Object Tracking Using Background Subtraction and Motion Estimation in MPEG Videos

Ashwani Aggarwal, Susmit Biswas, Sandeep Singh,
Shamik Sural, and A.K. Majumdar

Indian Institute of Technology, Kharagpur,
West Bengal -721302, India
shamik@sit.iitkgp.ernet.in, akmj@cse.iitkgp.ernet.in,
Ashwani.Aggarwal@iitkgp.ac.in

Abstract. We present a fast and robust method for moving object tracking directly in the compressed domain using features available in MPEG videos. DCT domain background subtraction in Y plane is used to locate candidate objects in subsequent I-frames after a user has marked an object of interest in the given frame. DCT domain histogram matching using Cb and Cr planes and motion vectors are used to select the target object from the set of candidate objects. The target object position is finally interpolated in the predicted frames to obtain a smooth tracking across GOPs.

1 Introduction

Visual content in a video can be modeled as a hierarchy of abstractions. At the lowest level are the raw pixels with color information leading to lines, curves, edges, corners and regions. At the highest level are the human level concepts involving one or more objects and relationships among them. The first step in high level video processing is to identify the objects present in a scene. The next step is to see how these detected objects move with respect to each other. The above two problems combined, can be termed as "Object Tracking".

An important application of object tracking is video surveillance [1]. Airports, train stations, departmental stores, religious places, courts and public buildings are only a few examples of places where video surveillance has an extremely high priority. In addition to this, military, astronomy, navigation, road/air traffic regulation, medical imaging, augmented reality and robotics are some of the other major applications of object tracking [2],[3],[4].

There are primarily two sources of information in video that can be used to track objects: visual features (such as color, texture and shape) and motion information. A typical strategy is to segment a frame into regions based on color and texture information first, and then merge regions based on similar motion subject to certain constraints such as adjacency [5]. Extraction of these two types of information can be done either in the pixel domain or in the compressed domain. Tracking in videos especially from large databases, requires an enormous

P.J. Narayanan et al. (Eds.): ACCV 2006, LNCS 3852, pp. 121–130, 2006.

amount of processing since the videos are usually stored in a compressed form and must be decompress every time any processing is done on the video frames for object tracking. In order to reduce computational time and save processing resources, it is imperative that the tracking takes place in compressed domain itself.

Sukmarg and Rao [6] performed object detection and segmentation using color clustering, region merging based on spatiotemporal similarities, and background/foreground classification. The features extracted from the blocks of segmented object in compressed domain are used for fast object tracking.

Mezaris et al [7] exploit the motion vectors available directly from the MPEG stream for object tracking in the compressed domain. Park and Lee [8] use tracking using Mean Shift algorithm along with motion vectors. Yoo and Lee [9] suggested it is not possible to get sequential motion flow directly between compressed B-frames. Hence some kind of interpolation is normally used. Kartik et al [10] perform a block matching technique over the frames. This is combined with an adaptive block based approach for estimating motion between two frames.

In this paper, we propose a novel object tracking technique from MPEG videos. We employ a background subtraction method in the compressed domain using DC values of the Discrete Cosine Transform (DCT) coefficients of luminance blocks in the I-frames. To distinguish a target object from a set of candidate foreground objects, we use histogram comparison on color components in I-frames (Cb and Cr blocks) and distance between centroid of the candidate object and projected object using motion vectors. The object positions in the intermediate P and B-frames are obtained by interpolation.

In the next section we describe the motion estimation, background subtraction and interpolation algorithm in detail. We present the experimental results in section 3. The conclusions are drawn in the last section of the paper.

2 Object Identification and Interpolation

The proposed scheme for object tracking is mainly concerned with video surveillance applications where the camera is assumed to be fixed with a fairly wide angle of view. The algorithms presented in this paper consider supervised tracking of objects in which a user marks an object in an I-frame. The marked object is tracked by our algorithm in subsequent frames till the object disappears from the field of view of the camera. Such applications typically have a model of the background which is effectively used in our approach for identification and tracking of objects. It should be noted that, although the background is considered to be fixed, our system is robust in the presence of variations in the lighting conditions and extraneous movements.

