A Sensor Fusion Based Object Tracker for Compressed Video

Radhakrishna Achanta  
National University of Singapore  
School of Computing  
Singapore-117543  
(65)68744925  
achanta@comp.nus.edu.sg

Jun Wang  
National University of Singapore  
School of Computing  
Singapore-117543  
(65)68744925  
wangj@comp.nus.edu.sg

Mohan Kankanhalli  
National University of Singapore  
School of Computing  
Singapore-117543  
(65)68746738  
mohan@comp.nus.edu.sg

Abstract—Object tracking is very important for automatic indexing of video content. This work shows tracking of objects directly using compressed MPEG video data. Two sensors, one using motion vectors and the other using DCT coefficients obtained from compressed video stream, provide measurements for the location of the object being tracked. The optimal estimate from the two measurements is found using Kalman filtering based state vector fusion approach.

Keywords—MPEG compressed domain; Kalman filtering; sensor fusion

1. INTRODUCTION

The need to manage large amounts of video data like home video in a fast and easy manner have led to research focus on the development of video indexing systems. In order to provide versatile query based retrieval, the content of the shots (objects and events of interest) needs to be indexed. Objects can only be indexed with the help of tools that can track the object through the sequence of frames in a video. The object tracker presented here does that, serving as the annotation tool for the DIVA project [http://diva.comp.nus.edu.sg:8080].

2. RELATED WORK

There are several instances of object trackers in the pixel domain, some of which may not be suitable for practical applications because of their computational overhead. There are instances of tracking work done in compressed domain too. Motion vectors are used in [1] for tracking. In [2] object tracking is treated as a macroblock-linking problem; macroblocks are linked with the help of motion vectors and Discrete Cosine Transform (DCT) AC coefficient energies. Directed graph approach is taken in [3] where objects are tracked using DC difference images.

Unlike previous work in compressed domain, this paper presents an object tracker that relies on DCT color information for the color sensor; and uses both forward and backward motion vectors for the motion vector based sensor. For the two sensors mentioned, this paper uses a variation of the algorithm presented in [4]. The earlier work in [4] was found not to be very accurate. This motivated use to consider the sensor fusion approach for obtaining better tracking results. To this end, Kalman filtering based state vector fusion is used for sensor fusion.

3. SENSOR FUSION AND TRACKING

In this paper, in order to make more robust and smooth object tracker, we assume the measurements coming from our previous object trackers (color based tracker and motion vector based tracker) have noise and try to fuse these two noise measurements together to get optimal combined results. In our sensor fusion approach, given the fact that two trackers are ‘noisy’ measurements, we consider them to be two sensors. A state vector fusion method based on Kalman filter is proposed to fuse these two sensors to achieve optimal results. Before we explain the algorithms itself, a brief introduction to Kalman filtering as sensor fusion method is in order.

3.1 Kalman filtering for sensor fusion

Kalman filtering provides a good mechanism to estimate (or predict) states or fuse ‘noisy’ measurements from sensors by minimizing the mean square error (MSE) in radar tracker field for a long time. In our work we use Kalman filter for the purpose of estimation in non-intracoded frames as well as for sensor fusion in I frames.

3.1.1 Kalman filtering

A discrete-time dynamical system can be presented using graph model (Dynamical Bayesian Networks (DBNs)) as shown in Figure 1. In Figure 1, X (k) and X (k+1) are true states in time k, and time k+1 respectively, which are hidden units and cannot be observed. Z (k) and Z (k+1) are measurements of true state X in time k, and time k+1.
respectively and are observed. In this dynamical system, the task is to perform inference, that is to compute the belief state \( P(X(k)|Z(1):Z(k)) \) (estimate the state of \( X(k) \) utilizing the observations from \( Z(1) \) to \( Z(k) \)). In order to develop an efficient algorithm to compute \( P(X(k)|Z(1):Z(k)) \), the dynamical system is often assumed to be linear and subject to Gaussian noise. This model, called Kalman filters, can be defined as

\[
x(k + 1) = \Phi(k)x(k) + \Gamma(k)w(k) \tag{1}
\]

