# Real-Time Video Object Segmentation for MPEG Encoded Video Sequences

Fatih Porikli*

Mitsubishi Electric Research Laboratories, Cambridge, USA

## ABSTRACT

We propose a real-time object segmentation method for MPEG[†] encoded video. Computational superiority is the main advantage of compressed domain processing. We exploit the macro-block structure of the encoded video to decrease the spatial resolution of the processed data, which exponentially reduces the computational load. Further reduction is achieved by temporal grouping of the intra-coded and estimated frames into a single feature layer. In addition to computational advantage, compressed-domain video possesses important features attractive for object analysis. Texture characteristics are provided by the DCT coefficients. Motion information is readily available without incurring cost of estimating a motion field. To achieve segmentation, the DCT coefficients for I-frames and block motion vectors for P-frames are combined and a frequency-temporal data structure is constructed. Starting from the blocks where the $ac$-coefficient energy and local inter-block $dc$-coefficient variance is small, the homogeneous volumes are enlarged by evaluating the distance of candidate vectors to the volume characteristics. Affine motion models are fit to volumes. Finally, a hierarchical clustering stage iteratively merges the most similar parts to generate an object partition tree as an output.

**Keywords:** MPEG, compressed domain segmentation, volume growing

## 1. INTRODUCTION

Video object segmentation is one of the most challenging task in video processing. It is important for video compression standards as well as recognition, event analysis, understanding, and video manipulation purposes. It enables video based search, indexing, and content retrieval.

Existing algorithms on the object segmentation show that segmentation in the uncompressed domain is computationally demanding. Besides, if a video source is provided in the compressed form, these operations can not be performed until that representation has been decompressed first. Since most video data is already compressed, it is computationally less expensive to directly process in the compressed domain rather than decomposing the video into the spatial domain. The block structure of the compressed domain data also drastically condenses the amount of data to be processed. In addition to reduced computational complexity, there are several other advantages of performing analysis in the compressed domain. Compressed video contains information about spatial energy distribution within the image blocks. Frequency domain representations relay information on image characteristics such as texture and gradient. Furthermore, motion information is readily available without incurring cost of estimation of motion field. Compressed domain analysis may also serve as an initial segmentation stage that steers the following uncompressed domain segmentation by providing fundamental information such as motion parameters and color properties to decrease the computational load of the further processing.

Compressed domain analysis have limitations as well. The Discrete Cosine Transform (DCT) removes the spatial correlation among the pixels within a block, thus the precision of the segmentation degrades by the block dimension. Since the goal of motion compensation is to provide a good prediction but not to find the correct optical flow, the the motion vectors (MV) are often contaminated with mismatching and quantization errors. On top of that, the motion fields in MPEG streams are quite prone to quantization errors.

In contrast to the immense amount of work performed over uncompressed video, only a few researchers have proposed the object segmentation algorithms in the compressed domain. Some algorithms are even restricted in DCT coefficients. For instance, Wang[8] proposed an algorithm to automatically detect faces where he use skin-tone statistics, shape constraints, and energy distribution of the luminance DCT coefficients to locate the face position. Similarly, De Queiroz[3]

---

*fatih@merl.com, phone: 1.617.621.7586
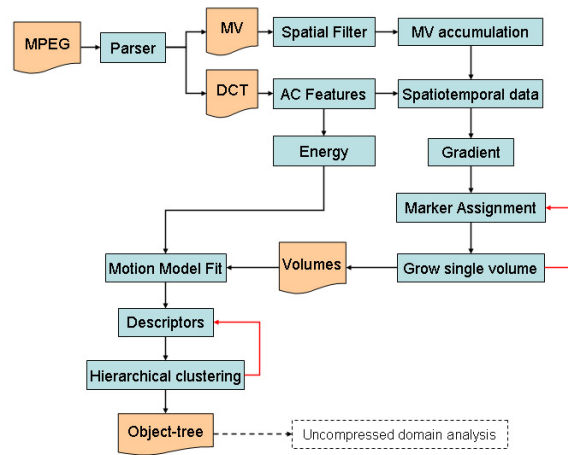[†]Throughout this paper, we refer to the MPEG-1/2 compression standards.