Our method for object tracking can be divided into four broad steps, namely, background subtraction, candidate object identification, target object selection and motion interpolation. All the four steps are executed directly on MPEG compressed videos. We consider a typical 12-frame Group-of-Pictures (GOP) sequence: IBBPBBPBBPBB in our discussions.

2.1 Motion Estimation

In an MPEG video, the I-frames are intra coded while the P and B frames are motion compensated and DCT is carried out only for error components after motion estimation. The macroblocks for which motion estimation cannot be carried out are also stored as intra coded macroblocks in these predicted frames.

Since we consider supervised tracking, that is, a user marks an object in an I-frame which is subsequently tracked by the system, let us assume that the user chosen pixel domain area of the object is covered by $A_p[(x_{min}, y_{min}), (x_{max}, y_{max})]$ and the equivalent compressed domain area of the object covered be given by $A_c[(p_{min}, q_{min}), (p_{max}, q_{max})]$.

We parse the next predicted frame (whichever P or B) as per the frame sequence and extract motion vectors of all the macro blocks, which are in A_c. If there are 'n' macroblocks in A_c, we get a set of n vectors (right, down) corresponding to the macroblocks. These n motion vectors do not actually provide the real optical flow since they are often inaccurate and hence, some filtering is needed. In our work we use a 'Mode filter' in order to straighten up noisy vectors and thus eliminate this problem.

We calculate the 'Mode Motion Vector' through the extracted set of 'n' forward motion vectors. This is in reference to the previous I/P frame. It gives the displacement of the window consisting of the tracked object in the current predicted frame (B/P) frames and the window coordinates are updated accordingly.

The window enclosing the object keeps updating its coordinates as the predicted frames keep coming and the tracked object moves frame by frame. This continues till the time a new reference frame within the same GOP is encountered. When a new reference frame within the same GOP is encountered, the target window in the new reference frame becomes the updated reference window. All subsequent predicted frames track the object based on the motion vectors held by them as dictated by the latest reference frame.

The above process continues till the 1^{st} I frame of the next GOP arrives. Normally in the bit stream order of a compressed video, the I frame of a GOP is followed by the B frame of the previous GOP, which are backward predicted, from this I frame. Also since the I frame does not have any motion vectors, we use the 'Backward Motion vector' of the last B frame in the display order to track the object into the I frame of the following GOP.

We consider the window of object position of the last B frame. The backward motion vectors of the macroblocks in this area give an approximate position of the window. Let the macroblock in the previous I frame (i_{th} frame in the sequence) be denoted by the term $P^i_{j,k}$, where (j,k) is the position of the macroblock in the frame. So,

$$P^m_{i,j} = P^n_{x,y} - MV((x-i),(y-j)) \tag{1}$$

Here m_{th} frame is a B frame in reference to the n_{th} I frame being described by the motion vector ((x -i), (y -j)). We can get the position of this macroblock by

$$P^n_{x,y} = P^m_{i,j} + MV((x-i),(y-j)) \tag{2}$$

To cross GOP boundary, the image region on the new I frame is obtained again 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.

2.2 Background Subtraction

We use only the successive I-frames for tracking in our algorithm and thereafter interpolate the object motion in the intermediate P and B frames. We initially acquire a DCT image of an I-frame representing the background, which is used as the reference image. Thereafter, all subsequent DCT images are compared with this reference image to segment the foreground object. The background image is based on the model of the application and is updated from time to time whenever there is a permanent change in the background.

Out of the three-color components Y, Cb and Cr, we read the DCT values of only the Y plane from an MPEG file and consider DC values of all the macroblocks contained in the background frame. These DC values contain the average luminescence for the entire frame at the macroblock level. Thus, we effectively create a DC image of the frames under comparison with only the Y component taken into consideration.

