FaceTrack: Tracking and summarizing faces from compressed video

Hualu Wang, Harold S. Stone*, Shih-Fu Chang

Department of Electrical Engineering, Columbia University, 500 W120th St., New York, NY 10027

*NEC Research Institute, 4 Independence Way, Princeton, NJ 08540

ABSTRACT

In this paper, we present FaceTrack, a system that detects, tracks, and groups faces from compressed video data. We introduce the face tracking framework based on the Kalman filter and multiple hypothesis techniques. We compare and discuss the effects of various motion models on tracking performance. Specifically, we investigate constant-velocity, constant-acceleration, correlated-acceleration, and variable-dimension-filter models. We find that constant-velocity and correlated-acceleration models work more effectively for commercial videos sampled at high frame rates. We also develop novel approaches based on multiple hypothesis techniques to resolving ambiguity issues. Simulation results show the effectiveness of the proposed algorithms on tracking faces in real applications.

Keywords: Face tracking, face summarization, the Kalman filter, compressed domain, MPEG, video analysis.

1. INTRODUCTION

Recent growth of the Internet and advances in computer technologies have brought digital video to a rapidly expanding audience. However, managing and accessing video content is difficult. Video data is lengthy and linear in nature, making the process of downloading and viewing very tedious. Just one minute of MPEG-1 compressed video is more than 10 megabytes of data, and takes more than 20 minutes to download using a 56 kbps modem connection. Even with broadband network connections in the near future, the sheer amount of video data still demands efficient techniques that help users quickly find the pieces of videos of their interest.

Human faces are one of the most important subjects in many types of videos, like news, talk shows, sitcoms, etc. Faces provide crucial visual clues that help us understand the content of the video, and it is very natural for a user to search for videos using faces, like:

- Find me all *Larry King Live* shows in which Secretary of the State is a guest;
- Tell me if George's parents show up in a particular episode of *Seinfeld*.

We are developing a system called FaceTrack to track and summarize faces in compressed video sequences. In this context, *face tracking* means the detection of faces and tracing of their temporal continuity in video shots; *face summarization* means the clustering of faces across video shots and the association of faces with different people. By face summarization, a concise representation of lengthy videos can be generated, which helps the users quickly grasp the core content of the videos and find the particular segments of their interest.

Previous work on face detection and tracking mostly focuses on uncompressed images and videos. A neural network scheme for face detection is reported comprehensively[7], which works on still images at full resolution. Current face tracking techniques are designed mainly for human computer interface applications, where only a single face is in the video and face detection is always assumed to be accurate[5]. In FaceTrack, we take a different approach that detects and tracks faces directly from compressed videos. Our face tracking framework takes into consideration possible errors of face detection and ambiguities caused by multiple moving faces.

In this paper we present the system architecture of the FaceTrack system. We propose a framework for face tracking directly from compressed video data. The framework is based on the Kalman filter and uses a multiple-hypothesis approach to solve the problems of tracking multiple faces in complex scenes. We also compare the effectiveness of various motion models when applied to the Kalman filter for tracking purposes in commercial videos (e.g., news, sitcoms, etc.), and find that constant-velocity and correlated-acceleration models work more effectively for commercial videos sampled at high frame rates. We use video shots extracted from commercial videos used in real applications as test data, in which multiple faces are usually present. Simulation results show that the proposed face tracking algorithms greatly reduce face detection errors and properly resolve tracking ambiguities. Face summarization is part of ongoing work and is also briefly discussed.

The paper is organized as follows. Section 2 gives an overview of the FaceTrack system. Section 3 introduces the face tracking framework. Section 4 compares various motion models and discusses their effectiveness when applied to commercial video. Section 5 presents the detailed face tracking algorithm. In Section 6 we discuss the face summarization problem. Simulation results are presented in Section 7. Section 8 concludes the paper.

2. SYSTEM OVERVIEW

The goal of the FaceTrack system is to generate summaries of video sequences using faces. The face summary is very useful in video/multimedia database applications. For example, a user can take a quick look at the summary before she decides whether or not to download and view the entire video. The summary also preserves crucial visual information of the original video that can be used by the database's search engines.

2.1. Problem definition and general approach

We want to solve the following problem in this project: Given a video sequence, generate a list of prominent faces that appear in this video, and determine the time periods of appearance for each of the faces.

To solve this problem, we first need to detect and track the faces in the video. A video sequence usually consists of multiple *video shots*, with scene changes at shot boundaries. The first task involves tracking faces within a video shot. Once the faces are tracked in each shot, we can group the faces across video shots into faces of different people. In this way, we can produce a list of faces for a video sequence, and for each face we can tell in which shots it appears, and when. We do not intend to recognize the faces, because face recognition is very difficult in unconstrained videos, where the size, number, and orientation of faces are not known *a priori*, and the lighting conditions vary greatly.

We also try to work in the compressed domain as much as possible to save the expensive computational overhead of video decoding. Specifically, we focus on MPEG-1 and MPEG-2 videos[10] that are widely adopted in applications like DVD and digital TV broadcast. Taking the compressed-domain approach does not preclude the decoding of a few selected video frames. When necessary, we can decode some frames to the pixel domain for further analysis. This is especially useful for grouping faces across video shots. Since we only need to do this for several representative frames for each face, the additional computational cost is low.

