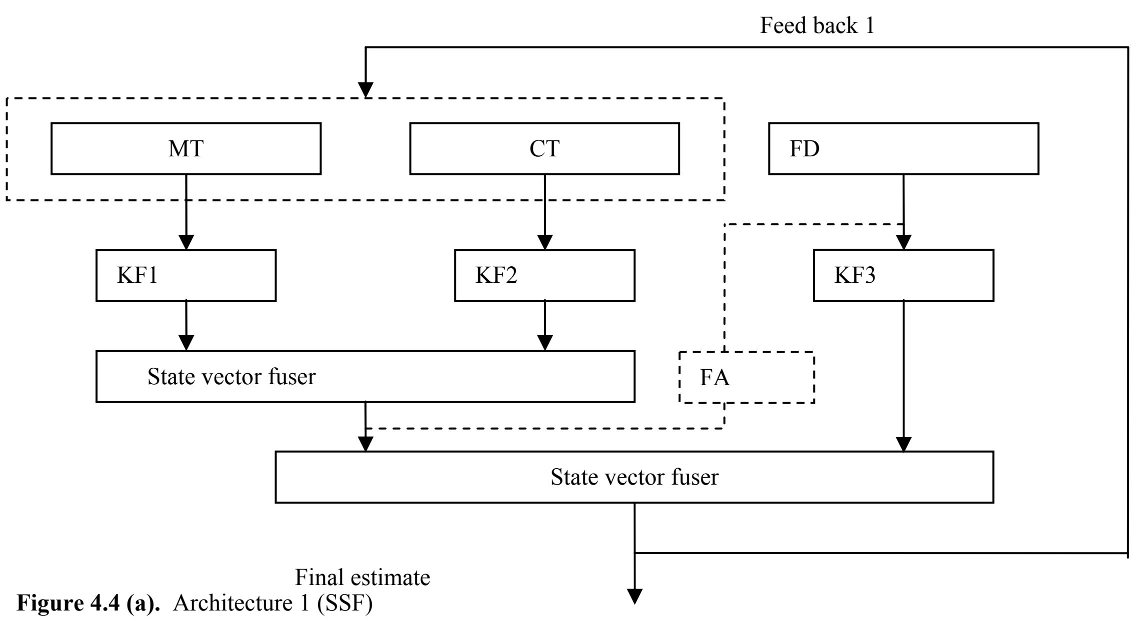


Figure 4.4 (a). Architecture 1 (SSF)
MT,CT, FD and KF stand for motion vector based tracker, color based tracker, face detector and Kalman filter respectively. FA stands for Face allocator.
In the stage 1, utilizing two Kalman filters (KF1 and KF2), state vector fusion is used to fuse the two object trackers (MT and CT). In the stage 2: state vector fusion is also used to fuse the stage 1 output and the face detector output (KF3).
Feedback 1:The final fused results are fed back to the tracking sensors as the correction mechanism. It helps to stop the error accumulation of object trackers
FA: with the help of the first stage of object tracker sensor fusion result, it allows us to assign each face detected (by the face detector) in the incoming I frame to the one being tracked by the tracking sensors.

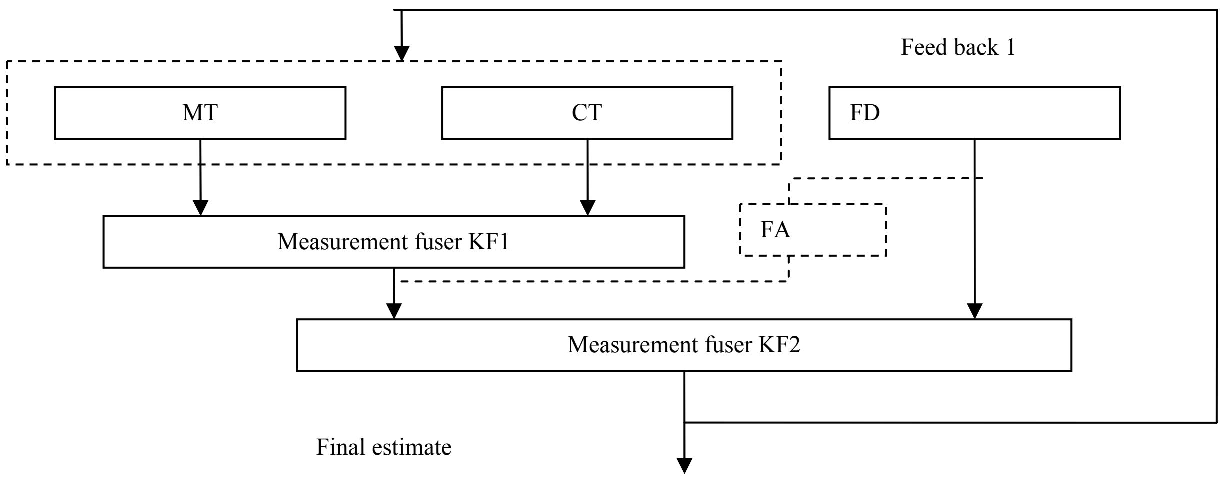


Figure 4.4 (b): Architecture 2 (MMF)
MT,CT, FD and KF stand for motion vector based tracker, color based tracker, face detector and Kalman filter respectively. FA stands for Face allocator.
In the stage 1, utilizing one the Kalman filter (KF1), measurement fusion is used to fuse the two object trackers (MT and CT). In the stage 2: utilizing the Kalman filter KF2, measurement fusion is used to fuse the stage 1 output and the face detector output).
Feedback 1:The final fused results are fed back to the tracking sensors as the correction mechanism. It helps to stop the error accumulation of object trackers
FA: with the help of the first stage of object tracker sensor fusion result, it allows us to assign each face detected (by the face detector) in the incoming I frame to the one being tracked by the tracking sensors.

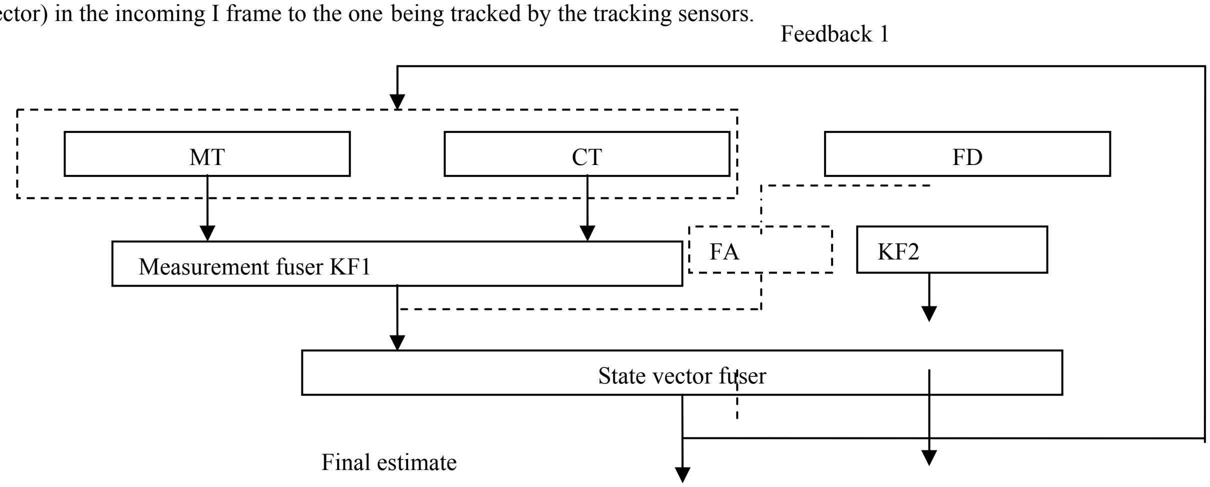


Figure 4.4 (c): Architecture 3 (MSF)
MT,CT, FD and KF stand for motion vector based tracker, color based tracker, face detector and Kalman filter respectively. FA stands for Face allocator.
In the stage 1, utilizing the Kalman filter (KF1), measurement fusion is used to fuse the two object trackers (MT and CT). In the stage 2: state vector fusion is also used to fuse the stage 1 output and the face detector output (KF2).
Feedback 1:The final fused results are fed back to the tracking sensors as the correction mechanism. It helps to stop the error accumulation of object trackers.
FA: with the help of the first stage of object tracker sensor fusion result, it allows us to assign each face detected (by the face detector) in the incoming I frame to the one being tracked by the tracking sensors.

4.1.3.1 Sensor fusion Architecture 1: SMF

Stage one: As shown in figure 4.4 (d), two measurements, which come from normalized color histogram based matching and motion vector based tracking, are fed into their respective Kalman filters KF1 and KF2. Then the estimated tracker results (state-vector) are fused using state-vector fusion method in the state vector (SV) fuser to obtain the tracking estimate.

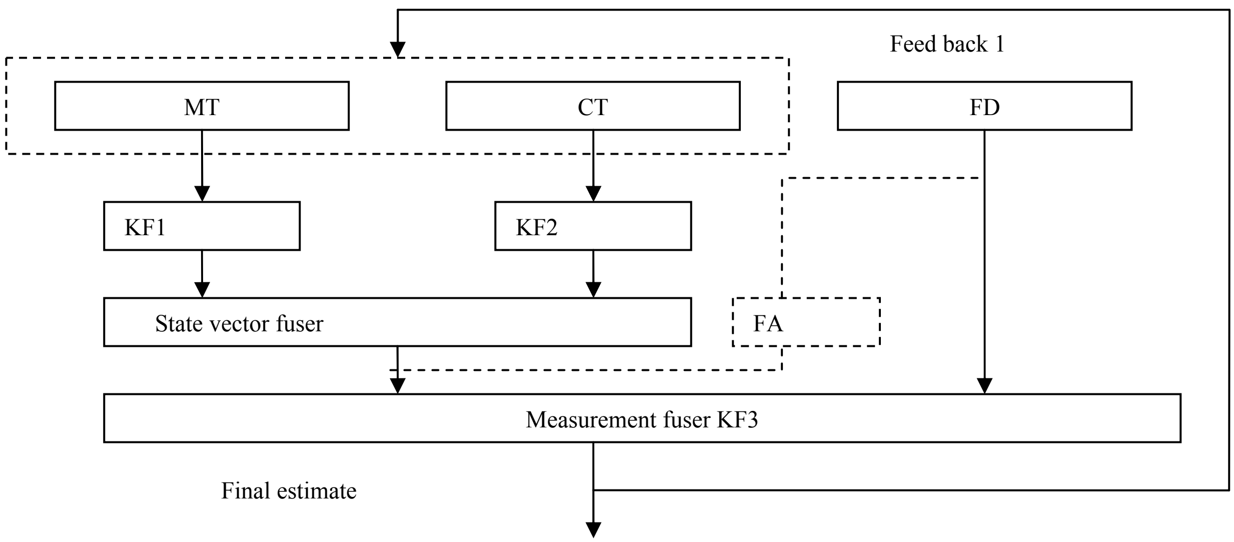


Figure 4.4 (d): Architecture 4 (SMF)
MT,CT, FD and KF stand for motion vector based tracker, color based tracker, face detector and Kalman filter respectively. FA stands for Face allocator.
In the stage 1, utilizing two Kalman filters (KF1 and KF2), state vector fusion is used to fuse the two object trackers (MT and CT). In the stage 2, utilizing the Kalman filter KF3, measurement fusion is used to fuse the stage 1 output and the face detector output.
Feedback 1:The final fused results are fed back to the tracking sensors as the correction mechanism. It helps to stop the error accumulation of object trackers
FA: with the help of the first stage of object tracker sensor fusion result, it allows us to assign each face detected (by the face detector) in the incoming I frame to the one being tracked by the tracking sensors.

Stage two: Face Detector measurement and the estimation of face tracker obtained in step 1 are fused using the measurement fusion method in Kalman filter K3.

Feedback: The output from the second step is fed back as the measurement for the tracking module. As explained before, the feedback is considered to be partial because it is not fed to the face detection module.

Face allocator (FA): As shown in figure 4.4 (d), with the help of output of state vector fuser from two object trackers, we can assign each face detected (by the face detector) in the incoming I frame to the one being tracked by the tracking sensors.

4.1.3.1.1 State vector fusion in stage one of SMF

The use of state-vector fusion method in the first step to fuse motion vector measurement and color histogram measurement results in lower computational expense and allows the advantage of fault tolerance ([41]), which in our case can be useful in situations where the motion vector based tracking results are way off the mark because of unreliable motion vectors. It is worth noting that if two measurements to be fused are independent, then measurement fusion and state vector fusion are equivalent ([42]).