**Figure 1.** Flow diagram of the segmentation algorithm.

segmented JPEG images into specific regions such as those containing halftones, text, and continuous-tone using the encoding-cost-map based on DCT coefficients. In a related work, Sukmarg and Rao[7] propose a region segmentation and clustering based algorithm to detect objects in MPEG compressed video. Their segmentation algorithm consists of four main stages; initial segmentation using sequential leader and adaptive k-means clustering, region merging based on spatiotemporal similarities, foreground-background classification, and object detail extraction. However, this algorithm does not have a mechanism to handle the motion vectors of multiple P-frames. It requires several preset thresholds and the value of the sequential clustering threshold is crucial to determine the number of objects. Besides, k-means clustering needs appropriate weights for the block coordinates, DCT coefficients, and motion information. A confidence measure based moving object extraction system was proposed by Wang[9] *et. al*. They suggested several confidence measures to improve motion layer separation. Their algorithm detects objects after a global motion compensation. Ji and Park[5] segmented dynamic regions based on the DCT coefficient similarity and true/false motion block classification. However, this method requires tracking of individual regions. Babu and Ramakrishnan,[1] on the other hand, used only aggregated motion vectors.

To address the shortcomings of the above approaches, we develop a fast, automatic, compressed domain segmentation algorithm that blends motion and frequency information. A flow diagram is shown in Fig.1. After parsing an MPEG-1/2 video into DCT coefficients and motion vectors, we construct a frequency-temporal data structure for the multiple Group of Pictures (GOP)'s between two scene-cuts. Each GOP is represented by a layer of vectors that correspond to blocks in a frame. Each vector consists of selected DCT coefficients and accumulated forward-pointing MV's. Then, we grow volumes within this 3D data structure by starting from the seeds. The seeds are chosen among the blocks that the local texture and gradient is minimum. The volume growing gives the connected parts of video that have consistent DCT coefficient and motion properties. For each volume, we determine volume descriptors, including affine motion parameters, using trajectories and MV's. In the final stage, we merge similar volumes pair-wise using their descriptors to obtain an object-partition tree. Our method is robust towards threshold perturbations and computationally simple at the same time.

In the next section, we explain the MPEG parser. In section 3, we introduce the frequency-temporal data structure. In section 4, we give details of the volume growing. In section 5 and 6 we present the motion parameter estimation and the hierarchical volume clustering algorithms.

## 2. MPEG PARSER

### 2.1. What is MPEG?

The basic idea behind the MPEG-1/2 video compression is to remove spatial redundancy within a video frame and temporal redundancy between video frames. DCT-based compression is used to reduce spatial redundancy. Motion-compensation is used to exploit temporal redundancy. The MPEG compression scheme converts the video bitstream in terms of I (intra-compressed), P (forward predicted), and B-frame (bi-directional predicted). An I-frame is encoded as a single image, with

no reference to any past or future frames. The I-frame stores DCT information of the original frame. The P and B frames store the motion information and residues after motion compensation. Though I-frame provides no motion information, still color and texture information can be grasped and propagated to the P, I frames by inverse motion compensation. A P-frame is encoded relative to the past reference frame. A reference frame is a P- or I-frame. The past reference frame is the closest preceding reference frame.

Frames are divided into 16x16 pixel macroblocks. Each macroblock consists of four $8 \times 8$ luminance blocks and two 8x8 chrominance blocks. Macroblocks are the units for motion-compensated compression. Each macroblock in a P-frame can be encoded either as an I-macroblock or as a P-macroblock. An I-macroblock is encoded just like a macroblock in an I-frame. A P-macroblock is encoded as a 16x16 area of the past reference frame, plus an error term. To specify the 16x16 area of the reference frame, a motion vector is included.

The block is first transformed from the spatial domain into a frequency domain using the DCT, which separates the signal into independent frequency bands. The DCT coefficients have a relationship with spatial frequencies and, given that the different components have different subjective importance, DCT gives an important tool to remove also the subjective redundancy. Thus, most frequency information is in the upper left corner of the resulting 8x8 block. In DCT, coefficient in location (0,0) is called $dc$ coefficient and the other values we call them $ac$ coefficients. After DCT transform, the data is quantized. Quantization can be thought of as ignoring lower-order bits. The resulting data is then run-length encoded in a zigzag ordering to optimize compression.

