Compressed Domain Video Object Segmentation

Fatih Porikli, *Senior Member, IEEE*, Faisal Bashir, *Member, IEEE*, and Huifang Sun

***Abstract*— We present a compressed domain video object segmentation method for the MPEG encoded video sequences. For a fraction of the raw domain analysis, compressed domain segmentation provides the essential *a priori* information to many vision tasks from surveillance to transcoding that require fast processing of large volumes of data where pixel-resolution boundary extraction is not required. Our method generates accurate segmentation maps in block resolution at hierarchically varying object levels, which empowers application to determine the most pertinent partition of images. It exploits the block structure of the compressed video to minimize the amount of data to be processed. All the available motion flow within a group of pictures is projected onto a single layer, which also consists of the frequency decomposition of color pattern. Then, by starting from the blocks where the spatial energy is small, it expands homogeneous regions while automatically adapting local similarity criteria. We also formulate an alternative solution that applies a kernel-based clustering where separate spatial, transform, and motion kernels are used to establish the affinity. We show that both region expansion and mean shift produce similar results as the computationally expensive raw domain segmentation. Finally, a binary clustering iteratively merges the most similar regions to generate a hierarchical partition tree.**

***Index Terms*— Compressed domain segmentation, mean-shift analysis, MPEG video, volume growing.**

I. Introduction

A. Motivation

VIDEO OBJECT segmentation, i.e., the detection of the constitutive parts of video frames, is conventionally performed in the spatial color domain, called *raw* data [1]–[5]. Since no motion information is readily available in raw data, the underlying flow is estimated from a pair of consecutive frames, often by employing block matching [6], [7], phase correlation [8], or gradient based approaches [9], [10].

To achieve a proper segmentation, accurate flow dynamics is required. Depending on the desired precision and resolution of the motion vectors, extraction of these dynamics can be computationally prohibitive. On the other hand, there is the challenge of processing incessantly growing amount of video data streaming from all kind of sources with limited resources in real time. To make things more complicated, any raw domain analysis involves the additional cost of decoding, as almost all video data is kept in a compressed format due to the storage space and transmission requirements. This inevitably brings the dilemma of processing the massive amounts of video data while maintaining an acceptable segmentation performance.

One favorable aspect of compressed video is that it already contains coarse but potentially useful motion cues encoded in its P-frames. Furthermore, the block structure of the compressed domain data drastically condenses the amount of data to be processed and reduces the processing time. Considering aforementioned need for fast yet competent solutions, it makes perfect sense to utilize this available information to improve the following raw domain segmentation. As a prior, a segmentation map that is obtained in compressed domain can effectively initialize regions in raw domain while supplying fundamentals such as motion and texture distributions of those regions. In other words, compressed domain segmentation can be imposed as a preprocessing step to decrease the computational load of the following stages. The importance of such priors becomes more apparent in processing of large HD resolution videos.

In addition, many applications do not demand detection of exact boundaries. For example, digital video recorders that are integrated into recent surveillance systems are designed to receive encoded streams from multiple network cameras and expected to find the locations of moving objects in all those streams in real time using the limited processing power encased in low-end CPUs. Detection of object motion and generation of backgrounds in the compressed domain effectively reduces the computational load of the object detection by filtering out the frames in which no object motion exists. Performing detection in compressed video noticeably relieves these systems.

Another application that benefits from compressed domain segmentation is video transcoding where the regions of interest have usually 8×8 block resolution. Since the segmentation is used to generate the object layers, it suffices to operate on the approximate boundaries. By applying compressed domain segmentation, transcoding from a frame-based coding schema to object-based schema is significantly accelerated. Compressed domain segmentation also enables efficient generation of video object descriptions to access visual information with the MPEG-7 standardization effort [16].

In this paper we propose a system to extract the object-based and global features from the compressed MPEG video using the motion vector information for video retrieval.

B. Background

Important as it is, video object segmentation (a survey is given in [5]) is proven to be one of the most challenging tasks in computer vision. It aims to fuse different modalities, i.e., color, texture, and motion, to bridge the semantic gap between

Manuscript received June 6, 2005; revised June 5, 2007. First version published April 7, 2009; current version published January 7, 2010. This paper was recommended by Associate Editor L. Onural.

F. Porikli and H. Sun are with the Mitsubishi Electric Research Labs, Cambridge, MA 02139 USA (e-mail: fatih@merl.com).

F. Bashir was with the Mitsubishi Electric Research Labs, Cambridge, MA 02139. He is currently with Retica Systems, Waltham, MA 02451 USA (e-mail: fatih@merl.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2009.2020253

the numerical information and the human perception. It has several important applications from indexing and retrieval to event detection.