By using maximum likelihood as a fusion strategy, the best estimated fused data is derived from equation (10) for stage one.

$$\hat{x}_F(k \mid k) = \frac{\sum_{i=1}^{n} \hat{P}_{KFi}^{-1}(k \mid k)\hat{x}_{KFi}}{\sum_{i=1}^{n} \hat{P}_{KFi}^{-1}(k \mid k)} \tag{10}$$

where, $\hat{x}_F(k \mid k)$ is the best estimated fused data in time k, $\hat{P}_{KFi}$ is estimated state vector's covariance matrix of Kalman filter ***i*** in time ***k***, and $\hat{x}_{KFi}$ is the estimated state vector in Kalman filter ***i***.

Following the equation (10), the best estimated fusion data of motion vector Kalman filter based tracker and color histogram Kalman filter based tracker can be written in another form as shown in equation (11) and the covariance matrix obtained using equation (12). (Here, Kalman filter1 and Kalman filter 2 represent motion vector tracker and color histogram tracker respectively.)

$$\hat{x}_F(k \mid k) = \hat{x}_{KF1} + \hat{P}_{KF1}(k \mid k)\hat{P}_E^{-1}(k \mid k)[\hat{x}_{KF2} - \hat{x}_{KF1}] \tag{11}$$

$$\hat{P}_F(k \mid k) = \hat{P}_{KF1}(k \mid k) - \hat{P}_{KF1}\hat{P}_E^{-1}(j \mid j)\hat{P}_{KF1}(k \mid k)^T \tag{12}$$

where

$$\hat{P}_E(k \mid k) = \hat{P}_{KF1}(k \mid k) + \hat{P}_{KF2}(k \mid k) \tag{13}$$

4.1.3.1.2 Measurement fusion in stage two of SMF

A better overall estimation performance can be generally obtained by measurement fusion method ([41]). The measurement fusion method is used to obtain the final fusion estimation in the SMF hybrid fusion architecture. By using measurement fusion, equation 4 can be replaced by equation 14. The new combined measurement $z(k)$ is composed of $z_{FD}(k)$ and $z_{FT}(k)$ (with *FD* implying face detector and *FT* representing face tracker). $z_{FD}(k)$ is the face detector result at time k. and $z_{FT}(k)$ is equal to $\hat{x}_F(k \mid k)$ (the best estimated fusion data of motion vector Kalman filter based tracker and color histogram Kalman filter based tracker) in equation 11 , which is already available from stage one.

$$z(k) = [z_{FD}^T(k), z_{FT}^T(k)]^T = \begin{bmatrix} H_{FD}(k) \\ H_{FT}(k) \end{bmatrix} x(k) + \begin{bmatrix} v_{FD}(k) \\ v_{FT}(k) \end{bmatrix} \tag{14}$$

where

$$H_{FD} = H_{FT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (15)$$

4.1.3.2 Sensor fusion architecture 2: MSF

Stage one: As shown in figure 4.4 (c), In the stage one, two measurements, which come from normalized color histogram based matching and motion vector based tracking are fused by measurement fusion in Kalman filter KF1. In KF1, these two measurements are combined to be one measurement $z(k)$ and fed into a Kalman filters KF1. Then the estimated tracker results (state-vector) are obtained.

Stage two: As shown in Figure 4.4 (c), in the stage two, the face detector measurement and the estimation of face tracker obtained in step 1 are fused using the state vector fusion method.

The mechanism of feedback and face allocator (FA) is same as that of SMF architecture.

4.1.3.2.1 Measurement fusion in stage one of MSF

The measurement fusion method is used to obtain the fist stage fusion estimation in the MSF hybrid fusion architecture. By using measurement fusion, equation 4 can be replaced by equation 15. The new combined measurement $z(k)$ is composed of $z_{CT}(k)$ and $z_{MT}(k)$ (with CT implying Color based tracker and MT representing motion vector based Tracker). $z_{CT}(k)$ is the measurement from color based tracker at time k while $z_{MT}(k)$ is the measurement coming from motion vector based tracker. The combined new measurement $z(k)$ is fed into Kalman filter KF1 to get optimal fused results.

$$z(k) = [z_{FD}^T(k), z_{FT}^T(k)]^T = \begin{bmatrix} H_{FD}(k) \\ H_{FT}(k) \end{bmatrix} x(k) + \begin{bmatrix} v_{FD}(k) \\ v_{FT}(k) \end{bmatrix} \tag{15}$$

where

$$H_{MT} = H_{CT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{16}$$

4.1.3.2.2 State vector fusion in stage two of MSF

The use of state-vector fusion method in the second step to fuse face detector measurement and tracker measurement results in lower computational expense and allows the advantage of fault tolerance ([41]), By using maximum likelihood as a fusion strategy, the best estimated fused data is derived from equation (10) for stage one.

Following the equation (10), as shown in Figure 4.4 (c), the best estimated fusion data of face detector Kalman filter KF2 and tracker Kalman filter KF1 can be written in another form as shown in equation (17) and the covariance matrix obtained using equation (18). (Here, Kalman filter1 and Kalman filter 2 represent face detector tracker and fused tracker in stage one respectively.)

$$\hat{x}_F(k \mid k) = \hat{x}_{KF1} + \hat{P}_{KF1}(k \mid k)\hat{P}_E^{-1}(k \mid k)[\hat{x}_{KF2} - \hat{x}_{KF1}] \tag{17}$$

$$\hat{P}_F(k \mid k) = \hat{P}_{KF1}(k \mid k) - \hat{P}_{KF1}\hat{P}_E^{-1}(j \mid j)\hat{P}_{KF1}(k \mid k)^T \quad (18)$$

where

$$\hat{P}_E(k \mid k) = \hat{P}_{KF1}(k \mid k) + \hat{P}_{KF2}(k \mid k)$$

4.1.3.3 Sensor fusion architecture 3: SSF

Stage one: As shown in Figure 4.4 (a), Two measurements, which come from normalized color histogram based matching and motion vector based tracking, are fed into their respective Kalman filters KF1 and KF2. Then the estimated tracker results (state-vector) are fused using state-vector fusion method in the state vector (SV) fuser to obtain the tracking estimate.

Stage two: As shown in Figure 4.4 (a), in the stage two, face Detector measurement and the estimation of face tracker obtained in step 1 are fused using the state vector fusion method.

The mechanism of feedback and face allocator (FA) is same as that of SMF architecture.

4.1.3.3.1 State vector fusion in stage one of SSF

By using maximum likelihood as a fusion strategy, the best estimated fused data is derived from equation (10) for stage one. Following the equation (10), the best estimated fusion data of motion vector Kalman filter based tracker and color histogram Kalman filter based tracker can be written in another form as shown in equation (19) and the covariance matrix obtained using equation (20). (Here, KF1 and KF 2 represent motion vector tracker and color histogram tracker respectively.)

$$\hat{x}_F(k \mid k) = \hat{x}_{KF1} + \hat{P}_{KF1}(k \mid k)\hat{P}_E^{-1}(k \mid k)[\hat{x}_{KF2} - \hat{x}_{KF1}] \tag{19}$$

$$\hat{P}_F(k \mid k) = \hat{P}_{KF1}(k \mid k) - \hat{P}_{KF1}\hat{P}_E^{-1}(j \mid j)\hat{P}_{KF1}(k \mid k)^T \tag{20}$$

where

$$\hat{P}_E(k \mid k) = \hat{P}_{KF1}(k \mid k) + \hat{P}_{KF2}(k \mid k)$$

4.1.3.3.2 State vector fusion in stage two of SSF

Same as stage one, by using maximum likelihood as a fusion strategy, the best estimated fused data is derived from equation (10) for stage two.

Following the equation (10), as shown in Figure 4.4 (a), the best estimated fusion data of face detector Kalman filter KF3 and outputs of state vector fuser in stage one can be written in another form as shown in equation (21) and the covariance matrix obtained using equation (22). (Here, KF3 and SVF represent face detector tracker and outputs from stage one respectively.)

$$\hat{x}_F(k \mid k) = \hat{x}_{KF3} + \hat{P}_{KF3}(k \mid k)\hat{P}_E^{-1}(k \mid k)[\hat{x}_{SVF} - \hat{x}_{KF3}] \tag{21}$$

$$\hat{P}_F(k \mid k) = \hat{P}_{KF3}(k \mid k) - \hat{P}_{KF3}\hat{P}_E^{-1}(j \mid j)\hat{P}_{KF3}(k \mid k)^T \tag{22}$$

where

$$\hat{P}_E(k \mid k) = \hat{P}_{KF3}(k \mid k) + \hat{P}_{SVF}(k \mid k)$$

4.1.3.4 Sensor fusion architecture 4: MMF

Stage one: As shown in Figure 4.4(b), in stage one, two measurements, which come from normalized color histogram based matching and motion vector based tracking are fused by measurement fusion in Kalman filter KF1. In KF1, these two measurements are combined to be one measurement $z(k)$ and fed into Kalman filters KF1. Then the estimated tracker results (state-vector) are obtained.

Stage two: Face Detector measurement and the estimation of face tracker obtained in step 1 are fused using the measurement fusion method in Kalman filter KF2.

The mechanism of feedback and face allocator (FA) is same as that of SMF architecture.

4.1.3.4.1 Measurement fusion in stage one of MMF

The measurement fusion method is used here to obtain the fusion estimation of two trackers in the MMF fusion architecture. By using measurement fusion, equation 4 can be replaced by equation 23.

Same as the frist stage of MSF, the new combined measurement $z(k)$ is composed of $z_{CT}(k)$ and $z_{MT}(k)$ (with CT implying Color based tracker and MT representing motion vector based Tracker). $z_{CT}(k)$ is the measurement from color based tracker at time k while $z_{MT}(k)$ is the measurement coming from motion vector based tracker. The combined new measurement $z(k)$ is fed into Kalman filter KF1 to get optimal fused results.

$$z(k) = [z_{FD}^T(k), z_{FT}^T(k)]^T = \begin{bmatrix} H_{FD}(k) \\ H_{FT}(k) \end{bmatrix} x(k) + \begin{bmatrix} v_{FD}(k) \\ v_{FT}(k) \end{bmatrix} \tag{23}$$

where

$$H_{MT} = H_{CT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{24}$$

4.1.3.4.2 Measurement fusion in stage two of MMF

A better overall estimation performance can be generally obtained by measurement fusion method ([41]). The measurement fusion method is used to obtain the final fusion estimation in the MMF fusion architecture same as SMF architecture. By using measurement fusion, equation 4 can be replaced by equation 14. The new combined measurement $z(k)$ is composed of $z_{FD}(k)$ and $z_{FT}(k)$ (with *FD* implying face detector and *FT* representing face tracker). $z_{FD}(k)$ is the face detector result at time k. and $z_{FT}(k)$ is equal to $\hat{x}_F(k \mid k)$ (the best estimated fusion data of motion vector Kalman filter based tracker and color histogram Kalman filter based tracker) in equation 11 , which is already available from stage one.

$$z(k) = [z_{FD}^T(k), z_{FT}^T(k)]^T = \begin{bmatrix} H_{FD}(k) \\ H_{FT}(k) \end{bmatrix} x(k) + \begin{bmatrix} v_{FD}(k) \\ v_{FT}(k) \end{bmatrix} \tag{25}$$

where

$$H_{FD} = H_{FT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{26}$$

4.1.4 Modeling the Face Dynamics

Human faces have mainly two types of movement in the video- pose change and translation. In addition there can be a change in size because of camera zoom. Change in position of face is mainly due to zooming and translation of human faces, since turning of head does not change the position of the face significantly. We try to take into account the change of size of face too, in addition to translation and turning. This has not been done in previous works ([26] and [32]). The change in size of face and translation of human faces are modeled as in the following:

The velocity of size change can be defined as:

$$width' = (width(k+1) - width(k))/T \tag{27}$$

where, T is the interval time. Here it is the time interval between two successive frames.