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

\[
z(k) = H(k)x(k) + v(k) \tag{2}
\]

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, \( \Gamma(k) \) the system noise coefficient matrix; \( H(k) \) is the measurement or observation matrix; \( w(k) \) and \( v(k) \) represent the process and measurement noise respectively. They are assumed to be independent of each other, white (zero mean), and with normal probability distributions. As shown in Figure 1, when causal-effect arrows are assumed to be governed by equation 1 and 2 (i.e., the transition from \( X(k) \) to \( X(k+1) \) is assumed to be linear and governed by process equation 1, the relation between the hidden state \( X(k) \) and its measurement \( Z(k) \) is modeled by equation 2), the Dynamical Bayesian Networks become Kalman filter. One of the main advantages of this model is that this algorithm is efficient and practical since typically the parameters of the model are assumed to be time-invariant. Many successful applications in radar tracking field have already proven that.

\[\text{Figure 1 Graph Model for Dynamic system. X(k) is state to be estimated in time k (can not be observed), Z(k) is the measurement of X(k) at time k. This system can be modeled by kalman filters when arrows are governed by equation 1 and 2.}\]

In our particular domain, \( X(k) \) presents the object positions (represented as locations or velocity of movements) in frame \( k \) and \( Z(k) \) presents the sensor measurement of \( X(k) \) from some features such as color, motion vectors and etc. in compressed domain in frame \( k \). When we only have one noisy measurement \( Z(k) \), the aim is to estimate the \( X(k) \) from \( Z(1) \) to \( Z(k) \) using Kalman filter model. We use Kalman filter for the purpose of estimation when only one sensor is available (There is only motion vector based object tracker available in non intracoded frames). However, in the I frames, there are two sensors (color based object tracker and motion vector based object tracker) available, sensor fusion method of Kalman filters has to be used.

\[\text{Figure 2 Graph Model for State Vector Fusion. There are two Kalman filters coming from color based and motion vector based object trackers respectively.}\]

3.1.1 State Vector Fusion

Kalman filters can also be used for sensor fusion purpose, when there is more than one ‘noisy’ measurement available, in contrast to be used for estimation in the case of one measurement as shown in previous section. To fuse sensors, two methods can be used: State Vector Fusion and Measurement Fusion. Measurement fusion needs the assumption that individual measurement (sensor) has to be independent. However in our specific domain, the measurements from two individual object trackers (color and motion vector based object trackers) are related and not likely to be independent. It causes the independent assumption does not exist any more. Our experiments on [5] have also proved that state vector fusion is better than measurement fusion in object tracker domain. Therefore in this paper, based on Kalman filter, we use state vector fusion approach.

In the case of two sensors coming from color, and motion vector, the State Vector Fusion Scheme can be shown as Figure 2.

\[\text{In our particular domain, X (k) presents the object positions (represented as locations or velocity of movements) in frame k and Z (k) presents the sensor measurement of X (k) from some features such as color, motion vectors and etc. in compressed domain in frame k. When we only have one noisy measurement Z(k), the aim is to estimate the X (k) from Z(1) to Z(k) using Kalman filter model. We use Kalman filter for the purpose of estimation when only one sensor is available (There is only motion vector based object tracker available in non intracoded frames). However, in the I frames, there are two sensors (color based object tracker and motion vector based object tracker) available, sensor fusion method of Kalman filters has to be used.}\]
In Figure 2, there are two Kalman filters used for two sensors (color based sensor and motion vector based sensors) respectively. In this graph, two more types of hidden units \((X_c(k) \text{ and } X_c(k+1))\) for state value of color based sensor in time \(k\) and \(k+1\) respectively, \(X_m(k) \text{ and } X_m(k+1)\) for state valued of motion vector based sensor in time \(k\) and \(k+1\) respectively). Hidden units \(X(k)\) and \(X(k+1)\) represent the final true state, which mainly influenced by \(X_c\) and \(X_m\). Our aim here is to get the estimation \(X(k|k)\) of \(X(k)\) in time \(k\).