In contrast to the large amount of effort spent on raw domain, only recently has compressed domain analysis gained attention [11]–[15], [18] thanks to the multimedia revolution proliferating down to handheld devices.

In this paper, we consider the MPEG compression scheme which converts a bit stream in terms of I (intra-compressed), P (forward predicted), and B (bi-directional predicted) frames. An I-frame is encoded as a single image, with no reference to any past or future frames. It stores the discrete cosine transform (DCT) information of the original frame. The DCT separates the signal into independent frequency bands. All I-frames are divided into 16 × 16 pixel macroblocks. Each macroblock consists of four 8 × 8 luminance (Y) blocks and two 8 × 8 chrominance (U, V) blocks. The P- and B-frames store the motion information and residues after motion compensation. The goal of motion compensation is to provide an approximate prediction for the macroblock. Motion-compensated prediction assumes that the current picture can be locally modeled as a translation of the pictures of some previous time. A P-macroblock is encoded as a 16 × 16 area of the past reference frame, plus an error term. The sequence of different frame types is called the group of pictures (GOP) structure. There are many possible I-, P-, B-frame arrangements. Since B-frames are interpolated from the remaining frames, we leave them out of our segmentation method.

Accompanying the motion, the MPEG compressed video embodies spatial information. For instance, the DCT coefficients carry image texture and gradient attributes. Block-based representation forms data in a block-resolution condensed space, which significantly expedites any processing. For certain applications, it is also computationally more convenient to process data in the compressed domain rather than decompressing it and then working on the raw domain.

It is worth mentioning that the compressed domain has intrinsic limitations. The DCT removes the spatial correlation of pixels within a block, and thus the precision of the segmentation degrades by the block dimension. The MPEG standard specifies how to represent the motion information; however, it does not specify how such vectors are to be computed. 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, MPEG motion vectors do not necessarily correspond to the true motion but the best matching of macroblocks.

Some compressed domain segmentation algorithms are concentrated on the spatial DCT coefficients. Wang *et al.* [23] segment faces where skin-tone statistics, shape constraints, and energy distribution of the luminance DCT coefficients are utilized to locate faces. De Queiroz *et al.* [19] partition JPEG images into specific regions such as those containing halftones, text, and continuous tone using an encoding cost map based on DCT coefficients.

As opposed to the DCT only approaches, Babu and Ramakrishnan [15], [16] aggregate motion vectors (MVs)

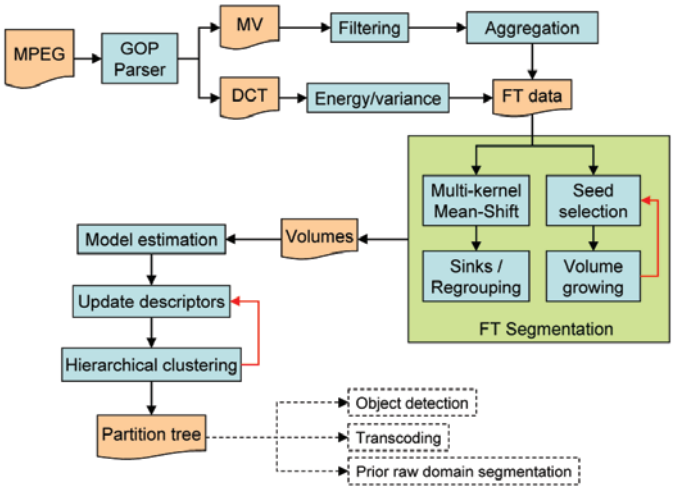

Fig. 1. Flow diagram of the volume growth-based compressed domain segmentation.

to segment the data. Similarly, Mezaris *et al.* [17] exploit the motion information to find spatiotemporal objects. Their method rejects motion vectors deviating from the single rigid plane assumption. As a result, certain blocks of the current frame are selected as a possible foreground regions. The selected blocks are filtered out in case they cannot be tracked to the previous frame. Remaining blocks are simply clustered to connected regions. Sukmarg and Rao [22] detect background and foreground regions using region segmentation followed by adaptive k-means clustering where the motion of a single P-frame is used and the remaining P-frames are omitted. Their results show that the preset thresholds in clustering impose a careful tuning of importance weights for different cues. A confidence measure-based moving object extraction method is proposed by Zhang *et al.* [23]. Their method introduces several confidence measures to help motion layer separation, and depends on a global motion compensation to differentiate layers. Ji and Park [20] track dynamic regions based on the DCT coefficient similarity and a binary classification constrained on block motion. As a shortcoming, this method requires initialization of individual regions.