The velocity of translatory motion of the face can be defined as:

$$x' = (x(k+1) - x(k))/T \tag{28}$$

Here the velocity of zooming and translatory motion is assumed to be constant since the constant velocity model has best performance for commercial videos sampled at high frame rates ([26]). Therefore, $x(k)$ state-vector is defined as in equation (29) and other parameters in processing equation (3) are defined as in equation (30).

$$x(k) = \left[x_{centroid} \quad y_{centroid} \quad width \quad height \quad x'_{centroid} \quad y'_{centroid} \quad width' \quad height' \right]^T \tag{29}$$

$$W = [w_x \quad w_y \quad w_w \quad w_h]^T \tag{30}$$

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \Gamma = \begin{bmatrix} T/2 & 0 & 0 & 0 \\ 0 & T/2 & 0 & 0 \\ 0 & 0 & T/2 & 0 \\ 0 & 0 & 0 & T/2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.1.5 Noise Modeling

We model the noise parameters (*R, Q*) offline. Both the positions of the faces $X = [x(0) \ldots x(N)]$ and their measurements $Y = [y(0) \ldots y(N)]$ are available from training data. Therefore the process and measurement noise (*R,Q*) can be modeled using the *Maximum likelihood* (ML) algorithm by maximizing

$$\begin{aligned} L(X, Y, \theta) = \\ -\sum_{k=1}^{N} \{\log |Q| + [x(k) - \Phi x(k-1)]^T Q^{-1} [x(k) - \Phi x(k-1)]\} \\ -\sum_{k=0}^{N} \{\log |R| + [y(k) - Hx(k)]^T R^{-1} [y(k) - Hx(k)]\} + C \end{aligned} \tag{31}$$

Here θ is the parameter needing to be estimated; *N* is the number of the frames in the training data; *C* is a constant.

4.2 Experiments and results

Tracking of faces was done in each of the video clips using all the four sensor fusion architectures for a fair comparison. Figure 4.5 shows the GUI for our face tracker in DVA project.

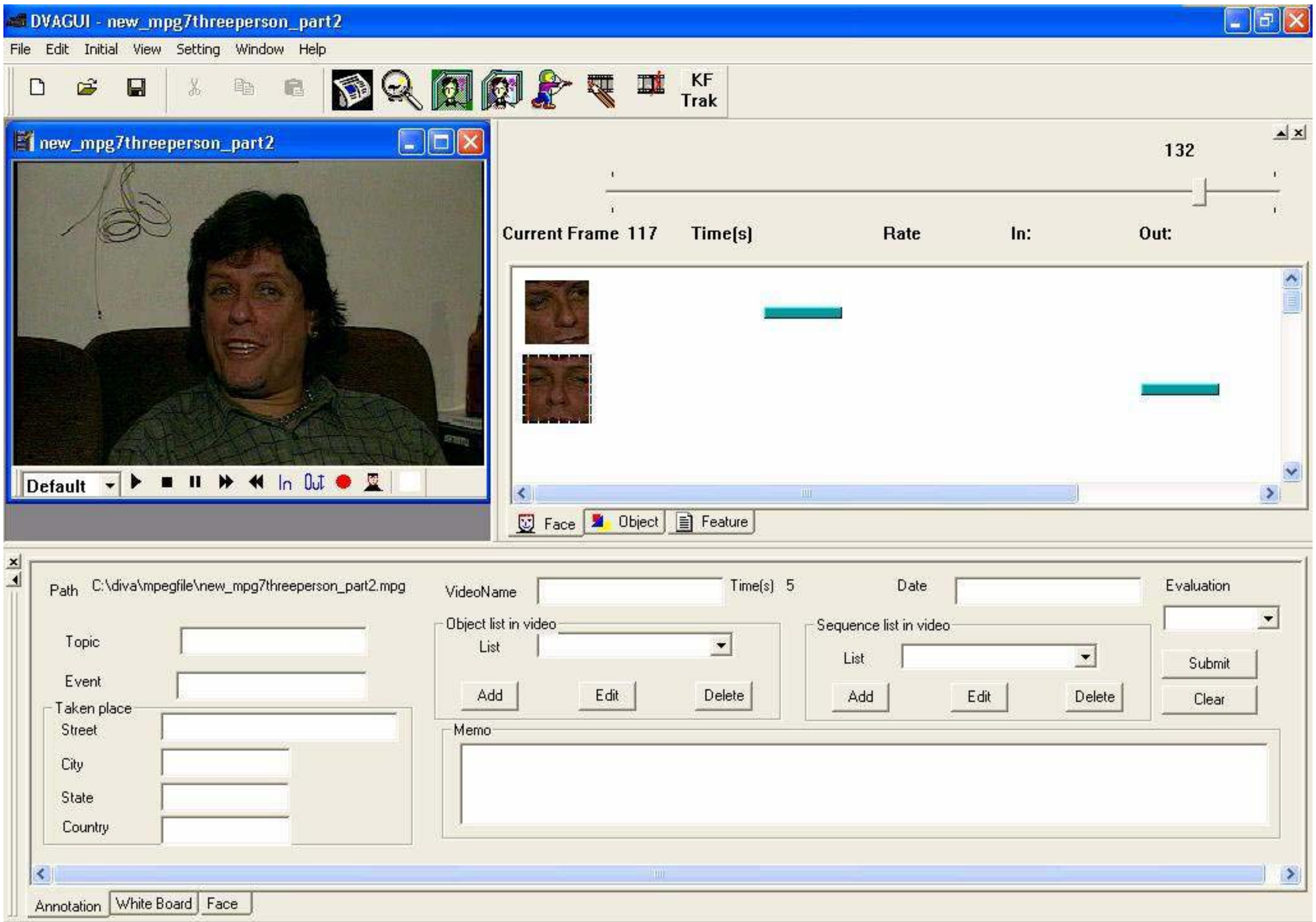


Figure 4.5 GUI for the face tracker in the DVA system
Video can be play in the left-up window. Tracking results are shown in the right-up window. The bottom window is used to annotate video and faces.

Firstly, we give one example to illustrate the effectiveness of our fusion method. In order to compare to other approaches in compressed domain, we use the video shot “Marcia”(from CNN news video), obtained from Columbia University. It is MPEG-1 compressed, with frame size of 352x240 pixels and has 556 frames, containing 38 I-frames.

Fused results in I frame Motion Vector based results Motion Vector based results

Motion Vector based results Fused results in I frame

Figure 4.6 Sample frames for video 1 "Marcia" with tracking results (1)

Figure 4.6 shows the scenario where the face detector helps object tracker in video shot of "Marcia". Note that in the first I and next I frame of Figure 4.6, the tracking results are fused by the face detector and two object trackers. In non-intracoded frames (B or P frame) of Figure 4.6, the tracking results are obtained only by motion based object tracker using Kalman filter for prediction purpose, due to the absence of face detector and color based object tracker in B or P frame. Given the low reliability of the motion vector and potential accumulation of the error, the motion vector based output may become worse gradually. However, motion vector based tracker results can be corrected by the fused result from next I frame as shown in last sample frame of Figure 4.6. According to our experiments, motion vector based object tracker normally performs well at least in the first couple of frames. For a typical MPEG video (30 frames/second, 15 frame GOP), this means about 15 frames have one I frame. Therefore, before motion vector based tracker becomes worse, the face detector has corrects it by partial feedback mechanism in our architecture.

fused results in I frame(3 sensors)   Motion Vector based results Motion Vector based results

fused results in I frame by the two object trackers only   fused results in I frame by three sensors

Figure 4.7 sample frames for video 1 "Marcia" with tracking results

Figure 4.7 shows the scenario that object tracker helps face detector in video shot of "Marcia". Since we have object trackers in I frame, in case that face detector fails in one I frame, the results got from color and motion vector based object trackers can intuitively recover the results. As shown in the last second sample frame in Figure 4.7, the face detector fails in this I frame. However, by the fused results from color and motion vector based object trackers, the result is recovered (even though the result is not robust enough).

As shown in figure 4.8, our algorithm overall performs well in video shot of "Marcia" except few B and P frames. This is mainly because that motion vector based tracker is lost before next I frame comes(the second sample frame of figure 4.8).

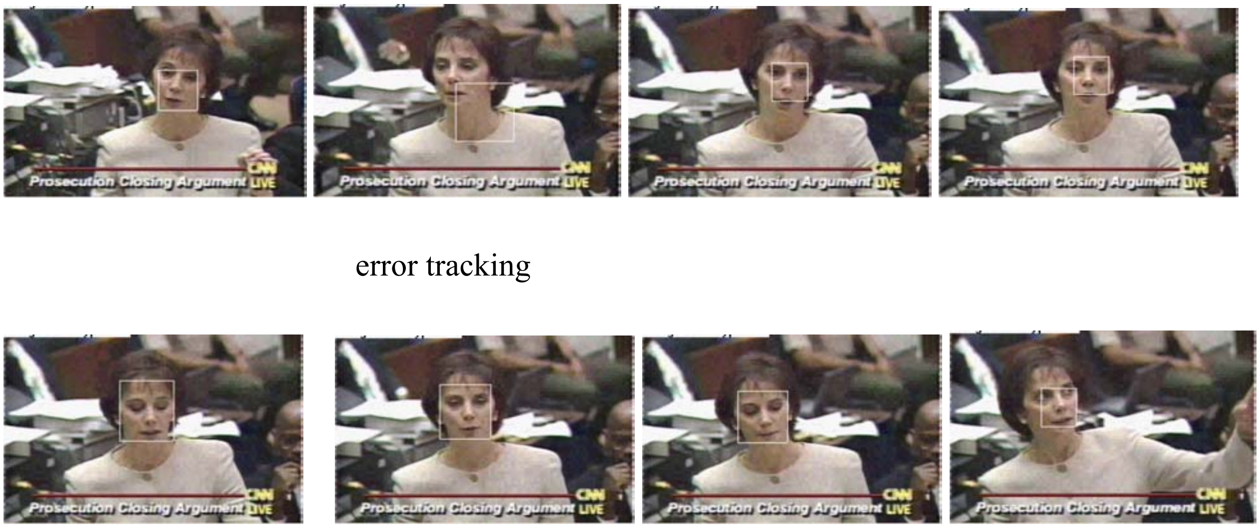


Figure 4.8 sample frames for video 1 “Marcia” with tracking results

In order to analyze the performance of the fusion results with respect to each individual sensor and to observe how the individual limitations of each sensor are overcome, we illustrate the following four graphs (Fig. 4.9 a, b, c, d) for each of the four sensor fusion architectures: Figure 4.9 a for SSF sensor fusion method, Figure 4.9 b for MMF sensor fusion method, Figure 4.9 c for SMF sensor fusion method, and Figure 4.9 d for MSF sensor fusion method. In order to demonstrate that the fused result is optimal one as compared to the simple mean of three sensors, here we also show the results of mean of the output of one face detector and two object trackers, in the graphs. The test video we used comes from MPEG 7 test clips. It is in the MPEG-1 format and has 276 frames.

The graphs (Figure 4.9) show the tracking errors of the three individual sensors, their means and fused tracking result. Tracking errors are measured as the distance between the centroid of the rectangle of the tracking results and that of the ground truth (the ground truth was found manually by drawing a rectangle around the face(s) of interest to be tracked in each I frame.).

We have found the sensor fusion based system usually outperforms motion vector and color based trackers and the mean. Especially the SSF and MSF sensor fusion architectures outperform any of its sensors and mean without having the drawback of missing detection in face detector.