Using these two Kalman filters, hidden state \(X_c(k)\) can be estimated as \(X_c(k|k)\) by finding maximum \(P(X_c(k)|Z_c(1):Z_c(k))\) utilizing measurements \(Z_c\) from color based sensor, and hidden state \(X_m(k)\) can be also estimated as \(X_m(k|k)\) by finding maximum \(P(X_m(k)|Z_m(1):Z_m(k))\) utilizing measurements \(Z_m\) from motion vector based sensor.

By using maximum likelihood as a fusion strategy, the best estimated fused data \(X(k|k)\) is derived from equation (3).

\[
X(k|k) = X_c(k|k) + \hat{P}_c(k|k)\hat{P}_E^{-1}(k|k)[X_m(k|k) - X_c(k|k)]
\]

(3)

\[
\hat{P}(k|k) = \hat{P}_c(k|k) - \hat{P}_c\hat{P}_E^{-1}(k|k)\hat{P}_c(k|k)\hat{P}_m(k|k)
\]

(4)

where, \(X(k|k)\) is the best estimated fused data in time \(k\), \(P_c\) and \(P_m\) are estimated state vector’s covariance matrix of Kalman filters for color based sensor and motion vector based sensor respectively, in time \(k\), and \(X_c(k|k)\) and \(X_m(k|k)\) are the estimated state vector in Kalman filter for color and motion vector respectively. The covariance matrix of fused results \(P(k|k)\) is obtained using equation (4).

3.2 Kalman filter based sensor fusion for object tracker

In compressed domain, there are two features that can be used for tracking purposes. One is Motion Vectors, the other is DCT Coefficients. Motion vectors are for coding purposes, so they are not reliable for object tracking. Any tracking errors due to motion vector based tracking prove to be cumulative. Color information coming from DCT is more reliable, but poorly indicates image region of interest because of the blocky nature of compressed data and can only be used in I frame (will discuss in late section). In I frame, since the results from two sensors are all variable, as shown in figure 2, a sensor fusion based tracker can be constructed by taking measurements coming from color (DCT) based object tracker and motion based object tracker as measurements \(Z_c\) and \(Z_m\), and the true positions of the object as hidden unit \(X\). So the measurements from the two sensors are fused at the I frames using the state vector fusion approach to overcome these drawbacks. This overcomes the problem of blockiness of DCT information in I frames, making color based tracking smooth, and at the same time serves a verification/correction mechanism for motion vector based tracking, preventing error accumulation. In other non-intracoded frames, since only motion vector based sensor is available, we just use Kalman filter as estimator to get more smooth results. The working of the two sensors is presented below (further details in [4]). Parameters setting of Kalman filter will be discussed afterwards.

3.2.1 Motion Vector Based Tracking sensor

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.

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

\[
R_p \leftarrow \text{Translate}(R_p, \text{ Mode (VM(R_p) )})
\]

(5)

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

3. To cross GOP boundary, the image region on the new I frame is obtained again using eq. 5, 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.
The velocity of size change can be defined as:

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

The velocity of moving movement of the object can be defined as:

\[ x' = (x(k+1) - x(k))/T \] (8)

Here the velocity of zooming and moving movement is assumed to be constant since the constant velocity model has best performance for commercial videos sampled at high frame rates ([28]). Therefore, \( x(k) \) state-vector is defined as in equation (9) and other parameters in processing equation (1) are defined as in equation (10).

\[
\begin{align*}
\mathbf{x}(k) &= \begin{bmatrix} x_{\text{centroid}} & y_{\text{centroid}} & \text{width} & \text{height} & \dot{x}_{\text{centroid}} & \dot{y}_{\text{centroid}} & \dot{\text{width}} & \dot{\text{height}} \end{bmatrix}^T \\
W &= \begin{bmatrix} w_x & w_y & w_w & w_h \end{bmatrix}^T \\
\Phi &= \begin{bmatrix} 0 & 0 & T & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & T & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & T & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
\Gamma &= \begin{bmatrix} T/2 & 0 & 0 & 0 \\
0 & T/2 & 0 & 0 \\
0 & 0 & T/2 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \end{bmatrix}
\end{align*}
\]