2.2. System diagram

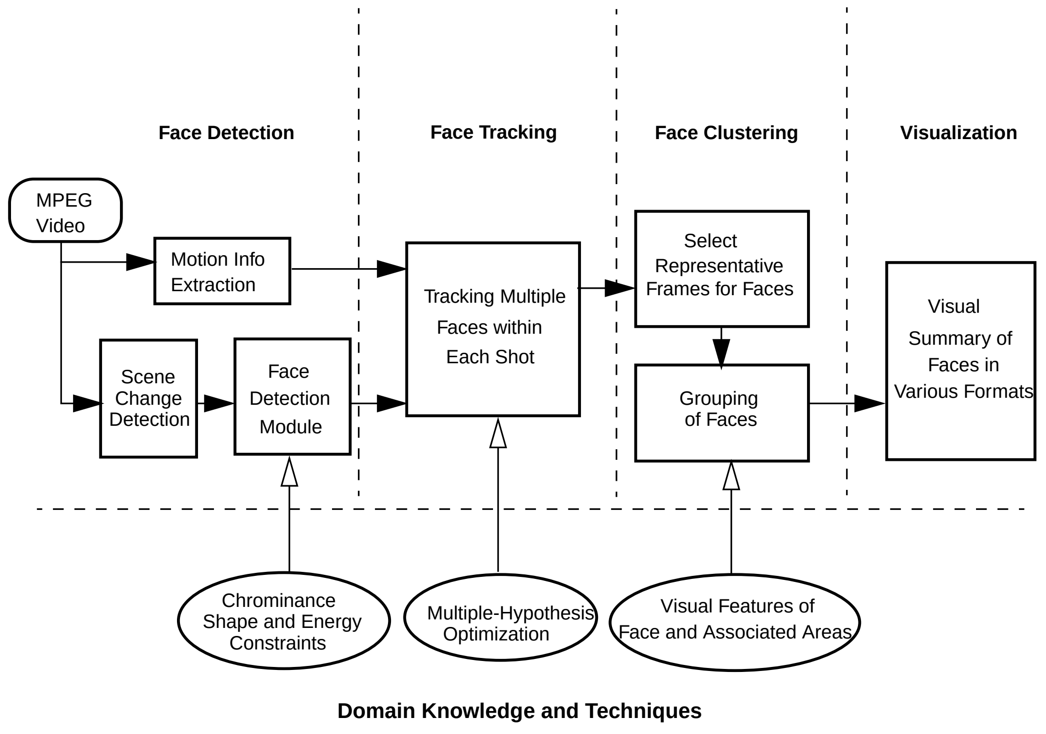

Face Detection
Face Tracking
Face Clustering
Visualization
MPEG Video
Motion Info Extraction
Scene Change Detection
Face Detection Module
Tracking Multiple Faces within Each Shot
Select Representative Frames for Faces
Grouping of Faces
Visual Summary of Faces in Various Formats
Chrominance Shape and Energy Constraints
Multiple-Hypothesis Optimization
Visual Features of Face and Associated Areas
Domain Knowledge and Techniques

Fig. 1. System diagram of the FaceTrack system.

Fig. 1 shows the system diagram of FaceTrack. The system consists of multiple modules in several stages. First, an MPEG video sequence is broken down into video shots using compressed-domain scene change detection algorithms[6]. Within each shot, faces are detected for I-frames, and motion information is extracted for P-frames. These are done only with minimal decoding of the MPEG video. The face detection module is based on an efficient face detection algorithm for MPEG-compressed video that we previously developed[9]. The algorithm has three stages, where chrominance, shape, and energy distribution (in the DCT domain) constraints are applied, respectively. Face detection results, along with motion information, are passed on to the tracking module.

Then, in the face tracking module, detected faces are tracked within each shot, using motion information and the Kalman filter prediction. Ambiguities are resolved by keeping multiple hypotheses and picking the most likely face tracks. Spurious face detection results (e.g., false alarms) are also reduced at this stage. Once face tracking is done, one or several representative frames are selected for every tracked face in each shot. These frames are decoded for pixel domain analysis and browsing.

Last, faces across shots are clustered using visual features in the face and its associated background regions. The summarized results are sent to the visualization module to generate the output to the user interface. The best way to visualize face summarization results depends on the application domain and the types of videos that are summarized.

3. FACE TRACKING – THE FRAMEWORK

The face tracking module uses the face detection results as the starting point. Its main purpose is to track the movement of the face in a video shot. The tracking result can be used to correct earlier errors made in face detection, and to resolve ambiguities such as close faces.

3.1. Related work

Much of the research on video object tracking has been done in the uncompressed domain. Manual input of object outlines is often required for initialization and periodic correction of the tracking process. Optical flows or affine motion models are usually used to estimate the movement of the object. Recently, people have started to work on object tracking in compressed video[8]. In this work, an accurate but computationally expensive object detector is applied to decoded I-frames periodically (e.g., once every 1000 frames), to initialize the tracking process and correct accumulated tracking errors. In between these “check points”, tracking is achieved by using motion vectors to approximate region motion that roughly corresponds to the object movement. By keeping a balance between tracking precision and computational cost, this method is able to track objects using compressed video only, with occasional decoding of I-frames.

In our project, the task of object tracking (in our case, the object is the face) raises new issues that are not present in the above examples: a) The locations of detected faces may not be accurate, because our face detection algorithm works on a coarse resolution level (e.g., 16x16 macroblocks). b) There are erroneous face detection results like false alarms and misses. c) There may be multiple faces in the videos that will cause ambiguities for tracking when they move close to each other. d) The region motion approximated using the motion vectors may not be accurate. Therefore, we need to investigate a framework for face tracking that is able to handle these issues properly while keeping face tracking mostly in the compressed domain.

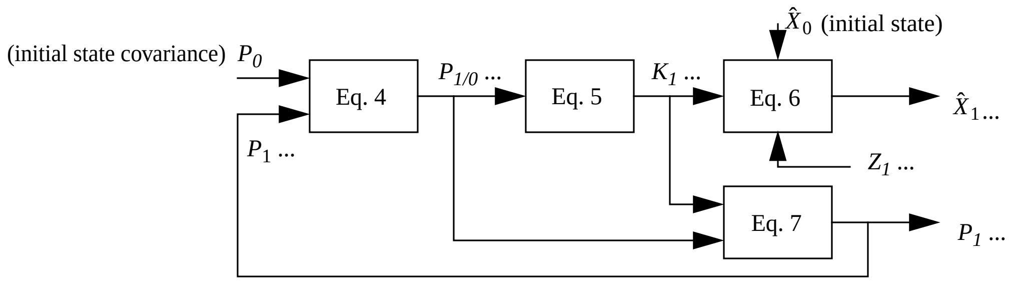
3.2. The Kalman filter

A linear, discrete-time, dynamic system is defined by the following difference equations:

$$\begin{cases} X_{k+1} = \Phi_k X_k + \Gamma_k W_k & (1) \\ Z_k = H_k X_k + V_k & (2) \end{cases}$$

where $\{X_k\}$ is the sequence of state variables of the system, $\{Z_k\}$ the sequence of observations (measurements), $\{W_k\}$ the system noise sequence, and $\{V_k\}$ the observation noise sequence. In practice, only the (noisy) observation data are available, and the problem is: Given $\{Z_1,..., Z_k\}$, find the optimal estimate of the unknown X_k, k=0, 1,... Given additional conditions that the noise sequences are Gaussian and mutually independent (with covariances $\{Q_k\}$ and $\{R_k\}$, respectively), and the initial state is Gaussian with known mean and covariance, both mutually independent of the noise sequences, there exists an optimal solution that can be calculated iteratively, which is called the *Kalman filtering algorithm*, or the *Kalman filter.* For detailed accounts on the Kalman filter, please refer to textbooks[1].