The goal of motion compensation is to provide a good prediction for the macroblock. Actually, in the macroblocks where prediction is applied, the DCT is performed to the prediction errors instead of to the image samples and more the prediction errors are low and more the entropy coding is effective. Therefore, with good predictions it's possible to have low bit rate and good quality. Motion-compensated prediction assumes that the current picture can be locally modeled as a translation of the pictures of some previous time. The MPEG syntax specifies how to represent the motion information for each macroblock of P and B frames. It does not, however, specify how such vectors are to be computed. Due to the block-based motion representation, many implementations use block-matching techniques, where the motion vector is obtained by minimizing a cost function measuring the mismatch between the reference and the current block. Thus, the MPEG motion vectors does not necessarily correspond to the true motion but the best matching of macroblocks.

The sequence of different frame types is called the Group of Pictures (GOP) structure. There are many possible structures but a common one is 15 frames long, and has the sequence IBBPBBPBBPBBPBBPBB. A similar 12 frame sequence is also common. The ratio of I, P and B pictures in the GOP structure is determined by the nature of the video stream and the bandwidth constraints on the output stream.

## 2.2. Parsing

To obtain the DCT coefficients and motion vectors, the parser retrace the encoding stages. First, the binary MPEG video stream is chopped into bytes and variable length decoding is applied. Then, the scan line is reconstructed to relocate the DCT coefficients into blocks. The inverse $ac/dc$ prediction is done to decode the DCPM of the $dc$ coefficients. At this point, all the DCT coefficients are in the quantized format. Thus, an inverse quantization is applied to find the original DCT coefficients. The motion vectors are obtained after variable length decoding.

Although the above process has several tasks, it is computationally simpler part of a full MPEG decoder, in which most of the computation is involved in the inverse DCT and motion compensation stages. On average, the parsing requires only 3 10% of the decoding time[2]-[6] for a GOP. At our P4 3Ghz platform, this corresponds to $0.2 \sim 0.7ms$.

## 3. FREQUENCY-TEMPORAL DATA STRUCTURE

After parsing the MPEG sequence, the DCT coefficients and MV's of GOP's are assembled into a frequency-temporal data structure. In other words, several GOP's within a vedeo shot between two scene-cuts are reindexed such that they form a 3D (m,n, and t) data. Since it contains both DCT coefficients and motion vectors, it captures the frequecy-temporal characteristics of the corresponding video shot. Each element of this data structure corresponds to the attributes of a 8x8 block. A feature vector denoted by $f_{t,m,n}$ is assigned to the elements of the frequency-temporal data. The components of the feature vector include the DCT coefficients ($dc$ coefficients of the Y-, U-, and V-channels, selected $ac$ coefficients of the Y-channel), an energy term $E$, and the accompanying forward-predicted motion vector:

$$f_{t,m,n} : [dc_y \ dc_u \ dc_v \ ac_1 \ ac_2 \ ac_3 \ ..., \ E \ mv_x \ mv_y]^T. \tag{1}$$

**Figure 2.** (a) Color image, (b) $dc_y$ coefficients, (c) $dc_u$ coefficients, (d) $dc_v$ coefficients.

where $m, n$ is the indices of the corresponding vector in the data structure, and $t$ is the GOP number. We will call vectors belong to the same GOP as a layer of the data. Note that, for a 352x288 spatial resolution, 15-frames GOP structure color video segment (352x288x15x3 intensity pixels), the size of the frequency-temporal data becomes $44x36x1x9$. This is equal to a drastic reduction of 320:1 in data size.

The $dc$ and $ac$ components only exists for the I-frame of the GOP. The $dc$ coefficients represent the average color of the block, thus they can be considered as a subsampled I-frame by a factor of 8 (Fig.2). The $ac$ coefficients convey information about the spatial energy distribution within the corresponding block. Since the DCT transform of an 8x8 image block is defined as

$$dct(u,v) = \frac{1}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} I_{xy} \cos \frac{\pi u(2x+1)}{16} \cos \frac{\pi v(2y+1)}{16} \tag{2}$$

where $u$ and $v$ are the horizontal and vertical frequencies ($u, v = 0, .., 7$), most of the higher indexed $ac$ coefficients become equal to zero after the quantization stage for an I-frame block where the texture is smooth. This property of the DCT is also known as energy compactness. Thus, we only include the certain number of lower indexed DCT coefficients into the feature vector. This simplifies the computations without sacrificing the accuracy. In our simulations, we used first 3 $ac$ coefficients in the zigzag order, however further investigation can be done to determine the optimum number of the $ac$ coefficients. The energy term $E$ is the sum of $ac$ coefficients and it indicates the amount of spatial texture

$$E = \sum_{u=1}^{7} \sum_{v=1}^{7} dct(u,v). \tag{3}$$

There is a strong correlation between the energy term and the confidency of the motion estimation for a block.