Let $I_k(M, N)$ be the k^{th} I-frame in a video sequence having a height of M pixel and width of N pixel and let the DC image of luminance blocks of this frame be denoted as $I_k^{DC,Y}(M/8, N/8)$. Width and height of this Y DC image is reduced by a factor of 8 as compared to the original I frame, since only 1 DC value is used to represent 8X8 pixel macroblock. The value of the $(i, j)^{th}$ element of the luminance DC image is given by

$$I_k^{DC,Y}(i,j) = \frac{C_u C_v}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} I_k^Y(8i+x, 8j+y) \tag{3}$$

The luminance DC image of the background frame I_0 can be similarly determined. If an object has been marked in the k^{th} I-frame, we identify its position in the $(k+1)^{th}$ I-frame by subtracting $I_{k+1}^{DC,Y}$ from $I_0^{DC,Y}$. Thus, we obtain a difference image in the form of block-wise absolute difference of the Y DC background frame and the Y DC $(k+1)^{th}$ frame. The difference image can be represented as $\triangle I_{k+1}^{DC,Y}(M/8, N/8)$ where the value of the $(i, j)^{th}$ element is given as follows

$$\triangle I_{k+1}^{DC,Y}(i,j) = \begin{cases} I_{k+1}^{DC,Y}(i,j) \text{ , } if I_{k+1}^{DC,Y}(i,j) - I_0^{DC,Y}(i,j) \geq T \\ 0 \qquad\qquad\qquad\quad \text{, } otherwise \end{cases} \tag{4}$$

Here T is the threshold of difference.

It should be noted here that the difference image $\triangle I_{k+1}^{DC,Y}$ is expected to show high values corresponding to the image regions where the object has moved in the $(k+1)^{th}$ I-frame. However, the difference image may also show high values in the regions where either a different moving object (not marked by the user) is present or there is a change in the background due to variation in lighting

condition or presence of spurious movements like that of tree leaves and clouds. To make our system robust against such noise, we use an *adaptive threshold* (T) to compare the two frames, namely, I_{k+1} and I_0. We used T as 10% of the average of the sum of the DC component of all the blocks in the frame.

In case the difference between the DC values of two macro blocks having the same coordinates in frames under comparison have value greater than the threshold, then the macroblock of the target image is considered to be a part of the foreground and could be part of the tracked object. If the difference value is less than the threshold, we conclude that there has been no change in background for that particular block. After performing the subtraction of the luminance DC images, we generate an image where only the regions showing possible presence of foreground objects are retained. For all other regions, we set the values of all 64 DCT values (one DC and sixty three AC values) to zero. Thus, the process of background subtraction is equivalent to the application of a compressed domain mask on the entire image where the background subtraction mask ($Mask_{BS}$) is given by

$$Mask_{BS}(p,q) = \begin{cases} 1\ , if|I_{k+1}^{DC,Y}(p,q) - I_0^{DC,Y}(p,q)| \geq T \\ 0\ , otherwise \end{cases} \tag{5}$$

On application of the mask, at positions where the mask has a value of 1, the original DC values are retained. The complete algorithm for compressed domain background subtraction can now be written as shown in Fig. 1.

```
For all I Frames
Begin
   For j ← 1 to Y_DC_Image_Height
      For k ← 1 to Y_DC_Image_Width
      Begin
         Compute Mask_BS(j,k)
         I_{k+1}^{DC,Y}(j,k) ← I_{k+1}^{DC,Y}(j,k) • Mask_BS(j,k);
      end
end
```

Fig. 1. Algorithm for background subtraction

2.3 Candidate Objects Identification

We have explained in the last sub-section how background subtraction in the $(k+1)^{th}$ I-frame is done in the compressed domain. It is also mentioned that, since we perform threshold of the DC component of luminance values, we may get multiple regions in the $(k+1)^{th}$ I-frame which show high difference values. In the next step, we locate these candidate objects in the difference image. For this, the difference image obtained by applying the $Mask_{BS}$ is used to construct a binary image. We determine the bounding boxes of the candidate objects using a variant of DFS traversal for finding all the connected components (CC) from the binary image. Regions with bounding boxes less than a threshold size are filtered out.

