Enhanced Video Representation using Objects

Abhijit Mahindroo amahindroo@rediffmail.com
Biswajit Bose biswajit_bose@hotmail.com
Santanu Chaudhury santanuc@ee.iitd.ac.in
Gaurav Harit gharit@ee.iitd.ac.in

Department of Electrical Engineering
Indian Institute of Technology, Delhi
New Delhi - 110016, India

Abstract

Object-based video representations such as MPEG-4 have opened up new possibilities for video content access and manipulation. In this paper, we present a new object-based video representation paradigm, using appearance spaces. Our scheme enables fully automated extraction of semantic video objects for a class of sequences, and the development of a compressed, highly flexible representation of the video sequence based on extracted content. Our video representation supports content-based retrieval, as well as numerous enhanced features such as hyperlinking of videos, motion-icons for video browsing, automatic annotation, content-authoring facilities and semantic transcoding.

Keywords: Semantic video objects, motion segmentation, appearance spaces, *EigenTracking*, video representation, video summarisation, content-based functionalities.

1 Introduction

With the proliferation of multimedia data on the Internet, the major challenge in multimedia research is to provide efficient and effective access to multimedia data in response to user demand. While compression is needed just to transfer images and video over networks with limited bandwidth, multimedia data must also be organised and classified so that users can easily find what they are looking for. In this paper, we present a novel object based video representation scheme which enhances functional utility of the video content.

MPEG-4 standard has proposed an object-based approach to video description. The term *semantic video object* (SVO) refers to the set of 2D image regions (across multiple frames), which correspond to a real, physical object. So far, the stumbling block in object-based video coding has been the automation of the entire representation process. In general, it is not possible to define SVOs using any condition of homogeneity of image features (such as colour, texture or luminance). It is also clearly impossible for a video-database administrator to manually identify objects in every video sequence and describe/classify them. Thus, researchers have either focused on developing algorithms for extracting content from specific types of video (such as sports sequences[4] or sequences containing human faces) or resorted to semi-automatic methods requiring manual initialization and operator supervision [7, 9].

In this paper, we present a scheme for automated construction of an object based representation scheme for video sequences. We assume that objects of interest have distinct relative motion from the background. We use motion segmentation and appearance-based tracking [3, 5] for extracting these SVOs. Our method handles sequences containing multiple objects which might occlude each other.

In contrast to other structured spatio-temporal representations of video content based on motion cues [6], our novel representation scheme, based on object appearance spaces[8], is naturally amenable to meaningful classification of extracted SVOs and content-based video retrieval. It also enables a variety of content-based functionalities, such as iconic video browsing, video hyperlinking, content manipulation and semantic transcoding.

2 Content Extraction

Starting with a video shot, there are two steps to object extraction : segmentation and tracking. While segmentation is needed to identify SVOs whenever they appear in a video sequence, tracking is used to follow the identified objects across multiple video-frames so that an object representation can be developed. The key to successful extraction lies in correctly locating the segmented object region, which is used as the starting point for the tracker. While low-level image features (colour, texture, shape) can be used to identify local regions of interest, motion is one of the best cues for SVO segmentation, as it is more likely to characterise real objects (which are not homogeneous for any single image feature). Two commonly used motion-based methods are background subtraction (when the background is static, known *a priori* or can be estimated) and

clustering of optical flow[4]. Since we are processing arbitrary sequences in which the background is unknown and may constantly be occluded by foreground objects, we make use of an optical flow-based method. We assume that object motion between adjacent frames approximately satisfies an affine-motion model.

2.1 Motion Segmentation

Dense optical flow calculation is used to estimate the 2D apparent motion at each pixel in the video frame. We have used a robust computation technique [1] to reduce the sensitivity to violations of the brightness constancy and spatial smoothness assumptions underlying flow computation, and thus improve the accuracy of detection of motion boundaries. For further processing, we use only that colour plane (R,G, or B) which has maximum variation in calculated flow values.