The following is a diagram of the Kalman filtering process:



One-step state prediction and its covariance

$$\hat{X}_{k|k-1} = \Phi_k \hat{X}_{k-1} \tag{3}$$

$$P_{k|k-1} = \Phi_k P_{k-1} \Phi_k' + \Gamma_{k-1} Q_{k-1} \Gamma_{k-1}' \tag{4}$$

Gain matrix for innovation

$$K_k = P_{k|k-1} H'_k (H_k P_{k|k-1} H'_k + R_k)^{-1} \tag{5}$$

State estimation and its covariance

$$\hat{X}_k = \hat{X}_{k|k-1} + K_k (Z_k - H_k \hat{X}_{k|k-1}) \tag{6}$$

$$P_k = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)' + K_k R_k K'_k \tag{7}$$

Innovation and its covariance matrix

$$\tilde{Z}_{k|k-1} = Z_k - \hat{Z}_{k|k-1} = Z_k - H_k \hat{X}_{k|k-1} = H_k \tilde{X}_{k|k-1} + V_k \tag{8}$$

$$S_k = E(\tilde{Z}_{k|k-1} \tilde{Z}'_{k|k-1}) = H_k P_{k|k-1} H'_k + R_k \tag{9}$$

Fig. 2. The Kalman filtering algorithm.

As shown in Fig. 2, the Kalman filter is an iterative algorithm that keeps taking in new observations $\{Z_k\}$. The error of the prediction of Z_k (Eq. 8) is termed the *innovation*, which is amplified by the gain matrix K_k (Eq. 5) as a correction term added to the one-step state prediction to get the new state estimate (Eq. 6).

3.3. The face tracking framework

If we model face movement using a linear system described by Eqs. 1 and 2, the state vector $\{X_k\}$ is the real kinematic information of the face, i.e., position, velocity, and sometimes acceleration. The observation vector $\{Z_k\}$ is the position of the *detected* face, which may not be accurate. The observation noise $\{V_k\}$ represents the error of detected face locations from the real face locations. The covariance of $\{V_k\}$ reflects the precision of the face detector. The statistics of $\{W_k\}$, on the other hand, shows the closeness of the motion model in the system equations to the real dynamics of the object. The Kalman filter is optimal in the ideal case. In practice, its performance depends on how well the noise sequences and the dynamics of the system are modeled. In Section 4, we will compare various motion models and discuss their effects on the tracking of faces in commercial videos.

Using the Kalman filter, we can predict and update the position and kinematic parameters of the faces. The detected face locations are used as the observations $\{Z_k\}$ to estimate the real locations of the faces. In practice, a 0.1 second time interval for state update is frequent enough for tracking objects in video. For a typical MPEG video (30 frames/second, 15-frame GOP, with IBBPBBP... structure), this means that state update is done for every I- and P-frame. For I-frames, we use the face detection results directly as the noisy observations. For P-frames, although the face detector works as well, we do not use the results directly, because P-frame face detection results are usually highly correlated with the results of adjacent I-frames, and are prone to more false alarms. Instead, we use the prediction technique based on motion vectors to estimate the location of faces in P-frames. Motion vectors are extracted from MPEG videos for P-frames, and are median filtered to remove anomalies. By projecting those motion vectors back into the face regions in the previous I- or P-frame, locations of the face regions in the current P-frame can be approximated. Unlike the related work[8], we do not take these approximated locations as the correct results, but treat them as noisy observations that are fed into the Kalman filter.

In the Kalman filter, given the assumptions of the statistical properties of the noise sequences and the initial state vector, the linear system equations lead to the Gaussian property of the state and observation variables. For example, the observation

Z_k has an n-dimensional (in the face tracking scenario, n=2) Gaussian distribution around its predicted value $\hat{Z}_{k|k-1}$, with covariance matrix S_k (Eq. 9). Contours of equal probability density can be defined by the *Mahalanobis distance* as in Eq. 10.

$$\gamma = \tilde{Z}'_{k|k-1} S_k^{-1} \tilde{Z}_{k|k-1} = [Z_k - \hat{Z}_{k|k-1}]' S_k^{-1} [Z_k - \hat{Z}_{k|k-1}] \tag{10}$$

It is known that the Mahalanobis distance is chi-squared distributed with n degree of freedom. It is a quantitative indicator of how close the new observation is to the prediction. In the face tracking scenario, this can be used as a discriminator that helps associate newly detected faces with previous ones. If the Mahalanobis distance is greater than a certain threshold, it means that the newly detected face is unlikely to belong to this particular face track.

When two faces move close to each other, or when there are false alarms and misses of face detection, Mahalanobis distance alone cannot help us keep track of the moving faces. In these situations, we need to hypothesize the continuation of a face track (in the cases of misses and occlusions), or the creation of new tracks (in the cases of false alarms and faces close to each other). The idea is to wait for new observation data to help make decisions – those unlikely tracks will have a low likelihood function, in the form of accumulated Mahalanobis distances (For a detailed discussion on various multiple hypothesis tracking techniques, see the reference paper[4]).

Overall, the Kalman filter provides us the basis of the framework for face tracking. With its iterative state updates and nice statistical properties, we can analyze the tracking results quantitatively, and generate multiple hypotheses as each of them evolves in time. In videos with complex scenes, such a framework is necessary, especially with imperfect face detection results. In Section 5, we will discuss the details of the face tracking module in the FaceTrack system.

4. COMPARISON OF MOTION MODELS

In the traditional application domain of the Kalman filter, the tracked objects are dynamic systems that follow relatively simple motion trajectories, e.g., orbiting satellites, launching rockets, aircraft, etc. Things are quite different in the video domain. Consider the faces in a commercial video: The movement of the face is pretty much unconstrained. The dynamics of the faces are unclear and hard to model. To ensure the performance of the Kalman filter, we test several commonly used motion models, along with one that we proposed, and investigate their effects on the Kalman filtering algorithm. We pick up two typical types of video shots from commercial videos: Type A are those in which a face has smooth and long movement, sweeping across the video frame (e.g., faces in a sitcom, or news stories). Type B are those in which the face movement is short and more irregular (e.g., faces of news anchors). These two types of video shots are commonly seen in commercial videos. To get the *ground truth* of the face trajectories, we manually identify the faces in these video shots, and locate the center of the faces frame by frame.

4.1. The motion models

We test four different motion models, namely the *constant-velocity model*, the *constant-acceleration model*, the *correlated-acceleration model*, and the *variable dimensional filter.* For the sake of brevity, we use CV, CA, AA, and VDF to denote these models, respectively. The CV and CA models are frequently used in the target tracking applications, while the VDF is proposed for the tracking of maneuvering targets. The AA model is what we proposed for the tracking of objects in commercial videos. In these models, the linear systems are assumed time-invariant so that the matrices Φ_k, Γ_k, H_k, Q_k, and R_k are constant. For different motion models, the state vectors and the system matrices will have different definitions.