It should be noted that the CC analysis is done in-memory since the output of the background subtraction mask is available in the data structures and no further reading/uncompress is required.

At the end of this step, we have a set of candidate objects from which the target object is selected using histogram matching as explained in the next subsection.

2.4 Target Object Selection

At this stage, our goal is to identify the tracked object from all the candidate objects in the foreground. We make use of the color and motion vector information available in the compressed domain for target object selection. We extract DCT coefficients of Cb and Cr components of all the macroblocks enclosed in the bounding boxes of the candidate objects. We also project the intended object (marked object) to the $(k+1)^{th}$ I-Frame by using motion vectors and then calculate the difference between the centroid of this projected object and the candidate object.

The objective here is to select that particular foreground object in the $(k+1)^{th}$ I-frame as the target object which has the maximum color-based similarity and minimum distance with the object marked by the user in the $(k)^{th}$ I-frame. To achieve this, we create '*Reference histogram*' of DCT coefficients of Cb and Cr components of the object marked by the user.

The histogram is generated from the DC value and first eight AC values of the Cb and Cr components since higher frequency AC values beyond the first eight do not contain much information and are often found to have very low/near zero values. From each of the 9 components (1 DC and 8 AC) of the two color planes, we create 128 bin histograms. Thus, we get 18 sets of reference histograms for the marked object.

For all the candidate objects identified in the previous step described in subsection 2.3, similar histograms are created called the '*Target Histogram*'. The reference histograms are then matched with all the target histograms. The difference between the Reference Histogram and Target Histogram is stored for a candidate object "i" as DiffHis(i).

The projection of the object marked by the user in the $(k)^{th}$ I-frame is taken on the $(k+1)^{th}$ I-frame. Its centroid is calculated by

$$x_m = (x_1 + x_2)/2, \quad y_m = (y_1 + y_2)/2 \tag{6}$$

Here (x_m,y_m) is centroid of the projected object. In the same way centroid of a candidate object 'i' is calculated and its difference with the centroid of the projected object is stored as DiffCen(i). Total distance is calculated for all the candidate objects by

$$Dis(i) = w_1 \bullet DiffHis(i) + w_2 \bullet DiffCen(i) \tag{7}$$

Here *Dis(i)* is the total difference of the candidate object "i",' w_1 is the weight given to the Histogram Difference and w_2 is the weight given to Centroid difference.

The candidate object with the minimum '*Dis()*' is selected as the one corresponding to the target object. The location of this object is considered to be the location where the object marked in the $(k)^{th}$ I-frame has actually moved in the $(k+1)^{th}$ I-frame. The same process is repeated for identifying the locations of the object in all the subsequent I-frames. The algorithm for target object selection is shown in Fig. 2. In the algorithm we consider *CreateHist()* to be a function that creates eighteen 128-bin histograms for each candidate object from Cb and Cr values. *HistDiff()* returns the difference between two histograms and *CenDiff()* returns the difference between the centroid of two objects.

```
Input : CbImg, CrImg, NumCandObj, CandObj[], ProjObj, RefHist[0..17][0..127]
Output : TargObj
Algorithm :
MinDiff ← INFINITY
for i ← 1 to NumCandObj
begin
    TargHist[i] ← CreateHist ( CbImg, CrImg, CandObj[i] );
    Diff His[i] ← HistDiff ( TargHist[i] , RefHist );
    DiffCen[i] ← CenDiff ( CandObj[i], ProjObj );
    Dis[i] ← w1 • DiffHis[i]  +  w2 • DiffCen[i]
    If ( Dis[i] ≤ MinDiffSum )
    begin
        TargObj ← CandObj[i];
        MinDiffSum ← Dis[i];
    end
end
return TargObj
```

Fig. 2. Algorithm for target object selection

A weighted sum of differences of the target and reference histograms with higher weights given to DC values and less to the AC values. The weights are chosen in such a way that the DC value, which most prominently conveys color information, is given the maximum weightage. It is followed by lower frequency AC values, which convey coarse texture and shape information and then higher frequency AC values [11], in decreasing order of importance.