The frame is divided into small non-overlapping square regions which we call *boxels*. The average optical flow magnitude is calculated in each *boxel*. We assume that flow magnitudes for boxels in both the moving and static regions of the frame are normally distributed. A threshold on flow magnitudes (computed iteratively) is set to the the flow value which corresponds to minimum probability between the maxima of these two normal distributions, so that best discrimination is obtained. This threshold is then used to classify *boxels* as foreground or background. We use the heuristic that the background always accounts for the majority of boxels. This ensures that our method works even if the camera moves to follows an object, so that object has very small apparent motion and the background has high motion. However, best segmentation results are obtained if the camera (and hence the background) is stationary. In this case, multiple foreground objects can be correctly identified as long as they do not overlap spatially.

Clusters of foreground boxels are identified as possible objects using connected-components detection. To eliminate spurious clusters (due to noise), we define a threshold on the minimum number of boxels required to form a valid object. We also remove any clusters near the frame boundaries, as these objects have partially entered/exited the frame. The remaining valid semantic video objects are then tracked by our appearance-based tracker.

2.2 Object Tracking

We have employed a modified version of the robust *EigenTracking* formulation [3] to track the newly identified foreground objects. This method makes use of the concept of an object's appearance space[8]. This is a low-dimensional vector space of object views defined by a small number of basis images (eigenvectors), such that a linear combination of these basis images can represent any view of that object (to a specified accuracy). *EigenTracking* involves estimation of affine motion parameters for the object region (which is initially defined by a rectangular bounding box), by minimisation of a matching-error. Instead of measuring the error in terms of pixel-intensity difference, the projection error (of the estimated object region in the next frame) on to the object's appearance space is used. For an object to be tracked, it's appearance space must therefore be known *a priori*.

In our modified tracker [5], pre-built appearance spaces for video objects are not available. Consequently, our tracker starts with the segmented object region as the first basis image, and dynamically updates (and thus builds) the appearance space of the tracked object by adding a new basis image whenever a novel view (that cannot be adequately represented by the existing basis images) is encountered. The tracker returns an appearance space for each tracked object, as well as the affine transformation of the each object's bounding box in every frame. Fig. 1 shows tracking results for two sequences.

After tracking all objects in a given sequence, the projection co-efficients for each object in every frame are calculated by projecting the object (within the bounding-box) on to each of the eigenvectors which characterize that object. These co-efficients, together with the affine transformation parameters, form the basis of our novel object-based video representation, as discussed in Section 3.

2.3 Coupling Segmentation and Tracking

As appearance spaces are incrementally built, incorrect tracking of video objects can corrupt their subsequent representation by admitting wrong views to their appearance space. Hence we need a *consistency check* to detect possible failure of the tracker and terminate the incorrect track. We check that the centre of mass (CoM) of the segmented region in every frame lies within a certain neighbourhood of the centre of the bounding box (CoBB) of the tracked object. If the tracker begins to drift away from the actual object (perhaps because of substantial background within the bounding box), the above check will soon flag an error. The tracker can then be re-initialized with the correct segmented region in the next frame.

The number of objects can vary arbitrarily as a sequence progresses because of certain critical events such as entry (exit) of objects into (from) the frame, appearance/disappearance of objects within the frame, merging of two objects into one, splitting of an object into two, or partial occlusion of an object by stationary scene elements. We use the segmentation results in every frame to match object regions across frames (based on spatial location and extent), to decide for which objects to start tracking, for which ones to continue tracking and which tracks to terminate. In order to match such corresponding regions in successive frames, and to identify possible merging and splitting of objects, we define two image primitives — the