The constant-velocity (CV) model:

$$X = \begin{bmatrix} x & y & x' & y' \end{bmatrix}' \quad Z = \begin{bmatrix} z_1 & z_2 \end{bmatrix}' \qquad W = \begin{bmatrix} w_1 & w_2 \end{bmatrix}' \qquad V = \begin{bmatrix} v_1 & v_2 \end{bmatrix}'$$

$$\Phi = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} T/2 & 0 \\ 0 & T/2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{11}$$

In this model (Eq. 11), state vector X is the position and velocity of the face centroid, in the two-dimensional Cartesian coordinates. Vector Z contains the coordinates of the observed position of the face. T is the time interval of the discrete system. Acceleration is modeled as system noise.

The constant-acceleration (CA) model:

$$X = \begin{bmatrix} x & y & x' & y' & x'' & y'' \end{bmatrix}' \quad Z = \begin{bmatrix} z_1 & z_2 \end{bmatrix}' \quad W = \begin{bmatrix} w_1 & w_2 \end{bmatrix}' \quad V = \begin{bmatrix} v_1 & v_2 \end{bmatrix}'$$

$$\Phi = \begin{bmatrix} 1 & 0 & T & 0 & T^2/2 & 0 \\ 0 & 1 & 0 & T & 0 & T^2/2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \Gamma = \begin{bmatrix} T^2/4 & 0 \\ 0 & T^2/4 \\ T/2 & 0 \\ 0 & T/2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

In this model (Eq. 12), two more elements, i.e., the accelerations are added to the state vector, and are modeled as constants disturbed by random noises. The system matrices are changed accordingly.

The correlated-acceleration (AA) model:

$$\Phi = \begin{bmatrix} 1 & 0 & T & 0 & T^2/2 & 0 \\ 0 & 1 & 0 & T & 0 & T^2/2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & \alpha_x & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha_y \end{bmatrix} \tag{13}$$

This model (Eq. 13) is the same as the CA model except the system matrix Φ. The accelerations are not constant. Instead, they are modeled as an AR(1) process with $A(k+1) = \alpha \cdot A(k) + r(k)$, where *A(k)* is the acceleration sequence and *r(k)* is a zero-mean Gaussian noise sequence. This is reflected in the system matrix by α_x and α_y as shown above. We proposed this model based on our observations in commercial videos. By examining the autocorrelation functions of the accelerations, we found that the accelerations can be modeled quite well by such an AR(1) model for both Type A and Type B videos.

Fig. 3 shows the frame-by-frame autocorrelation functions of accelerations (Fig. 3(a) for *x* direction. Fig. 3(b) for *y* direction) of a Type A video shot. There is a strong negative correlation between the accelerations of consecutive frames. Other examples show the same tendency, in both Type A and Type B videos. The noise sequence *r(k)* has a smaller variance than that of the original acceleration, which means this model fits well in real applications. We also test the autocorrelation functions of accelerations for every three frames (i.e., for I- and P- frames). The same pattern appears, but the magnitude of the negative correlation is smaller.

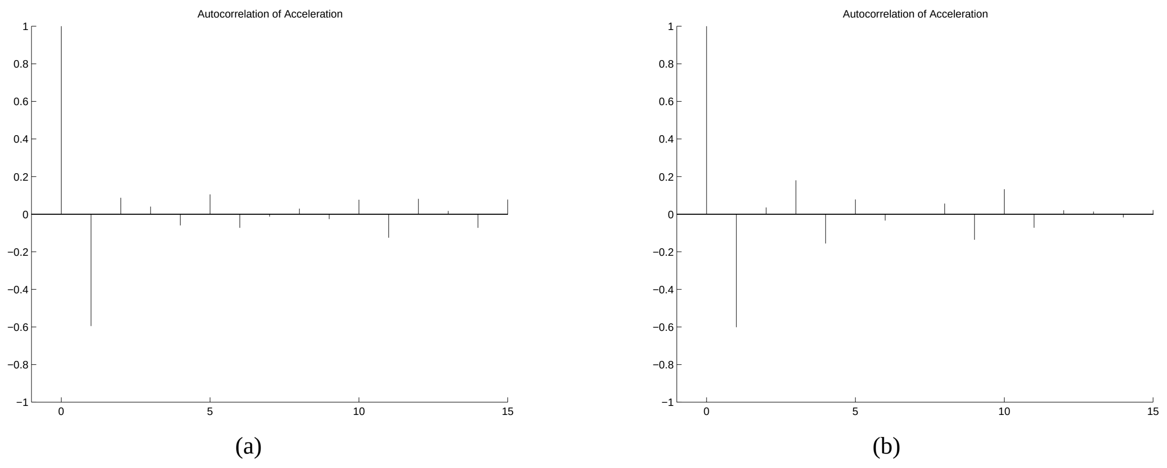


Fig. 3. Autocorrelation functions of accelerations of face movements in a video shot.

The variable dimension filter (VDF):

The VDF is a system that switches between a CV mode and a CA mode. The dimension of the state vector changes when the system changes mode, hence the name VDF[2]. It was proposed to address the problem of tracking highly maneuverable targets. The system detects maneuvers by looking at a fading memory average of innovations and switches to the CA model if a maneuver is detected. In the CA model, when the estimated accelerations are not significant it switches back to CV.

4.2. Comparison of different motion models

Table 1 shows an example of the tracking errors using various motion models on a Type A video. We use the ground truth of face locations for every I- and P-frame. Gaussian noise of different variances are added to simulate the observation data. We then plug in different motion models to the Kalman filter and track the faces. Mean distance of the tracking result to the ground truth and the maximum distance to the ground truth are recorded (both in the unit result to the of pixels). At each noise level, multiple runs of Gaussian sequences are generated and added to the ground truth. The results are averaged over the multiple runs of each noise level.

σ=2

Model	Mean-D	Max-D
CV	1.42	6.70
CA	1.50	6.70
AA	1.43	6.60
VDF	1.46	7.30

σ=4

Model	Mean-D	Max-D
CV	2.54	11.90
CA	2.79	14.40
AA	2.57	12.40
VDF	2.65	15.00

σ=16

Model	Mean-D	Max-D
CV	7.90	19.10
CA	8.01	18.70
AA	7.91	18.70
VDF	7.93	19.10

Table 1. Tracking errors of different motion models on a Type A video.