Since the original MPEG motion vectors are prone to errors due to the block-matching and quantization, we apply spatial filtering to prune the extremities of motion vectors. We convolute motion vectors with a balanced (summation of the filter weights is equal to one) 3x3 Gaussian template. We multiply the motion vector magnitudes with the template weights, and aggregate the motion vectors within the 3x3 window. As motion field changes abruptly, this filtering smears the object boundaries.

One problem of integrating the motion information into the feature vector is that the motion vectors of P-frames are back-predicted. In other words, for an I-P frame pair, only the blocks in the P-frame have motion vectors pointing their most similar placements in the I-frame. The motion vector for an I-frame block does not exists. Thus, we convert the motion vectors of the P-frame to the I-frame motion vectors as illustrated in Fig.3-a. Therefore, we can find a motion vector for an I-frame block that points the matching region in the following P-frame as opposed to original motion vectors after the parsing. We project each P-frame block by its motion vector to its I-frame, then compute the overlapping areas between the I-frame blocks and the projected block. We update the I-frame motion vectors of the overlapped I-frame blocks according to the ratio of the overlapping area to the total covered area of this block after all the vectors of P-frame are projected. For an I-frame block that is entirely covered by the projected P-frame blocks, the motion vector prediction is more accurate than another frame that is partially covered. The forward-prediction is only applied to the immediate neighboring P-frame, since the accuracy of the prediction exponentially degrades as illustrated in the Fig. 3-b. Note that, the accuracy forward-predicted motion vector is limited to the accuracy of the original vectors, which may be inaccurate as
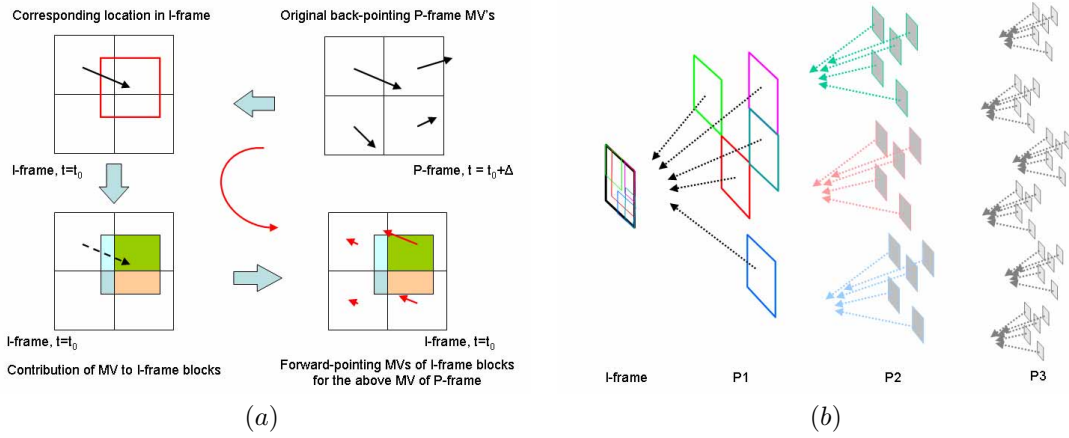
Figure 3. (a) A P-frame motion vector contributes at most four I-frame blocks. (b) Forward motion prediction exponentially branches out for more than one P-frames.

well. Furthermore, around the object boundaries more disturbance is introduced. One way to obtain the forward-predicted motion vectors is to decode the I and P frames and compute optical flow. However, this is computationally expensive.