C. Proposed Method

Based on the above observations, we present an automatic segmentation method that takes advantage of the embedded information available in compressed video and exploits the block and GOP structure to further accelerate the computational performance.

A functional flow diagram of the presented method is given in Fig. 1. After parsing the MPEG video into the DCT coefficients and MVs, we construct a multidimensional frequency-temporal (FT) data structure using not one but multiple GOPs between two scene cuts. Each GOP is restructured into a layer of feature vectors corresponding to grid of blocks. Each vector consists of a subset of dc coefficients, terms corresponding to various aspects of ac coefficients, and a set of accumulated forward-pointing motion vectors.

We achieve initial segmentation using two alternative solutions. The first method expand volumes within the FT data structure by starting from recursively derived seed points. The volume expansion marks the consistent and connected parts of video across each frame and across multiple frames. The seed points are selected from the points that have minimum local energy within their local neighborhood, which improves the likelihood of generating larger segments, and thus the consistency of final results.

The second method clusters the FT blocks in multiple kernels of spatial, motion, and frequency domains. For each point in the FT data, the iteration of the kernels enables us to find a sink point. The points converged to the same sink points constitute a segment. A nice property of the multikernel mean-shift method is that the similarity criteria are implicitly embedded into the algorithm, and thus it does not require fine tuning of the parameters. Since kernel size imposes a constraint on the granularity of the segmentation results, we use kernels to extract the minimum possible regions while preventing from oversegmentation. We present a comparison of these two solutions.

As a final stage, we iteratively merge the similar volumes using their descriptors to obtain a hierarchical object partition tree. At each iteration, we update the volume descriptors that consists motion and DCT-based terms. We tested different combination of these terms as well. We use a divergence score to assess the quality of the segmentation results at each level of the partition tree. Our experiments confirm the robustness of this score. After segmenting the FT data, we propagate the segmentation result to P- and B-frames.

Our method gives block accurate object boundaries. This information is essential in applications, which was explained in the previous section.

D. Main Contributions

Computational superiority is the main advantage of the proposed algorithm. As opposed to the existing approaches, our method processes a multitude of GOPs at the same time. Thus, it has significantly low computational load, which makes it ideal for processing of large volumes of streaming data with limited resources. The adapted block structure helps to reduce the computational load further by compacting data almost 360 times (Section II-B).

Given the unique properties of compressed domain data, this paper presents two advanced solutions for robust segmentation; using volume growth, and using mean-shift in FT data. Volume growth enables efficient propagation of intra-frame information to inter-frame segmentation. Multiple kernels in both I-frame DCT coefficients and P-frame motion vectors are used for kernel density gradient estimation in the spirit of mean-shift algorithm [25].

Another important advantage of the described solutions is that they adapt to the given data. Thus, we achieve robust segmentation with no cumbersome fine tuning of the parameters. Furthermore, we address the problem of extracting a representative motion vector for an I-frame block given the motion vectors of the individual P-frames. Toward this end, we propose motion accumulation of every block over all P-frames.

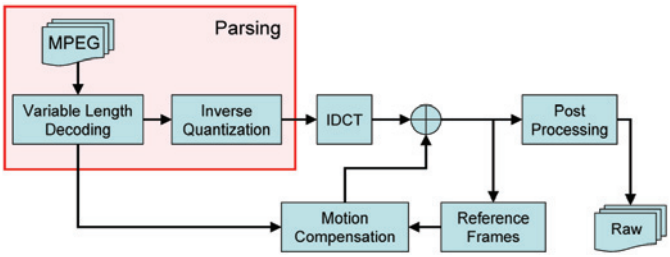

Fig. 2. Parsing requiring only constructing the DCT coefficients and motion vectors without IDCT and motion compensation.

In the following sections, we describe the details of the FT data structure, volume expansion, segmentation by kernels, and hierarchical merging to obtain a partition tree.