3.2.2 Color Based Tracking Sensor

At the I starting 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 3). For this, a window of the size of the object is moved at a time over the entire search area, and the candidate histograms like the ones made for the reference area are made. Finding the best match involves finding the minimum value of DiffSum:

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

where HDiffCr and HDiffCb are the histogram bin differences for Cr and Cb values respectively; W[n] is the set of weights used \{.4, .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 color information is given maximum weightage, followed by lower frequency AC values, which convey coarse texture or shape information ([1]), 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. 6, it is noteworthy that AC information proves very useful in distinguishing between two objects with the same color, 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.

3.2.3 Parameter Settings

3.2.3.1 Modeling the Object Dynamics

Objects have mainly two types of movements in the video-rotation and translation. In addition there can be a change in size because of camera zoom. We try to take into account the change of size of object too, in addition to translation. Simply, we use rectangle to encode this movements in frames. Initially, user specifies the object of interest by rectangle to locate the position of the object. The output of the object tracker is the rectangle along with the video to indicate the positions of the object in the video. By using rectangle, the change in size of object and translation of object are modeled in the following:

The velocity of size change can be defined as:

\[ \text{width}' = (\text{width} \ (k + 1) - \text{width} \ (k))/T \] (7)
MN = \left[ \sum_{N} (X - X') \cdot (X - X')^T \right] / N \quad (11)

Here X is the column vector containing the difference values of ground truth and measured values obtained from tracking; X' is the column matrix containing the mean value of N such column vectors for N number of frames (for all test the videos put together).

4. EXPERIMENTS AND RESULTS

Experiments were carried out on standard test video clips of used in similar research, and on MPEG-7 test video clips. For comparison, the ground truth was obtained by manually drawing rectangles over the desired object region in each frame of the video clips used.

Figure 4 shows screen shot of GUI for our object tracker. Initially, user manually marks object of interest by rectangle. Tracker tracks along with the video and give the results.

As an example, one of MPEG 7 test set, called “Girl on Stage” (MPEG 1 format, 564 frames) is shown on the screen shot with the tracking results obtained by our approach in Figure 5.

Figure 6 shows tracking results of two approaches compared with ground truth on video “girl on stage”. Figure 6(a) shows our previous object tracker, which just uses motion vector and color information. Figure 6 (b) shows results of our sensor fusion approach. Both are work well. However, sensor fused results perform slightly better than previous one. One can clearly see from Figure 8.

5. CONCLUSION

Compressed domain tracking presented here faces a few inherent problems (while overcoming certain ones like speed). Motion vectors are for coding efficiency, and not to help tracking; motion vectors can not be completely relied on, as their presence or absence largely depends on the compression algorithm’s requirements. Color information available in compressed domain is usually sub-sampled as most videos are of 4:2:0 format. Also, this data is ‘blocky’ in nature, making the less accuracy than some pixel domain method. unless DCT reconstruction methods are adopted (but that might defeat the purpose of compressed domain processing due to the computational overhead involved). In addition, for small and/or fast moving objects, the motion and color information available per frame does not allow good tracking. This is where a fusion approach like Kalman filtering is required to minimizes inaccuracies and smooth the results.

By applying Kalman filters to estimate in non-intracoded frames and fuse sensors in I frames, a more smooth and robust results are obtained. This encourages us to keep going in this direction by trying more general models, such as general Dynamical Bayesian Networks, nonlinear estimator.
6. REFERENCES


Figure 6(a) Tracking results compared with ground truth without using sensor fusion approach.
Figure 6 Tracking results compared with ground truth using sensor fusion

Figure 7(a) Tracking Error without using sensor fusion approach.

Figure 7(b) Tracking Error using sensor fusion approach.