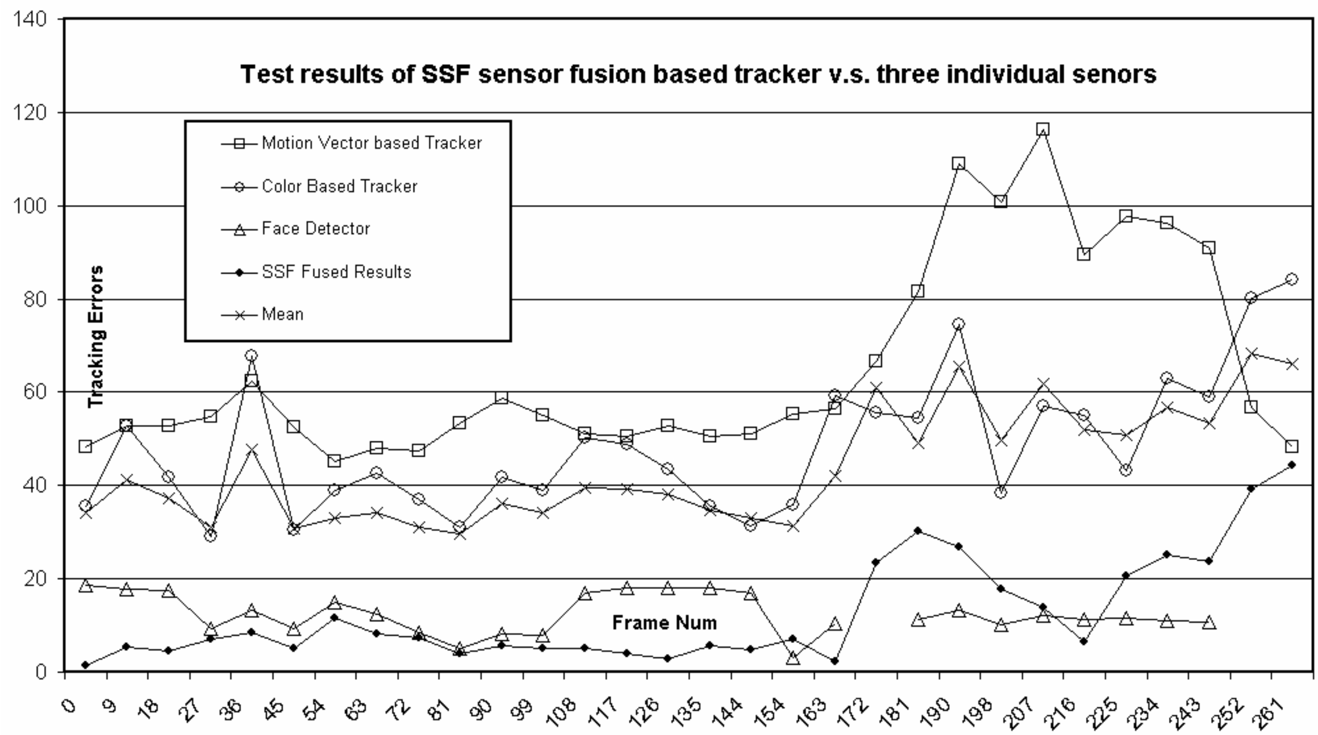


Fig 4.9(a): The test results of the SSF sensor fusion architecture

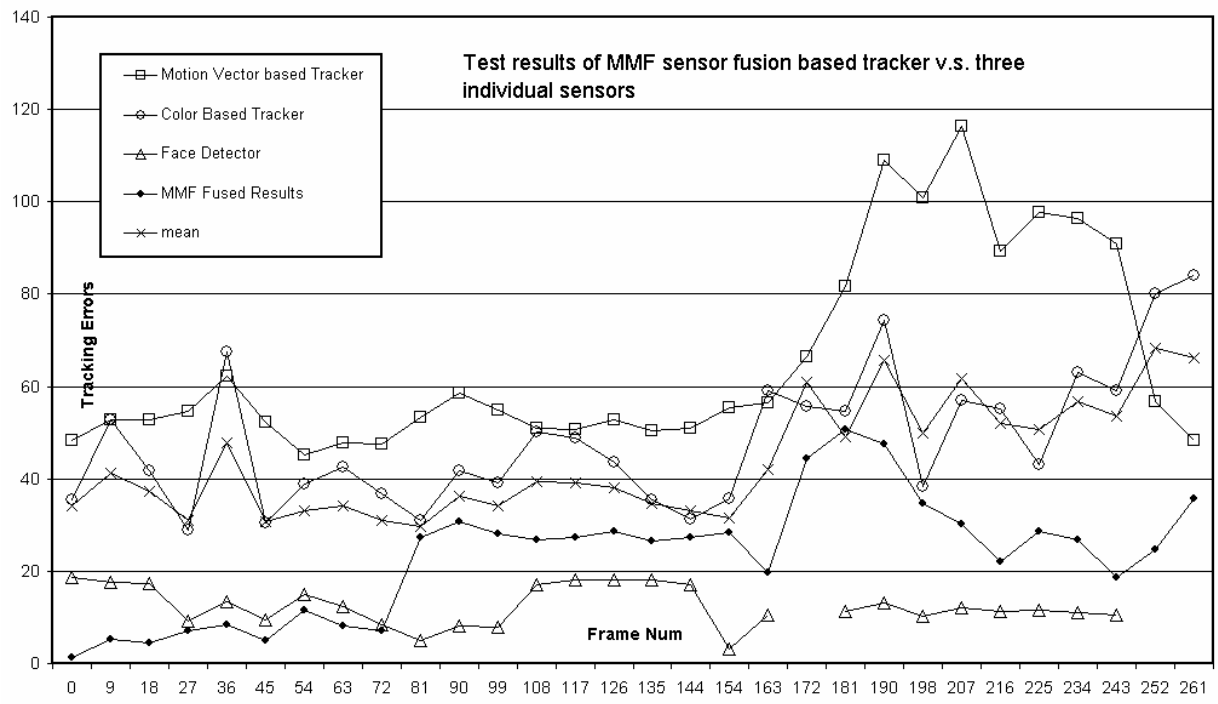


Fig 4.9(b): The test results of the MMF sensor fusion architecture

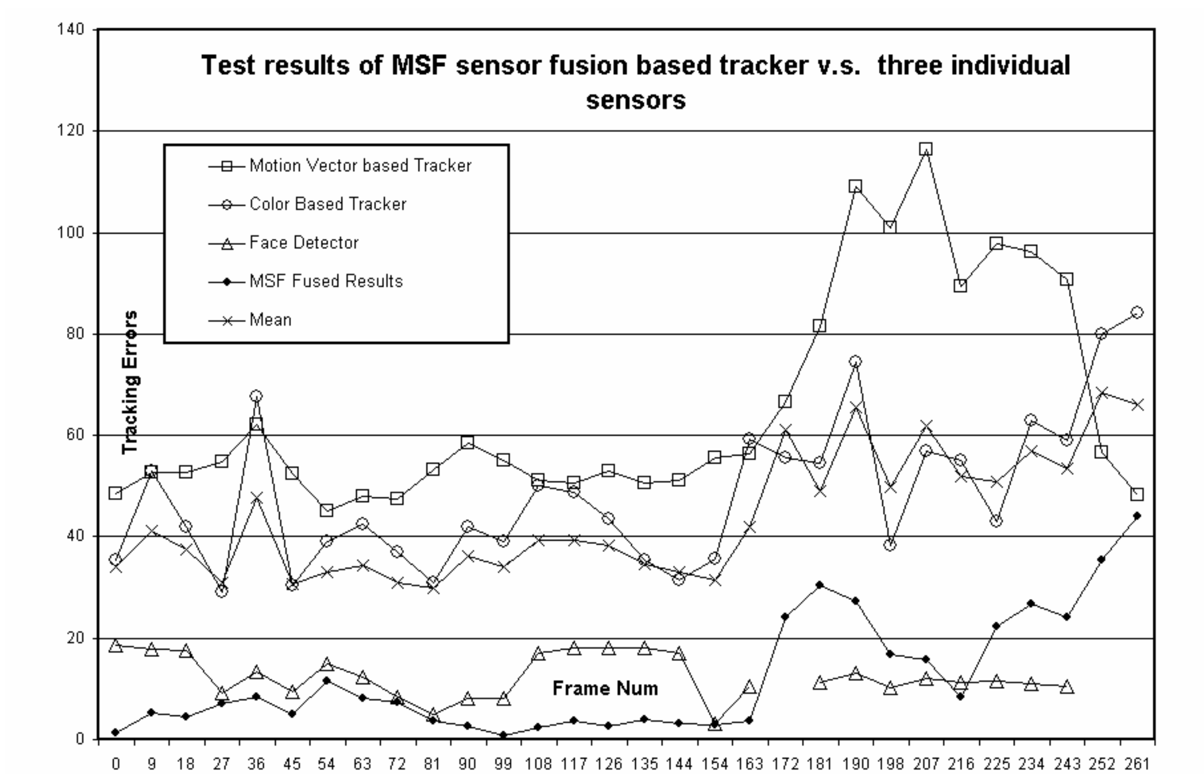


Fig 4.9 (c): The test results of the MSF sensor fusion architecture

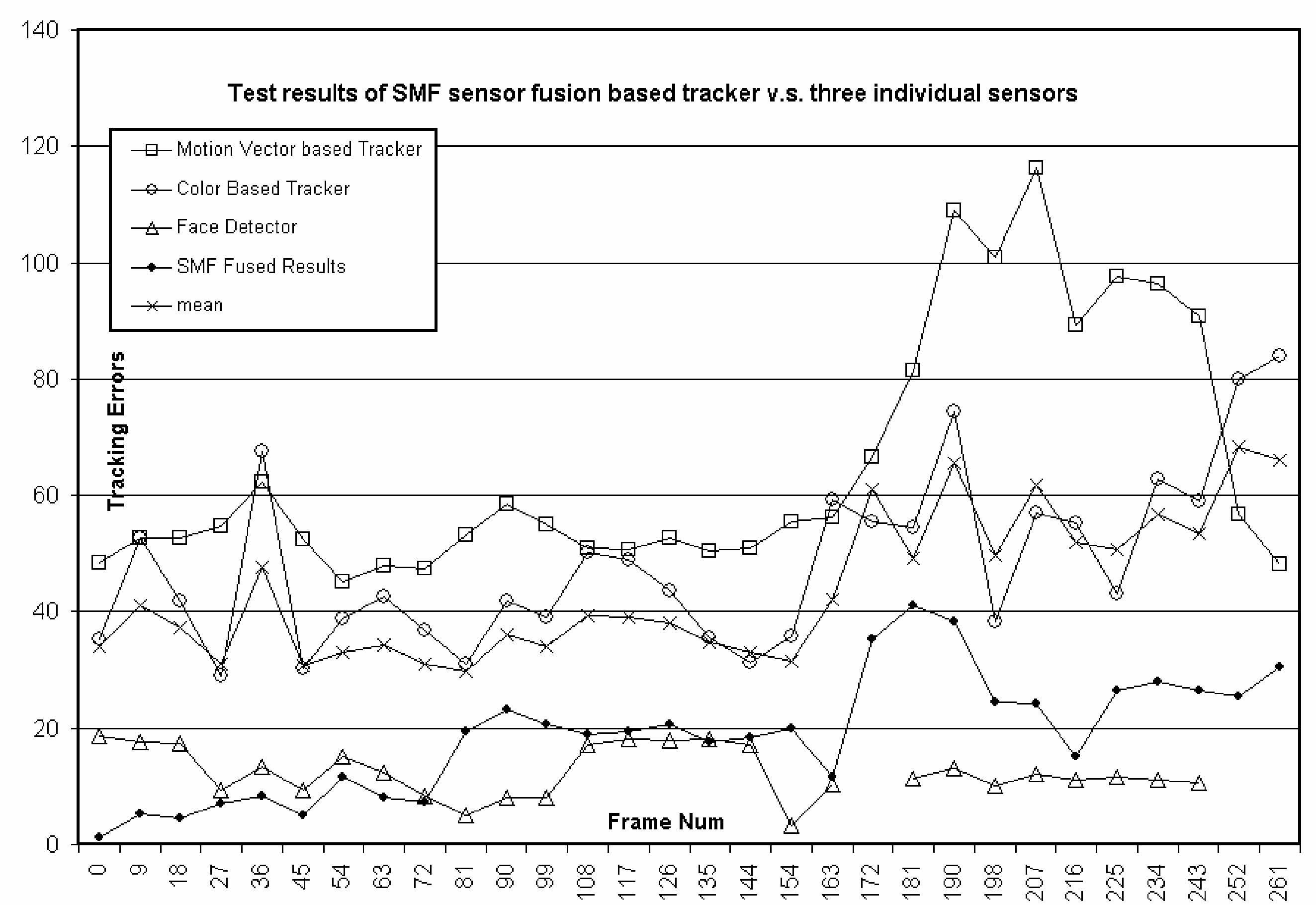


Fig 4.9 (d): The test results of the SMF sensor fusion architecture

Fig. 4.9 (a) to (d) show the performance of the three sensors for each of the four architectures of SSF, MMF, MSF and SMF respectively.

The gap in the results of face detector is caused by the changing of the face orientation or some inaccuracy of the face detector. However, this gap can be overcome by fused results from two object trackers in I frame.

Starting from frame 154, the two object trackers become worse, especially motion vector based object tracker because of error accumulation. However, by a corrective feedback in our sensor fusion approach, the fused results still perform well.

The graphs in Fig. 4.9 (a) to (d) show the performance of the three sensors for each of the four architectures of SSF, MMF, MSF and SMF respectively. They all solve two main problems by the mechanism that face detector helps object trackers and object trackers help face detector.

When the face detector fails completely (when the face changes pose or due to some inaccuracy of the face detector, as seen in Fig.4.9 (d)), the fused result still gives very good estimate of where the face is by two object trackers (the gaps of face detector in each figures can be overcome by fused results from two object trackers in I frame).

This also implies that the tracking sensors get a corrective feedback to remove the accumulated error, even when one of the sensors (like the face detector) generates wrong results. This is the second problem that is tackled, i.e. that of providing correct feedback, despite the failure of a sensor. So it is interesting to note that in cases when either the face detector or the tracker cannot perform well independently, the fused estimates from our proposed architectures perform successfully.