Fig. 4 shows face tracking errors for different models for the first 16 runs of noisy observations (σ=2.0). Consistently, the CV and AA models have the best performance, and the CA model usually has the worst. Similar results are obtained for observations with other noise level, as well as for Type B videos.

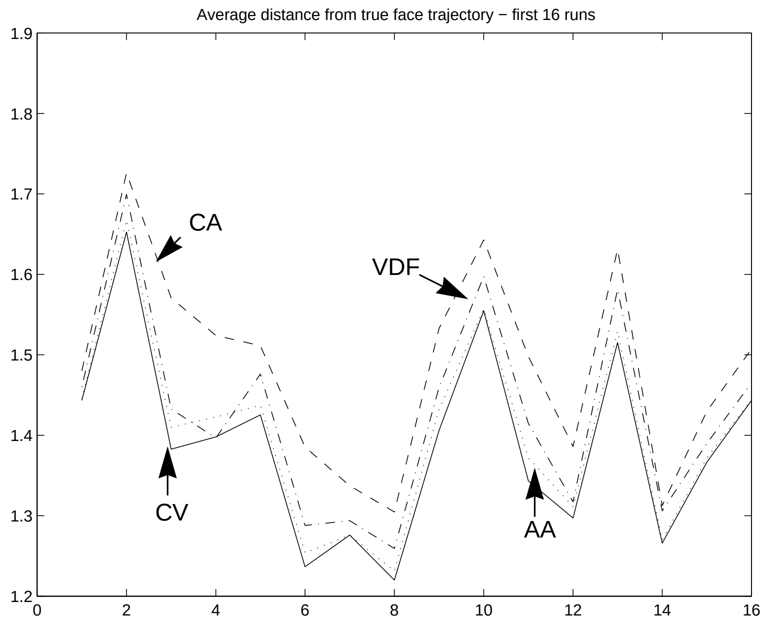


Fig. 4. Average tracking error of different motion models on a Type A video for the first 16 runs of noisy observations (σ=2.0).

4.3. Discussion

Experimental results have shown consistent performance of the four different motion models. The reasons that the CV model performs well are: a) The sampling interval is small (e.g., 0.1s), so that the face motion can be seen as piecewise rectilinear movements. b) Unlike highly maneuverable targets like a jet fighter, the face is not able to achieve very high accelerations. Based on our statistics, there are only about 15% of the time when a face can achieve directional acceleration greater than 3 pixels per sampling interval (0.1s). The AA model also performs well, because it fits well the nature of the face movements in commercial videos. The negatively correlated accelerations implies that faces have a tendency to “stabilize” themselves. A face seldom moves in an accelerating manner persistently. On the contrary, it usually keeps a relatively smooth motion and stays within the boundary of the video frame.

The CA model does not perform well because it is far from the real dynamics of faces. Rarely will we have a face that constantly accelerates in the video. If this happens, the face will move out of the video frame very quickly. The VDF makes decisions to switch between two modes based on accumulated information. The decision has a delay and may not detect the accurate boundary of mode changes, especially when the accelerating segments are brief.

In summary, for the purpose of face tracking in commercial video, when the sampling rate is fairly high, it suffices to use the simple CV model, or the AA model. The differences between models decreases as the observation noise gets stronger, which is intuitive because when the observation data becomes very noisy, the true kinematics of the moving faces are buried by the noise.

5. INTRA-SHOT FACE TRACKING

As stated in Section 3, we use a Kalman filter framework in the face tracking module. Fig. 5 shows the flow chart of the tracking algorithm.

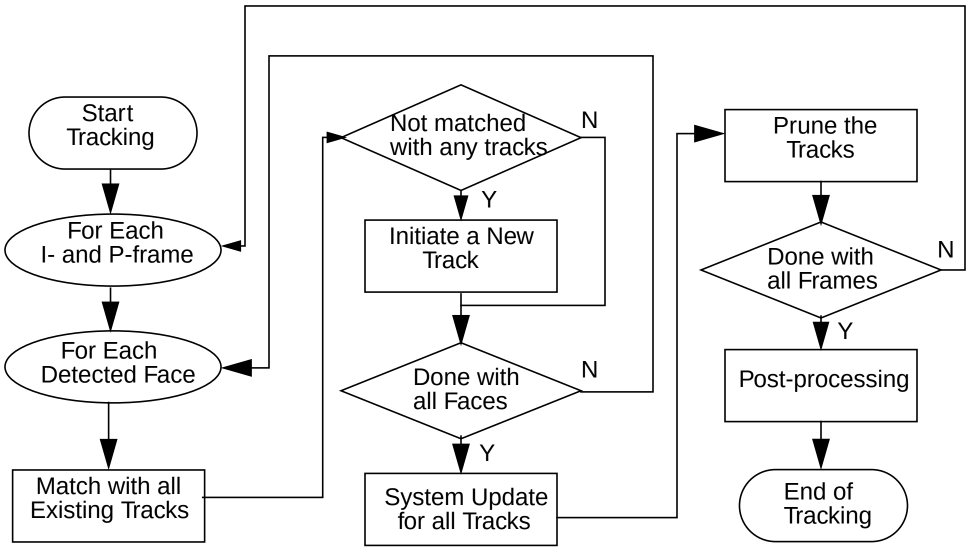


Fig. 5. The flow chart of the intra-shot face tracking algorithm.

In the matching stage, each face in the incoming I- or P-frame is matched against the existing tracks using the Mahalanobis distance (Eq. 10). Given the hypothesis that the *i*th face detected at frame *k* is an extension of track *j*, the Mahalanobis distance is defined in Eq. 14 (Note that we changed the notations a little from Eq. 10). $\gamma_{i,j}$ is n=2 chi-squared distributed. From the chi-squared distribution tables we know that $P(\gamma_{i,j}<d_m) = 0.95$ when d_m=5.99. We use d_m as the threshold of matching to eliminate unlikely extensions of existing tracks.

$$\gamma_{i,j} = [Z_i(k) - \hat{Z}_j(k|k-1)]'S_{i,j}(k)^{-1}[Z_i(k) - \hat{Z}_j(k|k-1)] \tag{14}$$

The modified log-likelihood function of each track is the sum of the Mahalanobis distances of all measurements that are assigned to this track, and can be computed recursively as in Eq. 15, with the hypothesis that the *i*th face detected at frame *k* belongs to track *j*. We use it as the cost function of each track, which means that tracks that minimize the sum of Mahalanobis distance are selected. One disadvantage of this cost function is that the effect of very old measurement stays so that it responds poorly to new observations. In practice, we use a modified version of the cost function as shown in Eq. 16, where *c* is a positive number smaller than one. In the case where no face is matched with the existing track, a penalty is added to the cost function (Eq. 17), the Kalman filter keeps going until the cost function goes above a threshold T_m.