2.5 Object Interpolation

We exploit the compressed domain features, namely, DCT coefficients of the luminance blocks and chrominance blocks for background subtraction and target object identification in subsequent I-frames after a user has marked it in a given I-frame. We interpolate the positions of the tracked object in the intermediate frames. Consider a GOP sequence IBBPBBPBBPBBI. Let the number of predicted frames be denoted by N. We divide the displacement vector between two I -frames by N. Let $\partial\bar{v}$ be the unit displacement vector per frame is given by

$$\partial\bar{v} = \frac{ObjectPos(I_{k+1}^{DC,Y} - I_k^{DC,Y})}{N} \tag{8}$$

The marking window in the intermediate frames, enclosing the object, is updated in accordance with their sequence number and temporal reference between the two I-frames. The interpolated rectangle coordinates is the predicted object's location. For intermediate P and B frames denoted by F_i, the object position is obtained using Eq. 9

$$ObjPos(F_i) = ObjPos(I_k) + \partial\bar{v} \bullet (TR(F_i) - TR(I_k)) \tag{9}$$

Here $TR(F_i)$ represents the temporal reference of the frame F_i which comes between I_k and I_{k+1}.

3 Experimental Results

We have performed large-scale experiments with our object tracking system. The video sequences for experiments were shot in outdoor as well as indoor environments to cater to different situations like change in background, object size, illumination, etc. We converted these video clippings into a standard MPEG video. Various kinds of moving objects were used for testing, including cars, slow and fast moving humans as well as multiple objects.

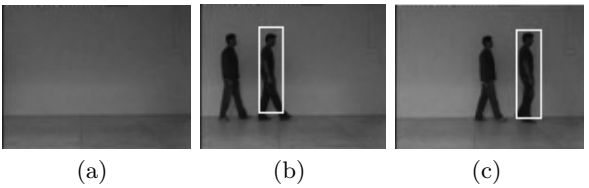
We show results of object tracking in a number of complex situations. In Fig. 3, there are two objects of same color, but only one is marked by the user, which was successfully tracked.

In Fig. 4, we show tracking results under indoor lighting conditions.

Quantitative result on accuracy is shown in Table 1. PA denotes the combined background subtraction and motion estimation based method as proposed in this

(a) (b) (c)

Fig. 3. Tracking of object in outdoor location (a) Background (b) Object marked by user (c) Tracked object



(a) (b) (c)

Fig. 4. Tracking of object in indoor location (a) Background (b) Object marked by user (c) Tracked object

Table 1. Comparison of accuracy

Algorithm	Environment			
	Outdoor			Indoor
	Sunny	Cloudy	Sunset	
MV	77.5%	73.7%	73.3%	82.5%
PA	92.5%	79.3%	77.2%	91.2%

paper and MV denotes a method based on motion vectors only. Performance evaluation was done "frame wise" in this study. This means, tracking output and ground truth were compared on a frame-by-frame basis. An object was considered "missed" in a frame by our system if less than 50% of it was tracked correctly. Results have been grouped together under various shooting conditions, like indoor, sunny, cloudy and sunset conditions and compared with a motion vector based tracking algorithm proposed in [11].

We also compared the time efficiency of our algorithm with the motion vector based prediction algorithm and derived that object tracking using background subtraction and motion estimation is much faster. A comparative performance in terms of speed is presented in Fig. 5 for a 1.8 GHz Pentium IV machine.