Now it is also observed that the fused results from the four architectures are not always better than the face detector in MMF and SMF sensor fusions. This is often due to the larger error of the tracking sensors. So we need to compare the performance of the four architectures on the same video. Figure 4.9(e) provides such a comparison in another video (MPEG 1 format, 417 frames), which got from news 5 MediaCorp, Singapore. It is observed here that the performance of the four versions of the architectures is similar in the beginning of the video, but the differences start to appear as the video progresses. It is seen that the SSF and MSF architectures show good overall performance, while the MMF and MSF give poorer overall performance. It means that measurement fusion is not suitable for our cases. It is mainly because that measurement fusion needs the assumption that individual measurement of each sensor is independent. However, in our specific domain, the measurements from face detector and object trackers are not independent. The features (skin color, texture, motion vector and color) used for those sensors are related. It causes the violation of the assumption of independence.

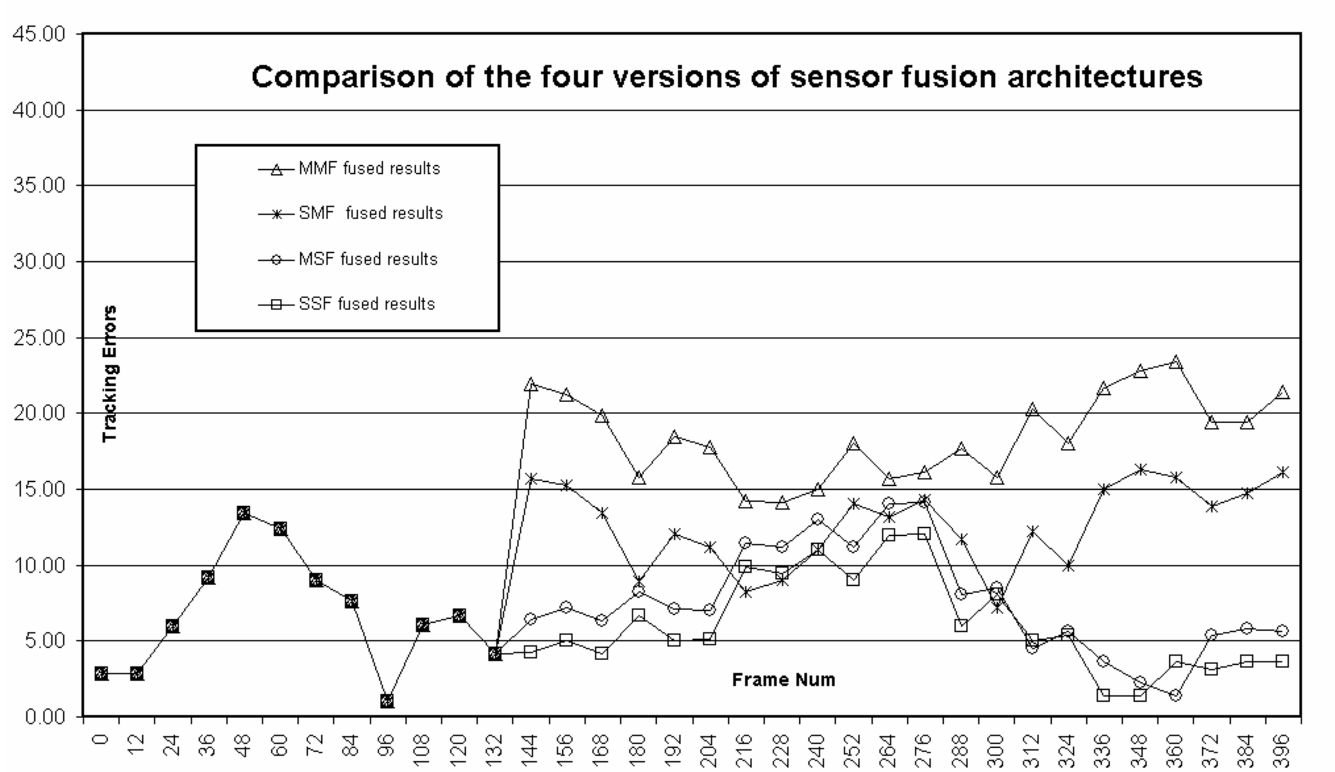


Fig 4.9 (e): Comparison of the four sensor fusion architectures

It is observed that the performance of the four versions of the architectures is similar in the beginning of the video, but the differences start to appear as the video progresses. It is seen that the SSF and MSF architectures show good overall performance, while the MMF and MSF give poorer overall performance.

We conclude from the experiments that the fused results are better than the results of the face detector and tracker independently. The results from the sensor fusion method are optimal compared to the mean of three sensors. Especially the SSF and MSF architectures have better overall performance. Figure 4.10 (a), (b) and (c) give the screen shots of some test sets with tracking results obtained by SSF architecture, and Figure 4.11 (a) and (b) give the face position trajectories of SSF tracking results compared to the ground truth.

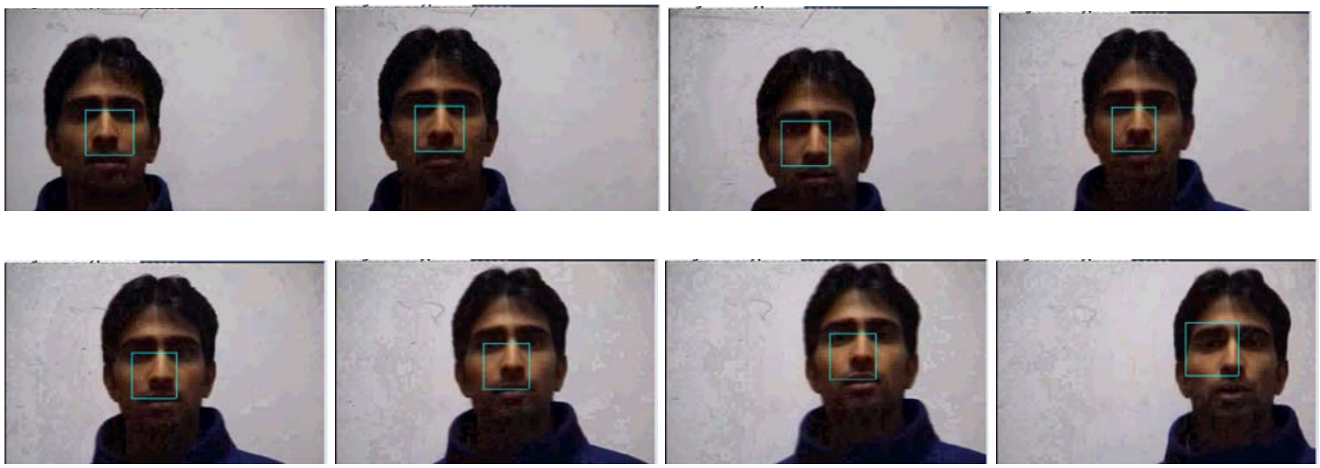


Figure 4.10 (a) Test Results in Lab Environment in the video 2.
(MPEG 1 formated Picture size 352x288)

Figure 4.10 (b) Test Results on MPEG 7 Test Set (video 3). (MPEG 1 formatted Picture size 352x288)

Figure 4.10 (c) Test Results on MPEG 7 Test Set (video 4). (MPEG 1 formatted Picture size 352x288)

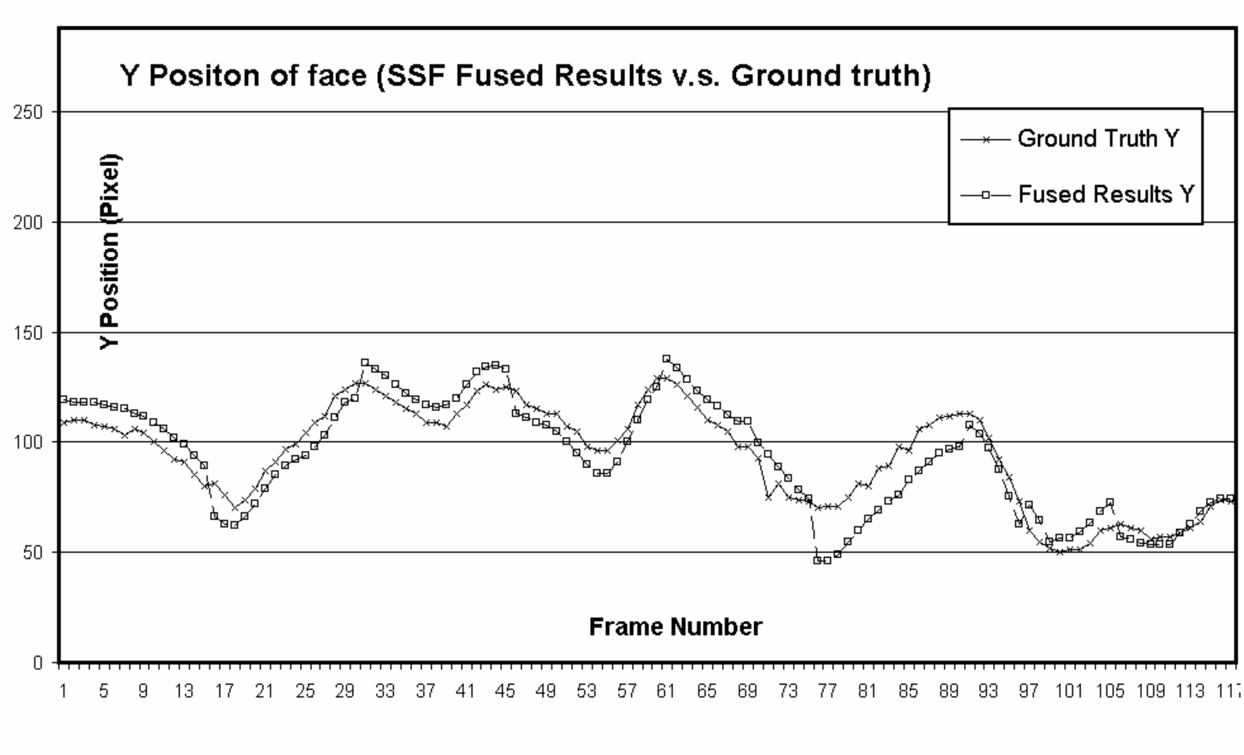


Figure 4.11 (a) (Results of SSF Tracker) Y position of face in the video 3

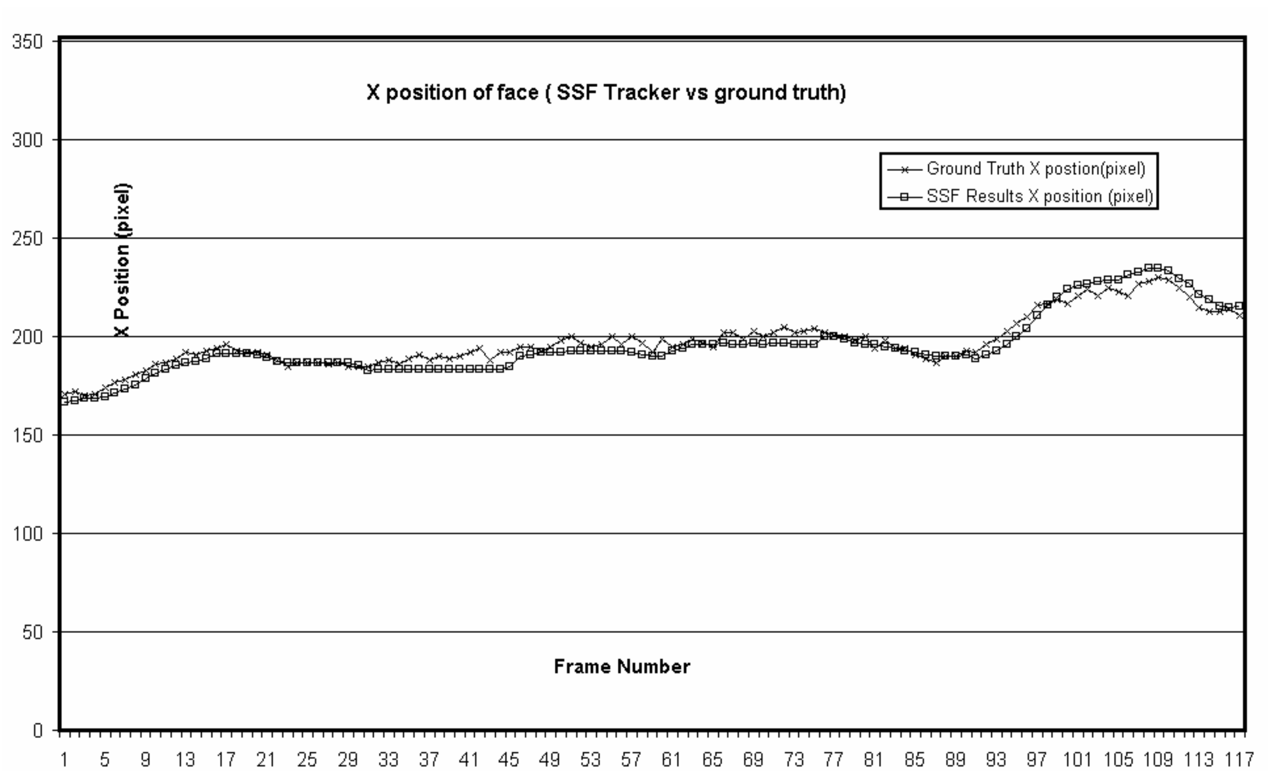


Figure 4.11 (b) (Results of SSF Tracker) X position of face in the video 3

Chapter 5

Conclusions

5.1 Conclusions

In this thesis, a two-step approach of finding human face in compressed domain is presented for efficient management of the massive amounts of digital video data. First, an algorithm for detection face using neural networks in I frames of MPEG video is described, which utilizes spatial distribution of human faces. Secondly, by capturing temporal correlation of human faces in video, we have developed a sensor fusion based face tracker in compressed domain video with the help of the face detector and the object trackers.