$$\lambda_j(k) = \lambda_j(k-1) + \gamma_{i,j} \tag{15}$$

$$\lambda^M{}_j(k) = c \cdot \lambda^M{}_j(k-1) + \gamma_{i,j} \tag{16}$$

$$\lambda^M{}_j(k) = c \cdot \lambda^M{}_j(k-1) + p \qquad if \qquad \gamma_{i,j} > d_m, \forall i \tag{17}$$

The number of hypothesized tracks can grow exponentially if left unchecked. In the tracking pruning stage, we manage and contain the number of tracks as follows: a) Remove tracks with the cost function above the threshold. b) Merge tracks that have similar state estimates. c) Keep only the M-best tracks (in the sense of smallest cost functions). The post-processing stage uses domain knowledge and heuristics to further reduce tracking errors. For example, very short tracks (e.g., those no longer than three GOPs) are removed because they tend to be false alarms in normal videos.

Because of the complexity of commercial videos and the face detection errors, several scenarios may arise that are particularly challenging to face tracking.

1. Faces moving close: When two faces move close to each other, there is a possibility that both faces fall inside the validation region of the other face. Using the multiple hypothesis method, this ambiguity is resolved in later frames. This is illustrated in Fig. 6(a). At frame *k* four tracks are hypothesized, shown as the two solid lines and the two dashed lines. However, the dashed lines (false hypotheses) will soon become invalid because they lack supporting observations (shown as unfilled small circles) in later frames and the cost functions become too high. In practice, it is possible that the ambiguities cannot be resolved as quickly as shown in this example. To remove false tracks in this case, more observation data is necessary, and other criteria such as feature similarity should be used.

2. Faces occlude each other: This is an extreme case of the above scenario. When two faces collide, usually one face is occluded. There are two situations: either the face at the front will be detected (in Fig. 6(b), shown as the solid circle at frame *k*), or none of the faces will be detected (in Fig. 6(c), shown as the unfilled circle at frame *k*). If the occlusion lasts only for a short period of time, the faces can be tracked using the multiple hypothesis method. When the unoccluded face is detected, its visual features, along with associated body region features, are matched with those of the previous tracks and assigned to one of them. The occluded face is undetected, but its track is continued if the occlusion is short. If there is a long period of occlusion, the face tracks will be broken into segments with a gap in between. Visual feature similarities of the faces and associated body regions are used to bridge the gap and link corresponding face tracks.

3. Missed faces: The tracking module does not immediately stop a track when it detects a miss. Instead, the track is extended using the previous result and the Kalman filter algorithm continued, with a penalty added to the cost function. In this way, small gaps in the tracks caused by missed detection will be filled up. This is illustrated in Fig. 6(d). If a face is continuously missed for many frames, tracking will be difficult. The problem becomes, in essence, the same as the grouping of faces across video shots. In order to link the broken segments of face tracks, decoding of some frames is necessary.

4. False alarm of faces: Usually this happens sporadically and there is little temporal correlation between the false alarms. So, after tracking, they appear as very short track segments and will be removed at the post-processing stage.

5. Faces moving in and out: This is illustrated in Fig. 6(e). Note that the same person does not have to move back into the frame from the point it leaves the video frame. Again, the problem is like grouping faces across video shots, unless prior knowledge about the video content is available. For example, if we know there are no more than two people in this video shot, and person A is continuously tracked, we will be sure that this reappearing face is person B.

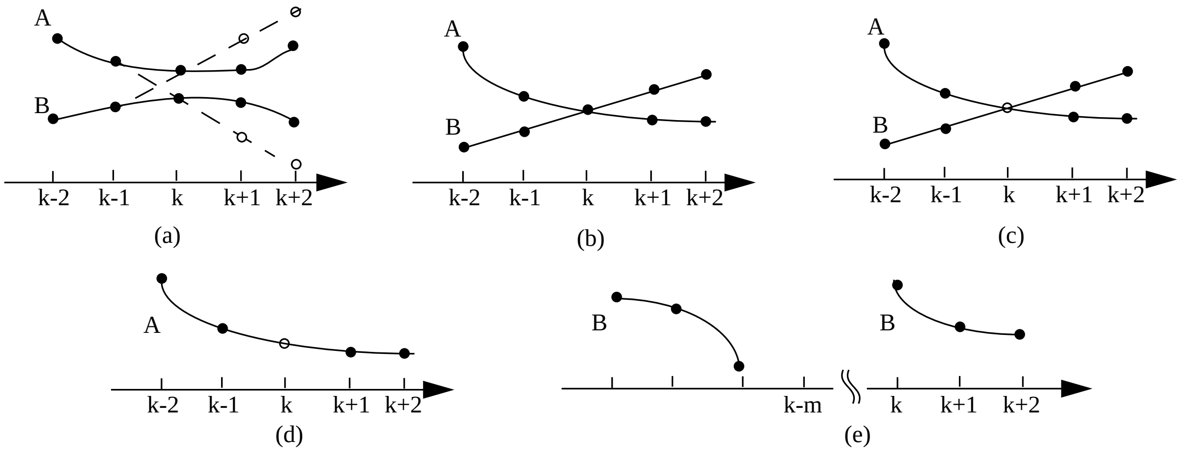


Fig. 6. Several challenging scenarios for face tracking.

6. FACE SUMMARIZATION ACROSS SHOTS

After faces are tracked within each video shot, faces of the same person across video shots can be grouped together, and the video sequence summarized according to the faces. First of all, we need to select representative frames for tracked faces in each video shot. Frames containing large, frontal-view faces are selected. For faces with high motion, frames that have faces at the turning points of their trajectories are also selected to roughly represent the motion of the faces.

The human faces have the same structures and similar colors. It is very difficult, if not impossible, to differentiate faces of different persons directly from the compressed domain. Therefore, once the representative frames are selected, we decode them into the pixel domain for further analysis. Visual features such as color and texture are extracted for the face regions, along with associated body and background regions. Various clustering algorithms can be applied to group the faces into different persons. We are at the early stage of the investigation into cross-shot grouping of faces. Based on some initial experiments, we found that for videos like *Larry King Live* where there are a limited number of prominent faces, using color histograms of the face and body regions can differentiate different people in a session (usually not more than 6 or 7 people). We realize that much more needs to be investigated for grouping faces across video shots, especially when the number of people grows large. Besides common visual features, we may also need to extract facial features from the tracked faces. Domain knowledge and rules about the videos plays an important role in the face summarization task. For example, if we know that this one-hour video is just one *Larry King Live* session, we are almost sure that the guests will not change their clothes in this video sequence. This is not true if the video is one episode of a sitcom. Obviously, for different application domains we need to consider this factor and adjust the technical approach accordingly.