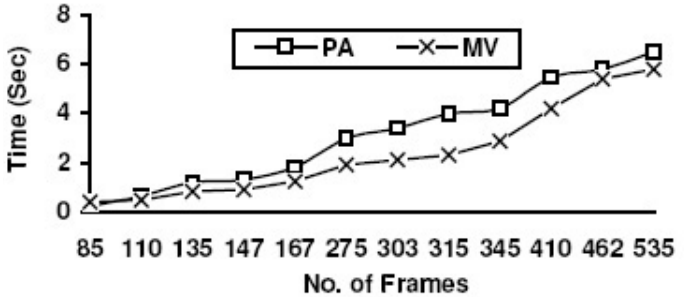


Fig. 5. Speed comparison with motion vector based tracking

As seen in the above figure, the combined approach is almost as efficient as a simple motion vector based approach, while its performance is much better as presented in Table 1. Another important observation is that, the proposed method can process frames at a rate of about 100 frames per second. Hence, it can be used in any real-time video tracking application.

4 Conclusion

We have proposed a novel tracking method that effectively tracks objects in a compressed video by combining background subtraction and motion estima-

tion. The system consists of four main components: background subtraction, candidate object identification, target object selection and object interpolation. Although we work strictly in the compressed domain, we still get high levels of accuracy with minimal processing time and computational cost. Several issues may further be addressed. This includes handling of full occlusions, fast camera motion, multiple object tracking and unsupervised tracking of objects.

Acknowledgment

The work done by Shamik Sural is supported by research grants from the Department of Science and Technology, India, under Grant No. SR/FTP/ETA-20/2003 and by a grant from IIT Kharagpur under ISIRD scheme No. IIT/SRIC/ISIRD /2002-2003. Work done by A. K. Majumdar is supported by a research grant from the Department of Science and Technology, India, under Grant No. SR/S3/EECE /024/2003-SERC-Engg.

References

1. G. L. Foresti and F. Roli: Real-time Recognition of Suspicious Events for Advanced Visual-based Surveillance. In Multimedia Video-Based Surveillance Systems: From User Requirements to Research Solutions, G. L. Foresti, C. S. Regazzoni, and P. Mahonen, Eds. Dordrecht, The Netherlands: Kluwer, pp. 84 - 93, 2000.
2. Y. Wang, R.E. Van Dyke and J F Doherty: Tracking Moving Objects in video Scene. Technical Report, Department of Electrical Engg, Pennsylvania State University, 2000.
3. V.V. Vinod and H. Murase: Video Shot Analysis using Efficient Multiple Object Tracking. Proceedings of IEEE International Conference on Multimedia Computing and Systems, pp. 501 - 508, 1997.
4. L. Wixson: Detecting Salient Motion by Accumulating Directionally-Consistent Flow. IEEE Transactions on PAMI, Vol. 22, No. 8, pp. 774 - 780, 2000.
5. I.O. Sebe: Object-Tracking using Multiple Constraints. Technical Report, Department of Electrical Engg, Stanford University, 2002.
6. O.Sukmarg and K.R. Rao: Fast Object Detection and Segmentation in MPEG Compressed Domain. Proceedings of TENCON, Vol. 3, pp. 364 - 368, 2000.
7. V. Mezaris et al: Real-Time Compressed-Domain Spatiotemporal Segmentation and Ontologies for Video Indexing and Retrieval. IEEE Transactions on CSVT, Vol. 14, No 5, pp. 606 - 620, 2004.
8. S. M. Park and J. Lee: Tracking using Mean Shift Algorithm. Proceedings of International Conference of Information Communications and Signal Processing, 2003, pp 748 - 752, 2003.
9. W.Y. Yoo and J.Lee: Analysis of Camera Operations in Compressed Domain based on Generalized Hough Transform. Proceedings of IEEE Pacific Rim Conference on Multimedia, Vol. 2195, pp. 1102 - 1107, 2001.
10. H. Kartik, D. Schonfeld, P. Raffy, F. Yassa: Object Tracking Using Block Matching. IEEE conference on Image Processing, Vol. 3, pp. 945 - 948, Jul 2003.
11. M.Kankanhalli, R. Achanta and J. Wang: A Sensor Fusion Based Object Tracker for Compressed Video. Proceedings of the Sixth International Workshop on Advanced Image Technology, 2003.