5.1.1 Neural network based face detection using DCT coefficients in the I frame

In the I frame of the MPEG video, given the fact that human faces are still clustered well in high dimensional feature (DCT coefficient) space in compressed domain, it is possible for us to use DCT coefficients directly as features to classify the face pattern. Based on this, a DCT coefficients extraction scheme was proposed and the DCT coefficients were used in two stages: in the first stage, a skin color filter, based on DCT coefficients of chrominance, was used to locate skin regions. In the second stage, a 4×4 block-sized window was used to scan the skin regions in the compressed domain image to extract Y-DCT features (luminance). A neural network then was trained using these DCT features to classify patterns as faces or non-faces.

The preliminary results obtained show that overall this face detection method in I frame works well for front-view human faces in a certain range of news and home videos with efficiency.

Inside the face detection system, for the purpose of speeding up the face detection processing, the compressed domain skin color filter initially segments the skin color regions in each I frame. The filter works satisfactorily for all ethnic groups when the background has obviously different colors with faces. However, the filter may not work well when the background has colors similar to that of faces such as wood, and yellow ceiling or the video comes from different sources (the attribute of CCD of camera has been changed). This problem is mainly because of the assumption of the unimodal Gaussian distribution and DC values of chrominance. In fact, the simple unimodal Gaussian distribution (used for efficiency) may not fully capture variations of skin color caused by illumination and different camera type. In addition, DC values may not be accurate enough in some cases.

The main method in our face detection system presented here is to use a neural network as a classifier and DCT coefficients as features to classify face pattern and non-face pattern. We have proved that it is suitable to directly use DCT coefficients as features to classify pattern. Moreover, since DCT coefficients have compact information about patterns, use of these compact features can reduce the complexity of the classifier. A DCT coefficients extraction scheme is proposed. Using this scheme, it is possible to adapt many successful face detection algorithms in the pixel domain to operate directly in the compressed domain.

With some errors, induced by DCT blocks recalculation and I frame downsampling, the compressed domain neural network face detector may not work as well as some pixel domain face detectors. In addition, since we just deal with front-view faces, the profile and other side-view faces cannot be detected in our system.

5.1.2 A sensor fusion based face tracker in the compressed domain

In this thesis we also presented an automatic face tracker based on a novel sensor fusion architecture, which works completely in compressed domain video. Four versions of the architectures, using state vector fusion and measurement fusion of Kalman filters, are compared to find the best one. We found first that, as desired, the fused output is better than the result of any single sensor as well as their mean. Among the four versions of the architectures proposed, results show that the SSF architecture performs the best, with the

MSF architecture being a close second. SMF and MMF architectures perform poorly in the later stages of the video. It is mainly because measurement fusion needs the assumption that individual measurement of each sensor be independent. However, in our specific domain, the measurements from face detector and object trackers are not independent. The features (skin color, texture, motion vector and color) used for those sensors are related. It causes the independence assumption to be violated. Therefore, the measurement fusion is not suitable for our case.

5.1.3 Applications

As we mentioned in chapter 1, in order to answer the following questions in videos, the human faces in videos have to be automatically or at least semi-automatically detected and recognized.

- Find the video showing my daughter *Lisa* in the swimming pool.
- Present the news clip that shows *Saddam Hussein* speaking.
- Summarize the video clips to show only the news anchor

As shown in Figure 5.1, being part of the DVA system (subsystem of DIVA project), the human face based indexing subsystem in DVA is mainly to find and recognize individuals in videos and save these information into the metadata. With this information, some functionalities such as video searching, video presentation, video summary, can be implemented. The works presented in this thesis are the face detector and the face tracker (the first gray part in Figure 5.1). The aim of the work is to find the locations of the human faces in videos. Consequently, with the help of the face recognition module, human faces can be individually identified.

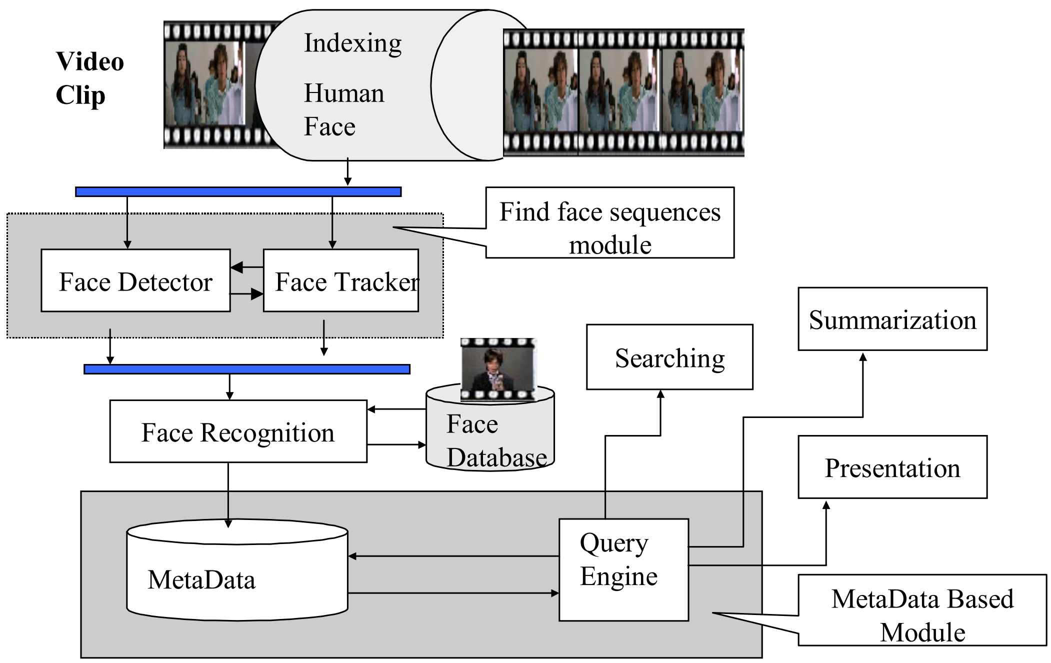


Figure 5.1 Human face based indexing system and it applications in the DVA system

Utilizing finding face sequences module as fist step, the high level human activity driven applications (video presentation, video query, and video summarization) were implemented in the DVA system.

To be able to achieve the functionalities as shown in Figure 5.1, medadata based module (the second gray port in Figure 5.1) is needed and it mainly provides functionalities of managing XML based metadata, including the indexing/annotation module and query engine. The indexing/annotation module primarily provides an interface for the DVA system to access the metadata stored in XML format. The annotator of the indexing module also allows users to manually input some data for some information, which cannot automatically be indexed, such as the names of persons, places, and events. With the helps of indexing/annotation module, the results of the recognized human faces and their locations are marked and stored as metadata in the XML format. For the purpose of retrieval, the query engine also helps the users build their query expressions to retrieve the video sequences from XML based metadata storage, and even presents the ranked video clips or sequences from the video database.

Once the information about the individual human faces and their locations in videos have been already known in metadata, with the help of the query engine, we can implement the high level human activity driven applications (video presentation, video query, and video summarization) in our DVA system.

The applications (video presentation, video query, and video summarization) in DVA are content based. It means that these applications can be also used for manipulating content other than humans. To answer the questions posed earlier, in the later sections, we will later discuss human activity driven functionalities in these applications, utilizing the results from finding human faces module. Note that the implementation of these functionalities still need other information in metadata, for which manual annotation is often required. These applications have been developed by other group members of the DVA project.

5.1.3.1 Human activity driven video query

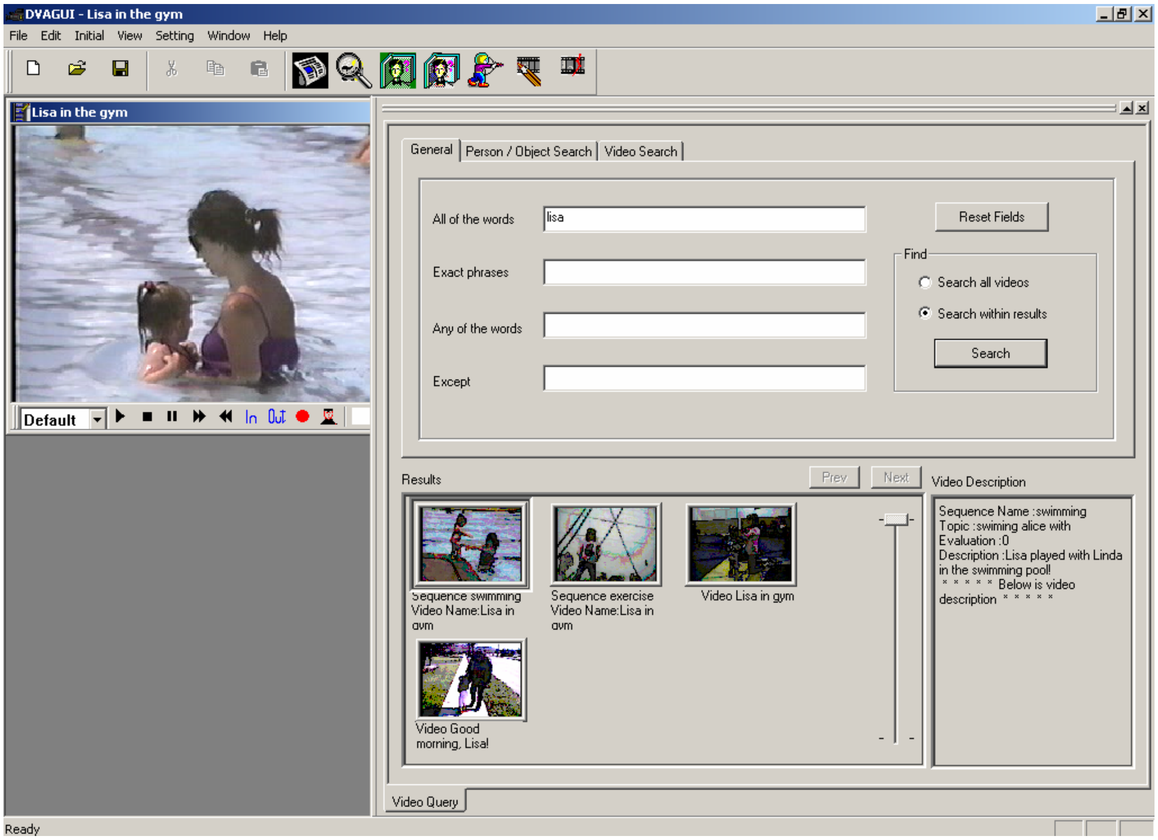


Figure 5.2 User interface for human activity driven video query in the DVA system *"Lisa in the swimming pool*" can be easily solved by searching "*Lisa*" keyword and subsequently selecting the video, which shows swimming pool, among the returned candidate videos.

As long as the information about locations of individual persons coming from face identification module (finding face sequences module plus face recognition) has been already stored in the metadata, the video query user interface in DVA system, as shown in Figure 5.2, can allow users to search video database to find persons of interests by flexible query GUI. Such questions as "*Find the videos or sequences showing my daughters Lisa in the swimming pool.*" can be easy solved by searching for "*Lisa*" keyword and consequently selecting the video, which shows swimming pool among the returned candidate videos.