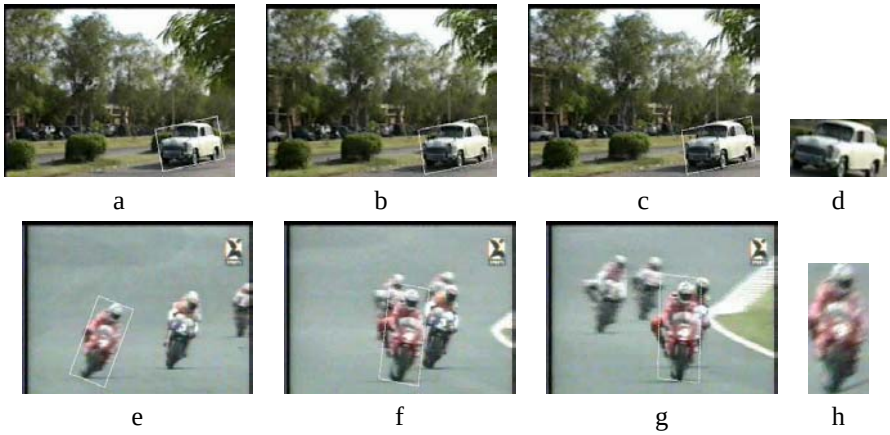


Figure 1: (a–c) Frames from the *ambassador* sequence, showing a car being tracked. (e–g) Tracking of a motor-bike in the *mobike* sequence. (d,h) Car and motor-bike objects reconstructed from their respective appearance spaces

motion-box and the *proximity-box* — centered about the CoM of each segmented object region.

For each object in the current frame, the motion-box is that region within which that object's CoM is assumed to lie in the next frame or previous frame, as the object propagates. Fig. 2(a,b) illustrates the CoM and motion-box (about the CoM) for a single object. The proximity-box of an object is slightly larger than its current spatial extent. It is used to check for cases of apparent merging and splitting of objects, such as two people crossing paths (apparent merging and splitting), a person passing behind a railing/pole (apparent splitting), or an actor jumping onto a moving car (merging). We need to ensure that merging (splitting) is not confused with propagation of one object and disappearance (appearance) of another, as this would lead to incorrect characterization of the object which is assumed to propagate smoothly. We have formulated the following rules to help identify these cases:

For a Motion Box in the current frame *if* # of CoMs in next frame is

0 *then* terminate track (Object Disappearance)

1 *then* continue track (Consistent Propagation)

For a Proximity Box in the current frame *if* # of CoMs in

next frame is >1 *then* case of Object Splitting

previous frame is >1 *then* case of Object Merging

The reader would note that, in all cases, object representation is strictly correct. A false alarm involving merging, splitting, appearance or disappearance will only split up a single object's lifetime into several epochs, with a separate, correct representation for the object in each epoch. This only leads to a slight reduction in representation efficiency.

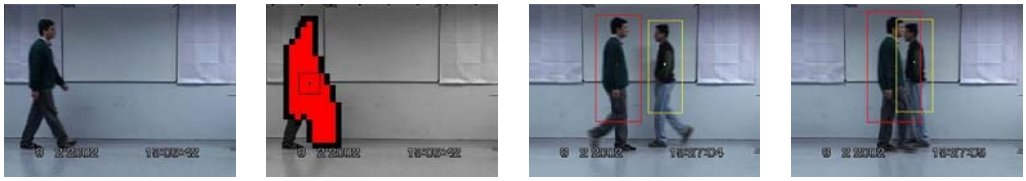
Frames from a sequence demonstrating three types of events (appearance, splitting and exit) are shown in Fig. 3. Here, a group of trucks is seen moving towards the right. As the trucks come closer to the camera, they are spatially separated from the group in the video frame (split) and are extracted (appearance) as independent objects.

3 Appearance-based Video Representation

The video-object extraction scheme that has been described thus far provides us with most of the information required to develop a complete representation for an arbitrary video sequence. Essentially this consists of the appearance spaces of objects, along with their projection co-efficients and affine parameters in each frame of the video in which they exist. We have chosen to represent both objects and frames explicitly, to allow for both content manipulation (object views/motion can be modified in any frame) and random starting of playback (after fast-forward/rewind, for example).