Another important issue is how to visualize the summarized faces results to the end user. What is the best way to use these results to help the user understand the video content? How can we preserve the dynamic structure of the video with such a brief summary? The answers to these questions are crucial in designing future applications.

7. SIMULATION RESULTS

In this section, we present simulation results of the intra-shot face tracking algorithm. It is a compressed domain approach. These results show that using the Kalman filter framework and multiple hypothesis tracking, face detection errors and ambiguities caused by the presence of multiple faces are greatly reduced. We give three examples that illustrate the effectiveness of this method. Each video shot is MPEG-1 compressed, with frame size of 352x240 pixels. We use the CV model for the Kalman filter, and set up the thresholds as $d_m = 5.99$, $c = 0.8$, $p = 7.2$, $T_m = 8.8$ (see Eqs. 15-17).

The first video shot ("Marcia") is from the CNN news video. It has 556 frames, which includes 38 I-frames. A single face is present from the beginning to the end. However, there are 2 misses and 6 false alarms. Several frames from the video shot are shown in Fig. 7, with the face detection results overlaid.

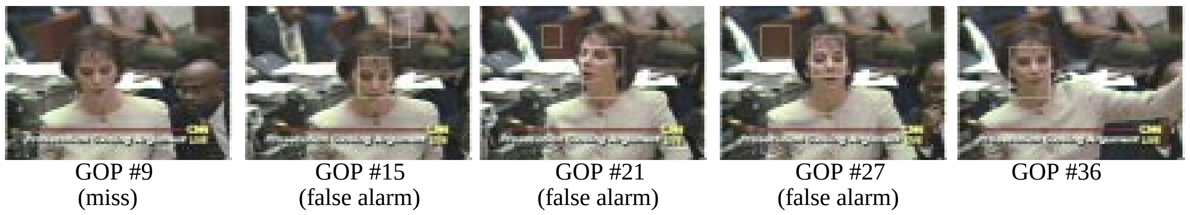


Fig. 7. Sample frames from video "Marcia" with face detection results.

The face tracking result can be illustrated using the graph in Fig. 8. T1 through T6 are the tracking result before the post-processing stage. T2 is the only valid track because the other tracks are all too short (no longer than 3 GOPs).

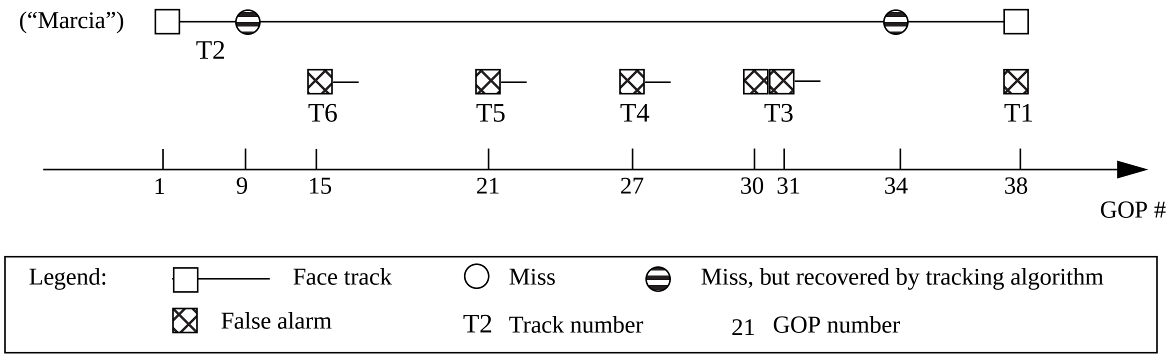


Fig. 8. Face tracking result for "Marcia".

The second video shot ("Clinic") is from the VideoQ database[3]. It has 361 frames, of which 25 are I-frames. There are 2 misses and 2 false alarms of face detection. Moreover, one face ("Doctor") moves out of the video frames twice. Fig. 9 shows sample frames from this video, again with face detection results.

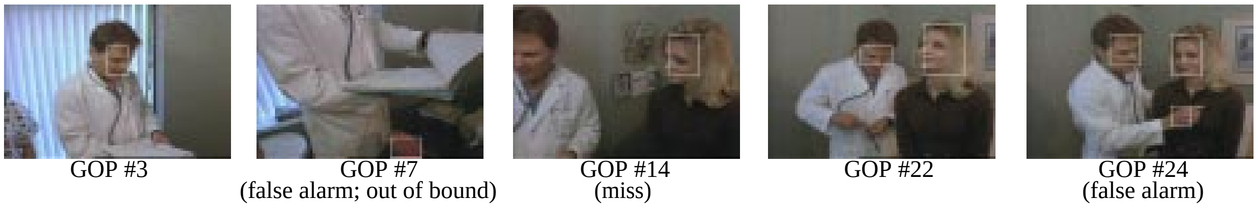


Fig. 9. Sample faces from video "Clinic" with face detection results.

The face tracking result is shown in Fig. 10. T1, T3, T4, and T5 are the remaining face tracks after post-processing. It is clear that T1 is a different person from T4 and T5, but it can not be determined from the tracking results whether T3, T4, T5 are the same person, or whether T3 and T1 are. This is because the "Doctor" moves in and out of the video frames and causes large gaps between face tracks of himself. If we assume that this video shot contains no more than two persons, we can use the compressed-domain color features of the body regions to link tracks T3 through T5. Otherwise, we need to decode several frames and resort to pixel-domain techniques.

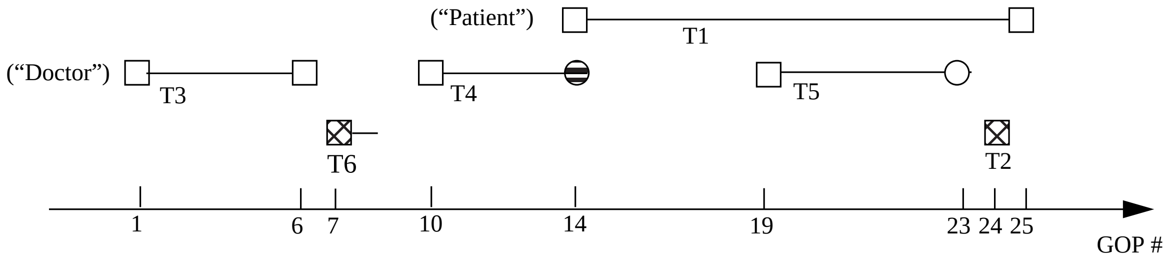


Fig. 10. Face tracking result for "Clinic".