5.1.3.2 Human activity driven video presentation

Recording home videos with camcorders is much more popular than playing them back. This is due to the fact that unedited home video is raw footage, and not organized well and thus tends to be boring to watch. In another aspect, home video is not for the general audience, but for friends, family and oneself. The playback priorities may not be same for different viewers. Therefore building flexible video presentations based on the content, especially human activities, is critical.

Utilizing the information of locations of human faces in video, we have built human activity driven video presentation in DVA project. Figure 5.3 shows one of the GUI for video presentation. This GUI allows us to select some information from the metadata to be shown while presenting. Suppose we want to show some information about *Saddam Hussein*. We can select this information in this GUI. When *Saddam Hussein* appears in video while presenting, some information about *Saddam Hussein* will be displayed. Note that face identification is used to find out when to display information along with video. The information to be displayed has to be annotated manually or by other automatic methods.

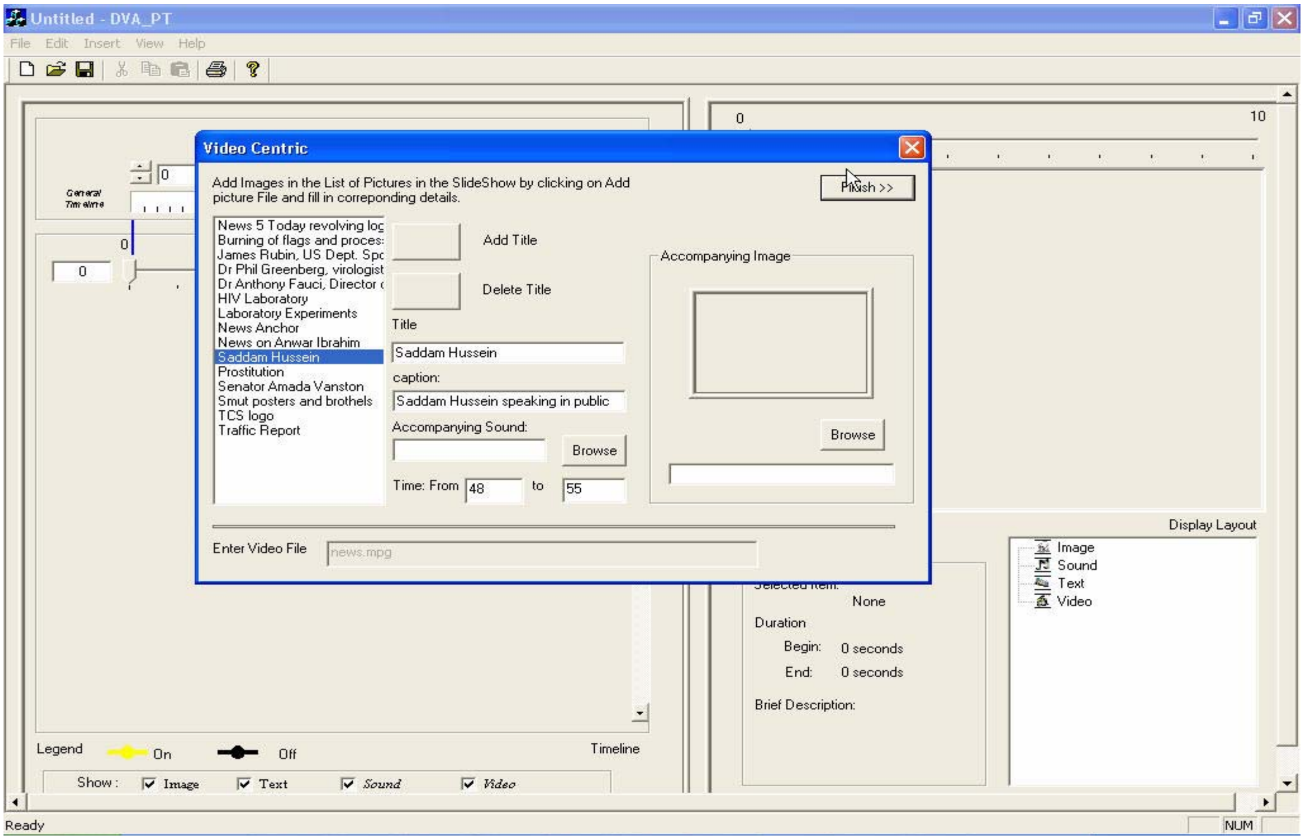


Figure 5.3 One of User Interfaces for human activity driven video presentation in DVA system

This GUI allows us to select some information from the metadata to be shown while presenting. Suppose we want to show some information about *Saddam Hussein*. We can select this information in this GUI. When *Saddam Hussein* appears in video while presenting, some information about *Saddam Hussein* will be displayed.

5.1.3.3 Human activity driven video summarization

Video summarization is to obtain a shorter version assembled by picking important segments from the original. Based the face identification system, one can summarize video by only including persons of interests in the video and leaving others out.

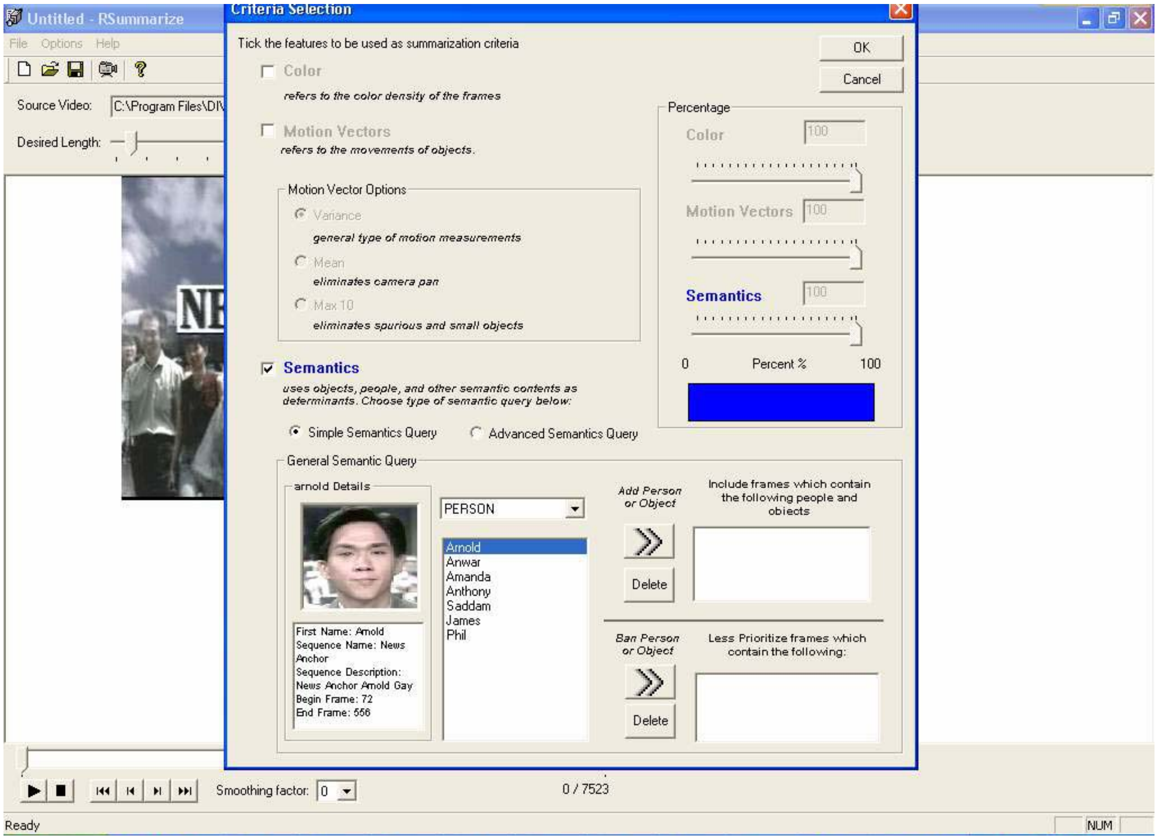


Figure 5.4 One of User Interfaces for human activity driven video summarization in the DVA system
We can just select some persons of interests, such as the news anchor. The final summarized video will only show some shots, which have the persons interested in.

As shown in Figure 5.4, we can just select some persons of interests, as the news anchor for instance. The final summarized video will only show some shots, which have the persons that we are interested in.

5.2 Future Work

Our future work will include experiments of other methods (Gaussian mixture distribution model, PCA, and etc.) in order to find the best classification method for use with DCT coefficient features. Use of other information available in MPEG video (such as motion vector information), and domain knowledge (video attributes) is also on the cards. In order to obtain an empirical evaluation of face detection methods in video, we plan to build a standard color video database for evaluation of face detection approaches in video.

5.2.1 Improving the neural network based face detection in compressed domain

5.2.1.1 Applying other classification methods in compressed domain

Using the DCT coefficients extraction scheme proposed here, we can make some experiments with other classification methods (Gaussian mixture distribution model, PCA, FDA, and etc.) in order to find the best classification method for use with DCT coefficient features.

Using *Fisher's Linear Discriminant* (FLD) could be one direction. In order to reduce the dimensionality of the features, features can be projected to a lower dimensional feature space. FLD provide a good way to project, since it maximize the variations between classes and minimize the variations within class while projecting. So the low dimensional space, which is projected using FLD, is good for classification. Recent research shows that the Fisherface method outperforms the widely used eigenface method in face recognition when the training samples are adequate.

5.2.1.2 Speeding up the face detection processing by building multi position face model

In face detection procedure, the reconstruction of the DCT blocks is time consuming. So in the future, we want to speed up detection processing by avoiding the reconstruction. However, without reconstruction, in order to detect faces, which are not aligned with block positions, during block-by-block scanning, more face patterns should be included to the training examples. Some additional face training patterns are shown in Figure5.5.

left-bottom shifted   left-top shifted   right-bottom shifted   right-down shifted

Fig.5.5 Different alignment position faces (shift by 3 pixels in both directions)

We may see from Fig.5.5.that in different alignment positions, some face features are shifted obviously, for instances, mouth and nose, and moreover, some face features may even cease to exist (brows in two bottom shifted face patterns and half of eyes are lost). These variations, which are induced by different alignment position face patterns, severely influence the stabilization of the face model. And this in turn makes the high frequency DCT coefficients unreliable for both face model training and face detection functions in [27]. This is one of the important reasons why [27] cannot achieve excellent performance.

Therefore, in order to solve this problem, we may label all the training samples by not only negative (non-face) and positive (face), but also the alignment position as well when we train the neural network classifiers. This could be explained by treating the face patterns in the different alignment positions as different classes. So the face detection problem changes from two-class problem to a multi-class problem. Following this philosophy, we can develop multi position face model based on our current compressed domain neural network face detection approach presented in chapter 3.

By analysis of the variations among face patterns, we find out that differentiating the spatial positions is the key to build a stable face model. There are mainly three variations, which should be removed or reduced to build a stable face model, among face patterns, the lighting variations (comes from different lighting conditions), the scaling variations (different scaling level), and the spatial alignment position variations. The lighting variation can be reduced by applying normalization. We have reason to believe that the spatial aligning position variations are more obvious than the scaling variations. So having different face classes in different spatial aligning position could build a stable face model and consequently improve the performance.

Therefore we may try to build multi position face model in the future. Multi position face model can be explained below. There are 64 possible spatial alignment positions for each training face in the compressed domain. It is neither necessary nor possible to differentiate all the 64 possible spatial positions. We may just use 9 out of 64 possible spatial positions to build 9 classes. By adding one non-face class, we have a total of 10 classes. A multi-output neural work is trained by training data of 10 classes to classify not only face and non face but also the spatial alignment position. The face detector has pixel level accuracy rather than block level accuracy unlike what could be expected from a compressed domain face detector where in data for processing is available in terms of blocks, even though the face model scans the images block by block. This in turn speeds up the face detection process without any resolution loss.

5.2.2 Improving the method of finding faces in video

5.2.2.1 Improving current sensor fusion based face tracker

Based on current work of sensor fusion based face tracker, there are several things that can be done to improve the overall performance of the system in the future. The performance of the individual trackers itself can be improved. The tracking sensors can be improved by the DCT reconstruction and the optic flow calculation on DC images for instance. The face detector can be improved to account for greater degree of pose change and apply other even more suitable classification methods (Gaussian Distribution Model, FLD and etc.). The Kalman filtering based sensor fusion method can also be improved by following possible approaches:

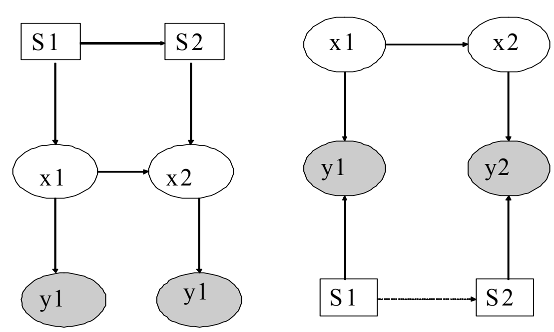


Figure 5.6 Some switching Kalman filter models represented as a graph model .Square nodes are discrete, oval ones are Gaussian. Shaded nodes are observed, clear nodes are hidden.