II. Frequency-Temporal Data

A. Parsing MPEG Video

Unlike the traditional parsers, we do not fully decode video down to the raw frames but only draw out the DCT coefficients C^t_{ijmn} and MVs $\mathbf{m}^{tk}_{mn} = [mx^{tk}_{mn}, my^{tk}_{mn}]^T$ where m, n are the macroblock indices, i, j are within-block indices, t is time, i.e., the corresponding GOP, and k is the order of P-frame in its GOP. The parsing components and other modules required to fully decode the MPEG video are shown in Fig. 2. As can be seen, we first chop the binary bitstream into bytes. At this point, all the DCT coefficients are in quantized format. Thus, we apply inverse quantization to find the integer-valued DCT coefficients. We reconstruct the scan lines of macroblocks by reshuffling the DCT coefficients. We obtain the MVs after variable length decoding.

The parsing process is computationally much simpler than the full decoding of MPEG video, which requires application of inverse DCT and motion compensation stages. On average, the parsing takes 3–10% of the decoding time (similar results have also reported in [21]) for a GOP. In our experiments, it takes approximately 0.5 ms to parse a GOP on a P4, 3-GHz CPU.

B. Frequency-Temporal Data Structure

After we parse the data, we assemble the DCT coefficients of I-frames and MVs of multiple P-frames in a GOP into a single layer of FT data. This data structure consists of multiple GOPs of a video shot. The visual content, i.e., the number of objects and their appearances, is assumed to remain consistent within the video shot.

Each element of the FT corresponds to a feature vector $f(t, m, n)$ that represents the attributes of a macroblock. Vectors that belong to the same GOP constitute a temporal layer. A feature vector consists of the dc parameters $C^{tY}_{00}, C^{tU}_{00}, C^{tV}_{00}$ of the I-frame for all Y, U, V channels, a reduced set of the ac parameters C^{tY}_{ij} where $i \neq j$ for Y-channel, a spatial energy term β, and the accompanying forward-predicted MVs $\mathbf{a}^t_{mn}$ obtained from the P-frames. C^{tY}_{00} represents the mean color and C^{tY}_{ij}'s indicate spatial texture.

The dc and ac components only exist for the I-frame. C^{tY}_{00} represents the mean color of the block, and thus it can be

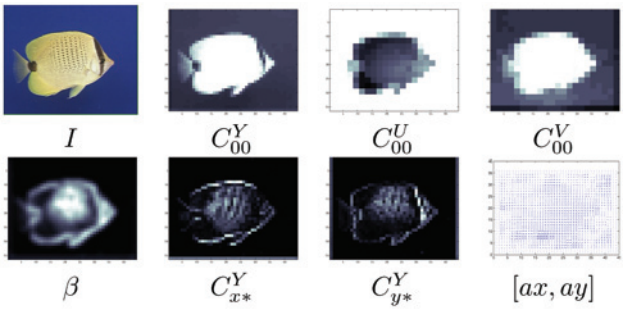

Fig. 3. Decoded I-frame and its corresponding layers in the frequency-temporal data.

considered as a pixel of the subsampled I-frame by a factor of 8 as shown in Fig. 3. Note that, not all the color channels are encoded in the same precision. Chrominance channels have half the resolution of the luminance channel due to the fact that human visual perception is more sensitive to the luminance variance. The DCT transform of an $M \times M$ image block is defined as

$$C_{ij} = \frac{2}{M} \sum_{x=1}^{M} \sum_{y=1}^{M} I(x, y) \cos \frac{\pi i (2x+1)}{2M} \cos \frac{\pi j (2y+1)}{2M} \tag{1}$$

where i and j are the horizontal and vertical frequencies ($i, j = 1, \ldots, M$), and $I(x, y)$ is a pixel. In the regular MPEG syntax, the block size is $M = 8$.

For a block in which the spatial texture is smooth, most of the higher indexed DCT coefficients have lower values and they reduce to zero after the quantization stage. Another key observation is that the higher DCT coefficients are sensitive to the pattern shifts. For instance, the spatially shifted versions of the same pattern will have different higher order coefficients. Since object movement will cause the shifted versions of the same pattern between the I-frames, the higher order, i.e., $i, j > 4$, coefficients will be different. The higher order terms are sensitive to speckle noise, which indicates that they are not confident cues in segmentation. Thus, we utilize the lower indexed DCT coefficients in the feature vector. To capture horizontal and vertical variance in a block, we define C_{x*}, C_{y*} using the luminance information as

$$C_{x*} = \sum_{i=2}^{K} C_{i1}^{Y}, \quad C_{y*} = \sum_{j=2}^{K} C_{1j}^{Y} \tag{2}$$

where $1 < K \leq M$, e.g., $K = 4$ to not include high order coefficients. For simplicity we dropped the time index. There is no need to normalize these numbers since the block size is constant. We tested the contribution of the diagonal variance. We observed that the diagonal coefficients provide minimal discriminating information mainly due to the fact that the vertical and horizontal components already include similar aspects.