The third video shot ("Couple") is also from the VideoQ database, with 286 frames, of which 20 are I-frames. There are two faces in the video that move very close and occlusion occurs (for about 2-3 GOPS). There are 8 misses and 10 false alarms of face detection. Fig. 11 shows sample video frames of "Couple" with face detection results.

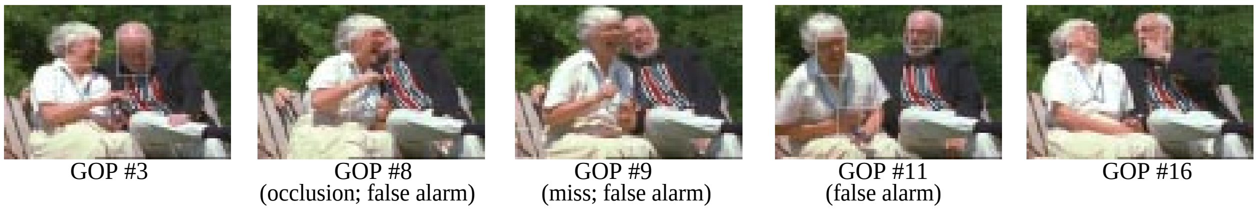


Fig. 11. Sample faces from video "Couple" with face detection results.

Fig. 12 shows the face tracking result in this video shot.

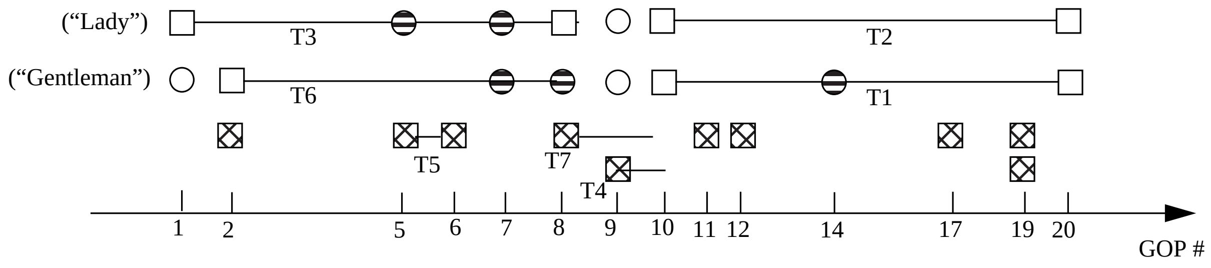


Fig. 12. Face tracking results for "Couple".

In Fig. 12, there are a few very short false alarms (1 GOP) that are removed directly from the track list. T4, T5, and T7 are short tracks (2 or 3 GOPs) and are removed as well. The maximum occlusion of the "Gentleman" happens at GOP #8 (see Fig. 11), so that the two GOPs around it are greatly interfered and the faces are not detected. However, the two tracks (T3 and T6) still continue to GOP #8 regardless of the misses at GOPs #7 and #8, with the detected face in GOP #8 assigned to T3 (because of the matched features in the body region), and T6 continued based on its previous trajectory. Both faces are missed in GOP #9, and a gap is formed between the face track segments. Since the gap between these truncated tracks is small, the color features of the face and body regions are matched to link T3 with T2, and T6 with T1. Note that none of the false alarms in GOPs #8 and #9 are linked with previous tracks.

From the above examples, we see that the tracking algorithm works well on faces detected in continuous frames. If there are a few sporadic misses in the detection result, the algorithm keeps working and is able to correctly track the face. In the case that continuous misses for many frames (either due to long occlusion, faces moving out of bound, or the error of face detection), the algorithm may end the track and start a new track when the detected face reappears. False alarms will start a new track which terminates quickly or is too short in the context of commercial videos. To achieve optimal results in face tracking, it is necessary to decode certain video frames for analysis to resolve ambiguities that cannot be resolved in the compressed domain. Specific domain knowledge of the video content and applications improves tracking results.

8. CONCLUSION AND FUTURE WORK

In this paper we introduce a face tracking and summarization system (FaceTrack) that works on the compressed video. We present the system architecture, the face tracking framework using the Kalman filter, and the face tracking algorithm.Various motion models for the Kalman filter are compared to see their effects on the tracking of video objects. We found that for commercial videos with moving faces, the CV and the AA models are usually good enough. Simulation results are presented to show the effectiveness of the face tracking algorithm in reducing face detection errors and resolving tracking ambiguities. We also discuss approaches for face grouping over video shots.

Future research will focus on optimizing the face tracking algorithm in adverse situations, investigating face summarization techniques based on the tracking results, and the integration of face summarization with other video techniques like video scene classification.

REFERENCES

1. A. V. Balakrishnan, *Kalman Filtering Theory*, Optimization Software Inc., Publications Division, 1987.
2. Y. Bar-Shalom and K. Birmiwal, “Variable dimension filter for maneuvering target tracking,” *IEEE Trans. on Aerospace and Electronic Systems*, Vol. AES-18, No. 5, pp. 621-628, Sept. 1982.
3. S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, “VideoQ – an automatic content-based video search system using visual cues,” in *ACM Multimedia Conference*, Nov. 1997, Seattle, WA.
4. I. J. Cox, “A review of statistical data association techniques for motion correspondence,” *Intl. J. of Computer Vision*, Vol. 10, No. 1, pp. 53-66, 1993.
5. T. Jebara and A. Pentland, “Parametrized structure from motion for 3D adaptive feedback tracking of faces,” in *IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997, pp. 144-150.
6. J. Meng, Y. Juan, and S.-F. Chang, “Scene change detection in a MPEG compressed video sequence,”, in *Proc. SPIE*, Vol. 2419, pp.14-25.
7. H. A. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, pp. 23-38, Jan. 1998.
8. D. Schonfeld and D. Lelescu, “VORTEX: Video retrieval and tracking from compressed multimedia databases - template matching from MPEG2 video compression standard,” in *SPIE Conference on Multimedia and Archiving Systems III*, Nov. 1998, Boston, MA.
9. H. Wang and S.-F. Chang, “A highly efficient system for automatic face region detection in MPEG video,” *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 7, No. 4, pp. 615-628, Aug. 1997.
10. ISO/IEC 13818 - 2 Committee Draft (MPEG-2).