Object data contain eigenvectors characterising the appearance spaces of the respective objects. Frame data consists of the affine parameters and projection co-efficients for each constituent object. The background is most commonly the first frame of the sequence, JPEG encoded. If the background does not appear to be static in the original video (that is, the camera moves), then the background image needs to be regularly updated by adding the residual difference (between the background in the previous and present frames). The residuals themselves are compressed by applying a discrete cosine transform and storing only the low frequency co-efficients.

Our appearance-based representation implies the use of the Karhunen-Loève transform for optimal data compression. The total size of our video representation for a 27 MB (uncompressed) video sequence is 5.2 MB. This result is reasonable considering that video compression has not been our only concern. The object eigenvectors form the bulk of our data stream (4.5 MB), as they are not



a b c d

Figure 2: Motion- and proximity-boxes. Frame (b) shows CoM and motion-box marked against the segmentation result for the single object present in frame (a). Frames (c) and (d) of the *persons* sequence containing two objects (about to merge) show the respective CoMs and proximity-boxes

a b c d e f

Figure 3: Frames of the *trucks* sequence showing critical events: appearance(a), exit(b & f) and splitting(d,e)

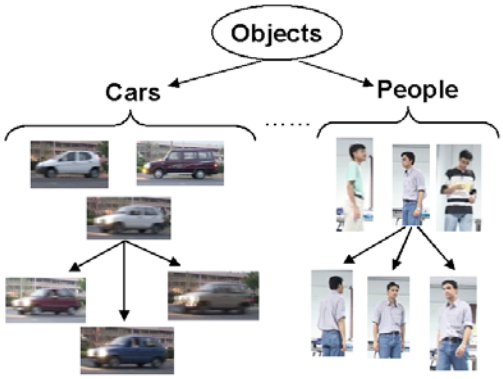


Figure 4: Illustration of Object Hierarchy. Appearance spaces of different cars or persons can be combined in an hierarchical manner to form generic categories.

compressed and are represented to floating-point accuracy. Compression of eigenvectors (by creating JPEG images of them) reduces the data size to 1.2 MB.

Object eigenspaces are particularly convenient for meaningful organisation of video data. By performing Gram-Schmidt orthogonalisation on the eigenvectors, appearance spaces of similar objects can be combined to form the appearance space of an object-class. These can be further combined, depending on class similarity, in the same manner, resulting in an object hierarchy (see Fig. 4). Hierarchy building is done in a semantically meaningful manner under supervision. By traversing this tree-structured hierarchy, a new object can be classified at each level by projecting its image onto each appearance space at that level and choosing the class which gives minimum reconstruction error. This organisation facilitates content-based video retrieval. Further details concerning object hierarchies and content-based querying can be found in [5].

3.1 *Micons*

While searching for videos online, users would like to browse through summaries of videos before downloading them. These summaries should give the user a good idea about the content of the video, by depicting critical events and scenes. Our object- and frame-based representation lends itself naturally to summary generation. In analogy with 'icons' used for representing images, our summary videos are called 'motion icons,' or just *micons*.

We define instances of appearance/disappearance , merging/splitting, entry/exit and occlusion to be critical events that provide maximum information about the video content. Our segmentation-tracking scheme (explained previously) can automatically flag key frames whenever these critical events occur. A micon thus simply consists of the compressed representation of these key frames. Fig. 5 shows some example *micons* to illustrate the concept.

3.2 XML Representation

We make use of XML (Extensible Mark-up Language) to represent the processed data after video analysis. Use of XML was originally proposed in MPEG-7[2] as standardised set of content description schemes to be attached to existing video data formats. However, by representing the video data itself as an XML document, we enable semantic processing of high-level video features (such as objects and their motion parameters) in addition to low-level features (such as pixel-values and frame numbers). A sample portion of an XML file is as follows:

Figure 5: *Micons* for (a) 100-frame *persons* sequence and (b) 140-frame *receding cars* sequence.

```
<?xml version="1.0" encoding="UTF-8"?>
<XML4VIDEO videoName="rata01" width="320" height="240"
startFrame="67" endFrame="88" nObjects="1"
bkgContentStreamID="rata01_001.jpg">

<XML4OBJ ID="1" name="Abhijit" width="48" height="126"
x-cood="197" y-cood="163" nvectors="6"
objectStreamID="seq1_obj01" residualID="rata_res067"/>

   <XML4FRAME Num="067" nObjects="1" isKeyFrame="yes">
      <XML4obj ID="1" HREF="class_human"
      AffineParams="70.21 0.97 -0.03 2.49 0.08 0.98"
      ProjCoeffs="14.60 14.60 15.12 10.65 8.65 9.37"/>
   </XML4FRAME>
                    .
                    .
                    .
</XML4VIDEO>
```

4 Enhanced Video Features

Our representation scheme provides for enhanced content-based functionalities for any video sequence that has been successfully processed by our system. These enhanced features are consequences of organising video objects in a meaningful hierarchy, based on the similarity of their appearance spaces.

4.1 Content Manipulation

Given a video sequence coded in our format, a user can replace the object data files or modify the affine transformation parameters to obtain an altered, authored video (with one object replaced by another, or moving in an arbitrary manner). Content authoring is extensively used in the entertainment industry for animations and special effects. It also allows end-users to produce their own videos by modifying existing ones. A typical example is shown in Fig. 6.

4.2 Object-based Video Hyperlinking

Textual hyperlinks are a convenient method of linking related text documents. We can now do the same for video sequences. Moreover, our object-based hyperlinking can automatically be done in a meaningful manner. To achieve this, we simply link together two objects whenever their appearances are similar enough for them to be classified in the same class of the object hierarchy. The actual links are implemented in the XML representation of the video sequence, and the user is shown icons corresponding to possible links in a sidebar as the video plays. A *micon* may be associated with each link. The hierarchical organisation of objects can also be used for automatically annotating video. By storing a small textual message, description or interesting fact at each level of the object hierarchy, video objects can be automatically tagged with relevant information. Our proposed interface for video hyperlinking and annotation display is shown in Fig. 7.

Figure 7: Proposed interface for video-object hyperlinking and virtual annotation. When a video object appears, the corresponding annotation appears below it, and possible hyperlinks are highlighted in the sidebar on the right.

4.3 Semantic Transcoding

We propose semantic transcoding as a means of bandwidth conservation, deriving from our ability to classify video objects. First, we arrange for a small, pre-built object hierarchy (consisting of representative objects) to be present at the user-end as part of the video-search software. Then, while processing a video sequence at the server end, constituent video objects are classified by traversing the complete object hierarchy. Now, instead of transmitting the entire appearance space for each object, the server only sends a class identifier to the client end. The client software matches this identifier against its pre-built hierarchy and inserts a representative object from the object-class to which the identifier corresponds. For example, all cars in a