## 4. VOLUME GROWING

Volumes are grown within the frequency-temporal data starting from the seeds. By definition, a seed should characterize its local neighborhood as relevant as possible because it is assumed to represent the initial volume accurately. Such vectors having relatively low local gradient are good candidates to represent their neighborhood. Thus, we compute a gradient term defined as

$$\nabla f_{t,m,n} = |f_{t,m-1,n}(dc_y) - f_{t,m+1,n}(dc_y)| + |f_{t,m,n-1}(dc_y) - f_{t,m,n+1}(dc_y)| \tag{4}$$

and select the vector that has the minimum gradient as a seed $f^*$. We used the Y-channel $dc$ component since it has the highest resolution. A volume $V_i$ is initiated, and a volume's representative is set as $f^i = f^*$. This vector does not correspond to a physical location but stores the current attributes of the volume. Then, the vectors are evaluated in a 8-neighborhood in all 3 directions. This is the main difference between region growing and volume growing.[4] Here, we assume that regions belongs to the same object are overlapping between the consecutive GOP's, which holds for the GOP's that heve small number of frames. In case a GOP consists hundreds of P-frames, the time distance between the I-frames of two consecutive GOP's will increase accordingly, which may invalidate the overlapping assumption. However, most MPEG sequences consists of GOP's that have 6-15 frames.

We compare the vectors in 8-neighborhood by computing the vector distance. Our vector contains different terms thus we adapted the following distance metric

$$\delta(f^i, f) = \omega_{dc}\delta_{dc}(f^i, f) + \omega_{ac}\delta_{ac}(f^i, f) + \omega_{mv}\delta_{mv}(f^i, f) \tag{5}$$

where

$$\delta_{dc}(f^i, f) = |f^i(dc_y) - f(dc_y)| + |f^i(dc_u) - f(dc_u)| + |f^i(dc_v) - f(dc_v)|, \tag{6}$$

$$\delta_{ac}(f^i, f) = \sum_{k=1}^{3} |f^i(ac, k) - f(ac, k)|, \tag{7}$$

$$\delta_{mv}(f^i, f) = \sqrt{(f^i(mv_x) - f(mv_y))^2 + (f^i(mv_x) - f(mv_y))^2}. \tag{8}$$

where $\omega_{dc}, \omega_{ac}, \omega_{mv}$, are the weights of the corresponding distances. These weights determine how much each attribute contributes to the distance metric. We observed that the higher values of the $\omega_{dc}$ provides more accurate segmentation in
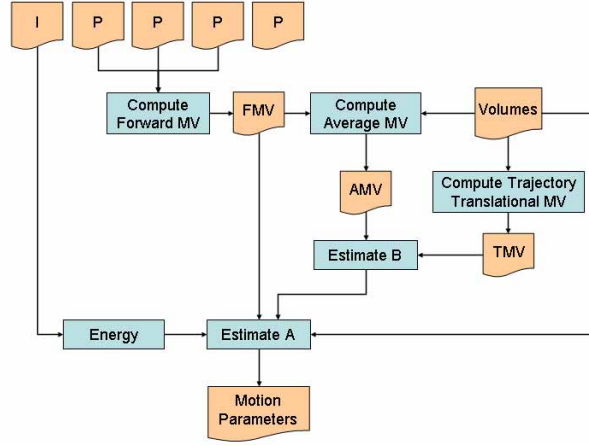
**Figure 4.** Affine motion parameters are fitted using translational motion information.

case of the motion is insignificant (for instance for *Akiyo* and other head-and-shoulder sequences). In case there is fast moving objects with multiple colored/textured regions, $\omega_{mv}$ becomes more dominant.

If the color distance is less than a threshold $\delta(f^i, f) < \epsilon$, the vector $f$ is included in the volume $V_i$, it is assigned as an active boundary, and the volume representative is updated by the averaged means of the corresponding components. After a volume is grown, all the vectors of the volume is removed from the available data set. The next minimum gradient vector in the remaining set is chosen, another volume is grown, and the selection process is iterated until no more vector remains. After volume growing, some of the volumes may be negligible in size. Such volumes are removed and the remaining volumes are inflated to fill up the empty space.

The seed selection is the computationally intensive task of the volume growing since it searches for a minimum gradient in the data. One way to improve the speed is to find the local minimum in the current layer and then grow a region. The next seed however is searched in another layer by covering all the layer sequentially. The fast seed selection and volume growing take $0.8 \sim 1.5ms$ for a GOP on average.