1. In current research, we assumed that the face dynamics are linear and noise is white and the parameters of model are time-invariant. Based on these assumptions, we used the constant velocity dynamic model for the face movements and white noise to get the parameters of Kalman filters. In the future, we can estimate these parameters from data using EM algorithm [43].

2. The face dynamics are not linear and are subject to non-Gaussian noise in real video. One approach to this problem is to use Dynamic Bayesian Networks. Among them, the Hidden Markov Models (HMMs) is the simplest example.

3. Another method to model the non-linearity is to have a bank of M different linear models and to switch between them or take some linear combination of them [44] as shown in Figure 5.6.

- "Switching dynamics" means there are K different linear dynamic models to model human faces movements. In different time t, the face dynamic model is not same, which is indicated by S_t. It could be constant velocity or constant acceleration dynamic model. Since S_t is hidden, we can use $Pr(S_t= I \mid y_{1:t})$ as weights to combine each sub-model. This can also be thought of as a mixture of Kalman filters with different dynamic models.

- "Switching observation" We can also assume that observation (measurement) parameters will be changed along with time, specified by S_t in Figure 6.1 b. By this assumption, the observation noise could become mixture of Gaussians instead of white noise. In our current research, white noise is assumed. However, in many cases, this is not true. By using "switching observation", we can use more general measurement noisy model "mixture of Gaussians" instead.

- "Switching dynamics and observation" We can also assume that dynamics and observation parameters will all be changed along with time. This assumption leads to an even more general case.

5.2.2.2 Finding faces in videos using offline methods

The aim of the research presented in this thesis is to find human faces (face sequences) in videos for the purpose of indexing. From the video point of view, human faces in videos have three dimensional distributions: two spatial distributions which describe appearance of human faces in the frame, and one temporal dynamics which describe temporal correlation of human faces along with time. The research in this thesis is trying to capture this three dimensional information by face detection (spatial distribution) and object tracking (temporal correlation) and use Kalman filtering to fuse them together to achieve a robust result.

Following this philosophy, we can even more go further. Given that fact that finding face sequences in videos is for the purpose of indexing, real time processing (frame-by-frame processing) is not necessary. Since human face dynamics are continuous, only restricted within one shot, we could process offline shot-by-shot instead of frame by frame. By performing shot-by-shot processing, finding face sequences is not separated into the two tasks: face detection and face tracker, any more. The 3D location of the face sequences within one shot can be found out in the same time. The most important benefits of doing this offline processing is that, with the help of understanding the temporal correlation of human faces, hopefully, the task of finding face sequences may not as difficult as that of face detection in an image. Moreover, the temporal correlation of human faces in videos (face dynamics) has its own attributes, such as occlusions, pose changes, camera movements. We can learn these face dynamics by applying statistical models. The trained statistical models in turn may help us to find face sequences regardless of occlusions and pose changes. These are all fruitful areas for further investigation.

REFERENCES

[1] M.H.Yang, D. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34-58, January, 2002.

[2] E. Saber and A. M. Tekalp, "Frontal-View Face Detection and Facial Feature Extraction using Color, Sharp and Symmetry Based Cost Functions," *Pattern Recognition Letters*, vol. 19, pp. 669-680, 1998.

[3] J. Wang and T. Tan, "A new Face detection method based on shape information," *Pattern Recognition Letters,* vol. 21, pp.463-471, 2000.

[4] K. C. Yow and R. Cipolla, "Detection of Human Faces under Scale, Orientation and Viewpoint Variations," In *Proc.Int'l Conf. Automatic Face and Gesture Recognition*, pp. 295-300, 1996.

[5] D.J.Beymer, "Face recognition under varying pose," A.I.Memo, 1461, Center for Biological and Computational Learning, M.I.T., Cambridge, MA, 1993.

[6] R. Brunelli and T. Poggio, "Face Recognition: Features versus Templates," In *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol.15, pp.1042-1052, 1993.

[7] K. K. Sung and T. Poggio, "Example-based learning for view-based human face detection," Tech. Rep. 1532, M.I.T., Artificial Intelligence Laboratory and Center for Biological and Computational Learning, 1994.

[8] H.Rowley, S.Baluja, and T. Kanade, "Neural Network-based Face Detection," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, pp. 23-38, January, 1998.

[9] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Detection," In *Proc. fifth Int'l Conf. Computer Vision (ICCV)*, June 1995.

[10] E. Osuna, R. Freund, and F. Girosi, " Training Support Vector Machines: An Application to Face Detection," In CVPR, 1997.

[11] M. F. Augusteijn and T. L. Skujca, "Identificaiton of human faces through texture-based feature recognition and neural network technology," In *proceeding of IEEE Conference on Neural Networks*, pages 392-398, 1993.

[12] Y. Dai and Y. Nakano, "Face-Texture model based on SGLD and its application In face detection in a color scene," *Pattern Recognition*, 29(6): 1007-1017, 1996.

[13] J. Yang, W Lu, and A. Waibel, “Skin-color Modeling and Adaptation,” CMU Technical Report CMU-CS-97-146, 1997.

[14] J. Cai, A. Goshtasby, and C, Yu, “Detecting human faces in color images,” In *Proceedings of the 1998 International Workshop on Multi-Media Database Management Systems*, pages 124-131, 1998.

[15] S.-H. Kim, N,-K. Kim, S.C. Ahn, and H. -G. Kim, “Object oriented face detection using range and color information,” In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 76-81, 1998.

[16] M.-h. Yang and N. Ahuja, “Gaussian mixture model for human skin color and its application in image and video databases,” In *Proceedings of the SPIE: Storage and Retrieval for Image and Video Databases VII*, volume 3656, pagers 458-466, 1999.

[17] S. Eickeler, S. Muller, and G. Rigoll, “High Quality Face Recognition in JPEG Compressed Images,” In *IEEE Int. Conference on Image Processing (ICIP),* Kobe, Japan, pp. 672—676, Oct. 1999.

[18] H. Stern and B. Efros, “Adaptive Color Space Switching for Face Tracking in Multi-Colored Lighting Environments,” In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition (FGR’02)*, 2002.

[19] L. Fan and K. K. Sung, "Face Detection and Pose Alignment Using Colour, Shape, and Texture Information," In *Proceedings of Workshop on Visual Surveillance*, Dublin, Ireland, June, 2000.

[20] T. Kanade, "Picture processing by computer complex and recognition of human faces," Technical report, Kyoto University, Dept. of Information Science, 1973.

[21] V. V. Kohir and U.B. Desai, "Face Recognition Using A DCT-HMM Approach," In *Workshop on Advances in Facial Image Analysis and Recognition Technology (AFIART)*, Freiburg, Germany, June 1998.

[22] M. Kirby and L. Sirovich, "Application of the Karhumen-Loeve procedure for the characterization of human faces," In *IEEE Tr. On PAMI,* vol. 15, no. 10, pp. 1042-1052, Oct. 1993.

[23] M.Turk, and A. Pentland, "Eigenfaces for Recogniton," *J. Cognitive Neuroscience*, vol.3, no. 1, pp. 71-86,1991.

[24] A. J. O'Toole, H.Abdi, K. A. Deffenbachar and D. Valentin, "Low dimensional representation of faces in higher dimension of the face space," *J. Opt.* Soc. Am. A, vol. 10, no. 3, pp. 405-411, 1993.

[25] J. N. S. Kwong and S.-Gang Gong, " Learning Support Vector Machines for A multi-view face model," In *British Machine Vision Conference (*BMVC99), The University of Nottingham, 13-16, September, 1999.

[26] H. Wang and S. –F. Chang, "A highly efficient system for automatic face region detection in mpeg video," In *IEEE Trans*. CSVT, 7(4), 1997.

[27] H.Luo and A.Eleftheriadis, "On Face Detection in the Compressed Domain," *ACM Multimedia-00*, Oct. 2000.

[28] J. Wang, M. S. Kankanhalli, P. Mulhem, and H. H. Abdulredha, "Face detection using DCT coefficients in MPEG video," In *Proc. IWAIT*, 66-70, 2002.

[29] B. Heisele, T. Poggio, and M. Pontil, "Face Detection in Still Gray Images," A.I.Memo, No.1687 M.I.T. 2000.

[30] W. Kou, and T. Fjallbrant, "A direct computation of DCT coefficients for a signal block taken from two adjacent blocks," In *IEEE Transactions on Signal Processing, vol* 39 (7), pp. 1423-1435, 1997.

[31] S. Spors and R. Rabenstein, "A Real-Time Face Tracker For Color Video," In *IEEE Int. Conf. On Acoustics, Speech & Signal Processing (ICASSP)*, Utah, USA, May 2001.

[32] J. M. Rehg, K. P. Murphy, and P. W. Fieguth, "Vision-Based Speaker Detection Using Bayesian Networks," In *Proceedings of Computer Vision and Pattern Recognition,* pp. 110-116.1999.

[33] T.-S. Chua, Y. –L. Zhao, and M.S. KanKanhalli, "An Automated Compressed-Domain Face Detection Method For Video Stratifications," In *Proceedings of MMM-2000 International Conference on Multimedia Modelling*, Nagano, Japan, USA, pp. 333-347 2000.

[34] H. Wang, H.S. Stone, and S.-F. Chang, "FaceTrack: Tracking and Summarizing Faces from Compressed Video," In *SPIE Multimedia Storage and Archiving Systems* IV, 19-22 Sept, Boston, MA,1999.

[35] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 711-720, July 1997.

[36] N. Merhav and V. Bhaskaran, "Fast Algorithms for DCT-Domain Image Down-Sampling and for Inverse Motion Compensation," In *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 468-476, June 1997.

[37] S. –F. Chang and D.G. Messerschmitt, "Manipulation and Compositing of MC-DCT Compressed Video," In *IEEE Trans. On Selected Areas in Communications*, Vol.13, No. 1, pp. 1-11, Jan. 1995.

[38] Y. –L. Zhao, M. S Kankanhalli and T. -S. Chua, "A Compressed-Domain Fractional Scaling Technique for Image and Video," Technical Report, School of Computing, National University of Singapore, 2000.

[39] R. Achanta, "Object Tracking in Compressed Domain MPEG Video for Automated Content-Based Indexing," M.Sc. Thesis, Department of Computer Science, National University of Singapore, 2002.

[40] L. Favalli, A Mecocci and F. Moshetti, "Object tracking and hypermedia links creation in MPEG-2 digital video sequences," In *Proc. Of IEEE Int'l Conf. On Circuits and Systems*, pp. IV-261 – 264, 1998.

[41] J. A. Roecker, and C.D. McGillem, "Comparison of two-sensor tracking methods based on state vector fusion and measurement fusion," In *IEEE Transactions on Aerospace and Electronic Systems*, Volume: 24 Issue: 4 , Page(s): 447 –449, July, 1998.

[42] K. C. Chang, R. K. Saha, and Y. Bar-Shalom, "On Optimal Track-to-Track Fusion," In *IEEE Transactions on Aerospace and Electronic Systems,* VOL. 33, No.4, October 1977.

[43] Z. Ghahramani and G. Hinton, "Parameter estimation for linear dynamical systems," Technical Report CRG-TR-96-2, Dept. Comp.Sci., Univ. Toronto, 1996.

[44] K. P. Murphy, “Switching Kalman Filters,” Tech. Report 98-10, Compaq Cambridge Research Lab, 1998.

[45] J.Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Mogran Kaufmann Publishers, 1998.

[46] S.Haykin, *Neural Networks.* Macmillan, 1994.

[47] P. Viola, and M. Jones, “Robust Real-time Object Detection,” In *the Second International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing, and Sampling,* Vancouver, Canada, July 13, 2001.

[48] S. Z. Li, L. Zhu, Z.-Q. Zhang, A. Blake, H,-J. Zhang, and H. Shum, “Statistical Learning of Multi-View Face Detection,” In *Proceedings of The 7th European Conference on Computer Vision*. Copenhagen, Denmark. May, 2002.

[49] T. M. Mitchell, *Machine Learning,* McGraw-Hill, 1997.

[50] R.O.Duda, P.E.Hart, and D.G.Stork, *Pattern Classification*, Wiley_Interscience, 2000.