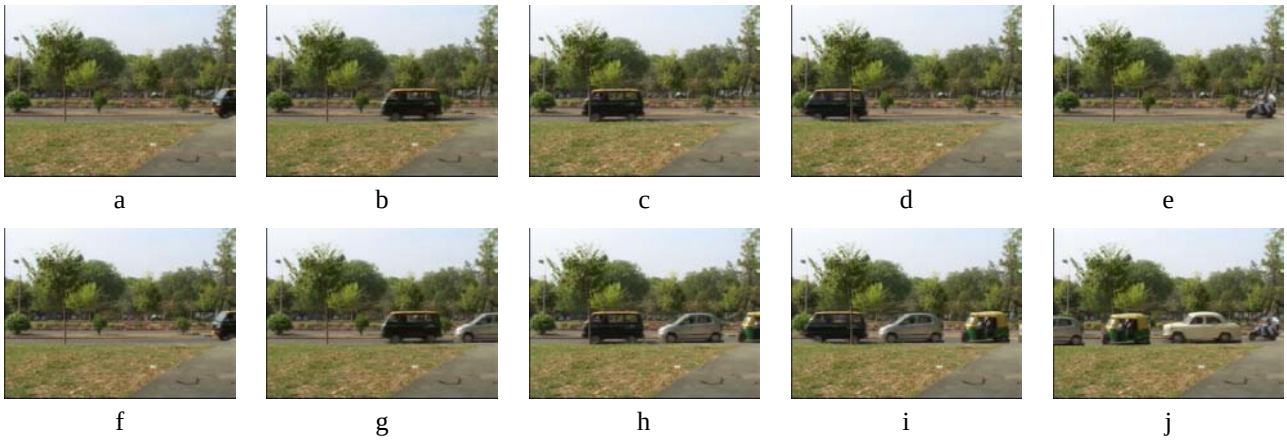


Figure 6: (a to e) Five selected frames of the *vehicle* sequence in which a Maruti van is followed by a scooter. (f to j) Content-authored video frames, after inserting 3 'virtual' cars between the original two vehicles. By manipulating the affine parameters and introducing the 'virtual' cars, the authored video seems to show a procession of cars.

video may be replaced by a prototype Maruti/Ambassador car, which is present in the pre-built hierarchy. Thus, the user still gets a general idea of the video content, though with a slight loss in detail.

Semantic transcoding, along with miconic representation, helps produce tremendous compression. For the uncompressed 80-frame, 16MB *vehicle* video of Fig. 6 our basic XML representation (of the complete video) is 4.2 MB in size. The 6-frame *micon* has a 0.9 MB XML representation. Finally, the size of the miconic representation with semantic transcoding of the *car* object-class is 300 KB — essentially the size of just one video-frame.

5 Conclusions

The key contributions of this paper are two-fold: the proposal of a completely automated scheme for object extraction for a certain class of videos, and the proposal of a new object-based representation allowing for enhanced content-based functionalities. The detailed description of object tracking and hierarchical category based object representation scheme using appearance spaces has been provided in [5]. Our representation scheme has allowed us to implement features such as video-object hyperlinking and semantic compression, which have not been done previously. Thus, our environment allows for seamless interaction of the user with visual media.

Ongoing work involves extension of the extraction scheme to handle scenes shot with a moving camera.

References

[1] Black, M. J. and Anandan, P., "The Robust Estimation of Multiple Motions: Parametric and piecewise-smooth flow fields", CVIU, 63(1):75–104, 1996.

[2] Chang, S.-F., Sikora, T. and Puri, A., "Overview of the MPEG-7 Standard", IEEE Trans. Circ. Sys. Video Tech., 11(6):688–695, 2001.

[3] Black, M. J. and Jepson, A. D., "EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation", Tech. Report: RBCV-TR-96-50, Dept. of Comp. Sc., University of Toronto, October 1996.

[4] Deng, Y. and Manjunath, B. S., "NeTra-V: Toward an Object-based Video Representation", IEEE Trans. Circ. Sys. Video Tech., 8(5):616–626, 1998.

[5] Garg, G., Sharma, P. K., Choudhury, R. and Chaudhury, S., "An Appearance based approach for Video Object Extraction and Representation", Proc. ICPR, August 2002.

[6] M. Gelgon and P. Bouthemy, "Determining a structured spatio-temporal representation of video content for efficient visualization and indexing," ECCV '98 Fribourg, Allemagne, Vol 1:595-609.

[7] Gu, C. and Lee, M. C., "Semi-automatic segmentation and tracking of semantic video objects," IEEE Trans. Circ. Sys. Video Tech., 8(5):572–584, 1998.

[8] Murase, H. and Nayar, S. K., "Visual Learning and Recognition of 3-D Objects from Appearance", IJCV, 14:5–24, 1995.

[9] Zhong, D. and Chang, S.-F., "An Integrated Approach for Content-Based Video Object Segmentation and Retrieval", IEEE Trans. Circ. Sys. Video Tech., 9(8):1259–1268, 1999.