## 5. MOTION PARAMETERS

After volume growing, we have the parts of the video that is consistent in terms of their DCT coefficients and translational motion distributions. The next task is to fit a motion model to each volume. We accomplish this by first estimating the affine motion parameters of the regions of a volume in the corresponding layers then averaging the set of individual parameters over all of the layers. Thus, we solve the region of support problem, which inherently comes with model fitting schemes, by using the segmented regions in the layers. A set of affine motion parameters $A = [a_1, ..., a_4, b_1, b_2]$ models the layer-wise motion

$$8 \begin{bmatrix} m \\ n \end{bmatrix} + \begin{bmatrix} mv_x \\ mv_y \end{bmatrix} = 8 \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} m \\ n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = 8Ax + b \tag{9}$$

where $[m, n]^T$ is the block indices. The constant multiplier 8 converts the block indices to spatial coordinates in which the original motion vectors are measured.

We have two translational motion information; one is the average of the motion vectors within the region $\overline{mv}$ and the second is the trajectory displacement $tmv$. Motion trajectory of a volume is defined as its frame-wise representative coordinates. The representative coordinates can be chosen as the center-of-mass. Trajectory is calculated by averaging the coordinates of the vectors belong to the volume in a layer. Then, we assign the translational motion as the median of these two vectors $b = [b_1, b_2]^T = 0.5(\overline{mv} + tmv)$. For a region that consists of $K$ blocks we accumulate the motion vectors
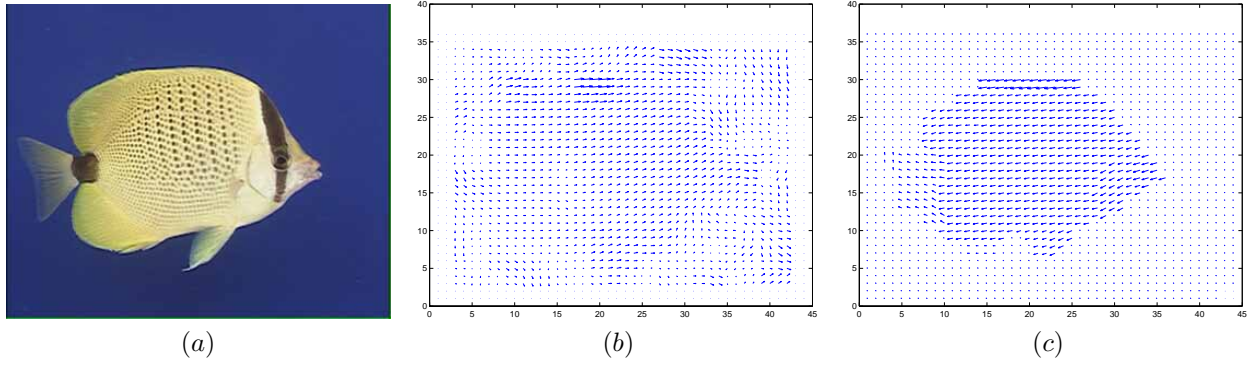
**Figure 5.** (a) A sample frame from *Bream*, (b) original MPEG motion vectors interpolated for 8x8 blocks, (c) estimated translational motion vectors.

$[mv_x^k, mv_y^k]^T$ and its originating coordinates $[m_i, n_i]^T$ as

$$8 \begin{bmatrix} m^1 & \cdots & m^K \\ n^1 & \cdots & n^K \end{bmatrix} + \begin{bmatrix} mv_x^1 & \cdots & mv_x^K \\ mv_y^1 & \cdots & mv_y^K \end{bmatrix} - \begin{bmatrix} b_1 & \cdots & b_1 \\ b_2 & \cdots & b_2 \end{bmatrix} = 8\, A \begin{bmatrix} m^1 & \cdots & m^K \\ n^1 & \cdots & n^K \end{bmatrix} \tag{10}$$

$$Y_{(2 \times K)} = A_{(2 \times 2)} X_{(2 \times K)} \tag{11}$$

where only unknown is the matrix $A$. Since $X$ is an $2 \times K$ matrix, $A = Y/X$ is the solution in the least squares sense to the overdetermined system of equations $Y = AX$. The effective rank $R$ is determined from the QR decomposition with pivoting. A solution $A$ is computed which has at most $R$ nonzero components per column. The above parameters are computed for each volume at every layer. The original and estimated motion vectors are given in Fig. 5. for a This process takes $8 \sim 10 ms$ for a GOP.

## 6. HIERARCHICAL CLUSTERING

The segmented volumes are clustered into objects using their motion parameters. Different approaches to clustering data can be categorized as hierarchical and partitional approaches. Hierarchical methods produce a nested series of partitions while a partitional clustering algorithm obtains a single partition of the data. Merging the volumes in a fine-to-coarse manner is an example to hierarchical approaches.

The pair having the most similar parameters are merged, and the motion parameters of the volumes are updated accordingly as demonstrated in Fig. 6. The similarity criteria is defined as

$$s(V_i, V_j) = 1 - \frac{1}{S} \sum_t \left[ c_R \sum_{k=1}^{4} |a_{k,i,t} - a_{k,j,t}| + c_T \sum_{k=1}^{2} |b_{k,i,t} - b_{k,j,t}| \right] \tag{12}$$

where the constants are set as $c_T \gg c_R$ to take into account of the fact that a small change in the rotation/scaling parameters can lead to much larger difference in the modeled motion field than the translation parameters. The scalar $S$ is equal to the maximum possible dissimilarity.

Clustering is performed until there are only two volumes remain. At a level of the clustering algorithm, we can analyze whether the chosen volume pair is a good choice. This can be done by observing the behaviour of the parameter similarity of the selected merge. If this score gets small or drops suddenly, the merge is likely to be invalid.

The segmentation algorithm supplies volumes, their attributes, and information about how these volumes can be merged. Since human is the ultimate decision maker in analyzing the results of video segmentation, it is necessary to provide the segmentation results in an appropriate format to user or for further analysis.
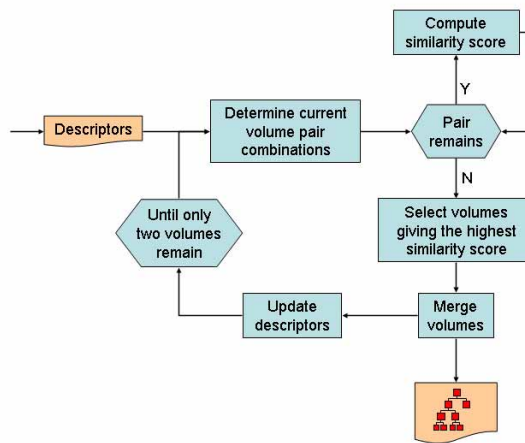
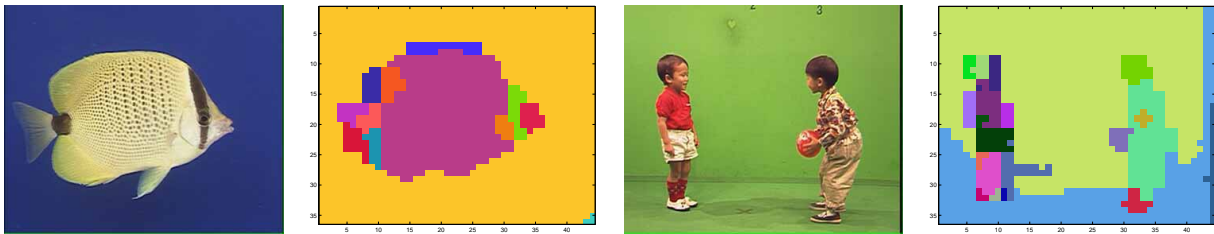**Figure 6.** At each iteration, two most similar volumes are merged.



**Figure 7.** Segmentation results from object-tree for *Bream* and *Children* test sequences. The regions are random-color coded.



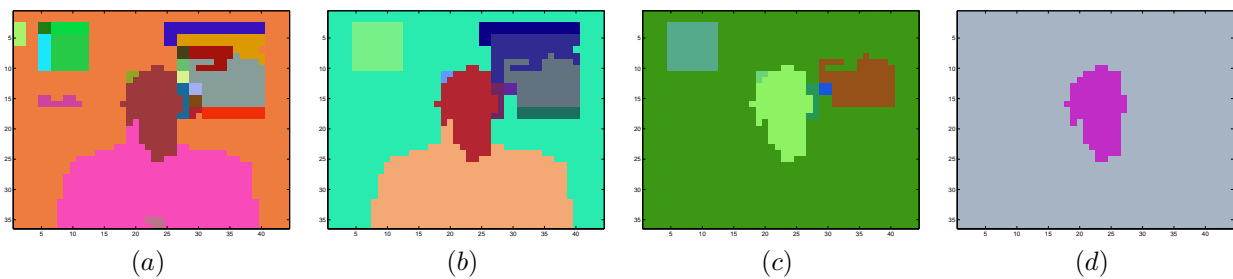$(a)$ $\qquad\qquad (b) \qquad\qquad (c) \qquad\qquad (d)$

**Figure 8.** Different levels of object-partition tree for *Akiyo*, (a) 21 objects, (b) 11 objects, (c) 7 objects, (d) 2 objects levels. The regions are random-color coded.

# 7. DISCUSSION

We present a real-time object segmentation method for MPEG encoded video. We tested the proposed algorithm for 352x288, 12-frame GOP structured, color MPEG-1 sequences. We used 8 GOP's for each case to construct the frequency-temporal data structure. Thus, at each segmentation cycle, we segmented 96 frames together. We also used only the first 3 $ac$ coefficients when we assemble the feature vector. We set the volume growing thresholds once and used the same thresholds for all test sequences. The distance threshold assigned to a higher value $(1.3\lambda)$ for inter-layer vector difference than the intra-layer vector difference $(\lambda)$ to exploit the inter-layer growing for the sequences that have fast motion. We only used the first P-frame after the I-frame to obtain the forward-predicted motion vectors. We removed the volumes smaller than 10 vectors.

Figure 7 shows initial segmentation results from two video sequences. As visible, the objects are accurately detected at the coarse block resolution. Sample hierarchical clustering results is given in Fig. 8 for different object levels. As visible in these results, the motion parameter based similarity measure can detect the small motion variances. Although a fast moving single small object may invalidate the overlapping regions assumption and appear as separate objects in different layers, we observed that, for the moderate motion sequences, the trajectories are continuous and segmented region boundaries are accurate. We also concluded that the segmentation process is not sensitive to the minor threshold perturbations which gives additional flexibility. The proposed algorithm is faster than real time video playing speed. The total segmentation time including the MPEG parsing varies in the range of $10 \sim 20ms$ for a GOP on a P4 3Ghz platform depending on the number of initial objects after the volume growing. Most computations are involved in motion parameter fitting stage. Favorably, the speed is not influenced by the complexity of the motion. The proposed algorithm reaches $0.9 \sim 2ms$ processing speeds per frame.

As future work, we plan to develop automatic threshold assignment methods and use the compressed domain processing results as a precursor to improve the uncompressed domain segmentation.

# REFERENCES

1. R. Venkatesh Babu and K. Ramakrishnan, "Compressed Domain Motion Segmentation for Video Object Extraction", IEEE International Conference on Acoustics, Speech, and Signal Processing, Florida, 2002

2. F. Cavalli, R. Cucchiara, M. Piccardi, and A. Prati, "Performance Analysis Of Mpeg-4 Decoder And Encoder", International Symposium on Video/Image Processing and Multimedia Communications, Croatia, 2002

3. R. DeQueiroz, Z. Fan, and T. Tran, "Optimizing Block Thresholding Segmentation for Multilayer Compression of Compound Images", IEEE Transactions on Image Processing, 2000

4. F. Porikli and Y. Wang, "Automatic Video Object Segmentation Using Volume Growing And Hierarchical Clustering", Journal of Applied Signal Processing, special issue on Object-Based and Semantic Image and Video Analysis, January 2004.

5. S. Ji and H. W. Park, "Image Segmentation Of Color Image Based On Region Coherency", IEEE Transactions on Image Processing, 1998

6. F. Kossentini and Y. Lee, "Computation-Constrained Fast MPEG-2 Video Coding", IEEE Signal Processing Letters, vol. 4, n. 8, 224-226, 1997

7. O. Sukmarg and K. Rao, "Fast algorithm to detect and segmentation in MPEG compressed domain", Proceedings of IEEE Region 10 Technical Conference Malaysia, 2000

8. H. Wang and S.F.Chang, "Automatic face region detection in MPEG video sequences", Electronic Imaging and Multimedia Systems, SPIE Photonics China, 1996

9. R. Wang, H.J. Zhang, and Y.Q. Zhang, "A Confidence Measure Based Moving Object Extraction System for Compressed Domain", IEEE International Symposium on Circuits and Systems, Switzerland, 2000