We define an energy term β to measure the spatial variance as

$$\beta = \sum_{i=2}^{M} \sum_{j=2}^{M} C_{ij}. \tag{3}$$

The energy term does not contain the dc coefficients. In other words, the energy is measured in terms of the total magnitude of the ac coefficients. There is a strong correlation between the energy term and the accuracy of the estimation of the true motion. A block matching-based method often estimates incorrect vectors if the macroblock has smooth texture, and therefore a small β value. Higher β values point out block to be on a segment boundary.

In addition to features computed from the DCT coefficients, we include an aggregated motion information that is extracted using all the P-frames in the GOP. This aggregated motion information represents vertical and horizontal flow $\mathbf{a} = [ax_{mn}^{t}, ay_{mn}^{t}]^{T}$ of the corresponding block. We define the feature vector $\mathbf{f}$ of the block

$$\mathbf{f}(t, m, n) : \left[C_{00}^{Y}\ C_{00}^{U}\ C_{00}^{V}\ C_{x*}^{Y}\ C_{y*}^{Y}\ \beta\ ax\ ay \right]_{t,m,n}. \tag{4}$$

Feature vectors form the FT structure that is much smaller in size than its raw domain counterpart, which does not even contain any explicit motion information. For instance, a GOP of 15 consecutive frames of 352×288 color video occupies $352 \times 288 \times 15 \times 3$ pixels in raw domain, whereas the corresponding FT data has only $44 \times 36 \times 8$ components, which is equal to a reduction factor of 360:1. This shows the compactness of the FT data. Constructing FT data takes 1–2 ms for a GOP.

C. Aggregated Motion

In the case of motion estimation error for a P-frame block being high, the MPEG encoder marks it as intra-coded block and skips the assignment of MV to this block. Such intra-coded blocks occur quite frequently in the compressed bit stream whenever there is large motion in the scene. To find a regular motion field, we first interpolate a motion vector for each intra-coded block within its immediate local neighborhood. To minimize marginal extremities of the motion field, we convolve the regular field with a Gaussian filter. Since this filtering is blind, there is a tradeoff between removing extremities and keeping original boundaries. Any blind filter smears the boundaries; on the other hand, not removing the extremities causes over segmentation. We empirically determined that a 3×3 Gaussian-shaped kernel gives acceptable results for the sequences we tested.

Another problem of integrating the motion information into the feature vector is that the MVs of P-frames are back-predicted. For an I-frame and consecutive P-frame pair, only the blocks in the P-frame have their motion vectors that point the most similar placements in the I-frame. In other words, MVs of I-frame blocks do not readily exist.

To find the I-frame forward MVs, it is possible to accumulate motion or use a dense aggregated motion field. The first approach is illustrated in Fig. 4. In [15], motion accumulation is achieved by forward tracking of the reversed macroblock resolution MVs from I-frame to last P-frame. At every frame, the position of macroblock is updated by the corresponding MV. However, such a tracking either requires the quantization of the MVs in a coarse macroblock-size resolution, or recomputing the MV of the destination macroblock. Since the

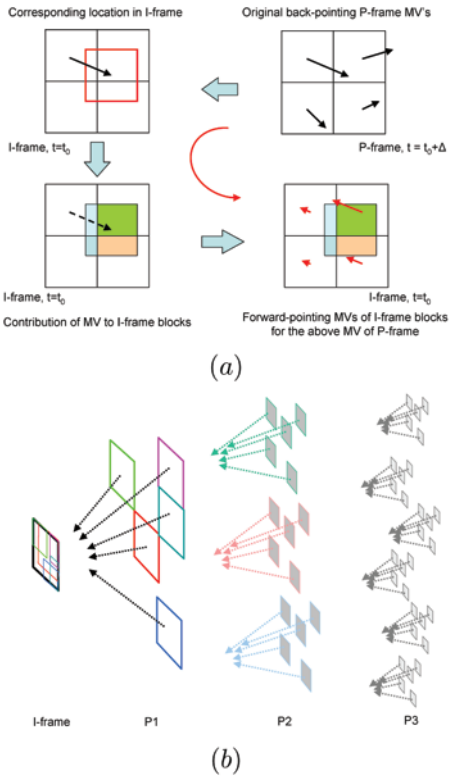

Fig. 4. (a) One P-frame motion vector contributing at most four I-frame macroblocks and (b) forward motion projection exponentially branching out for more than one P-frames.

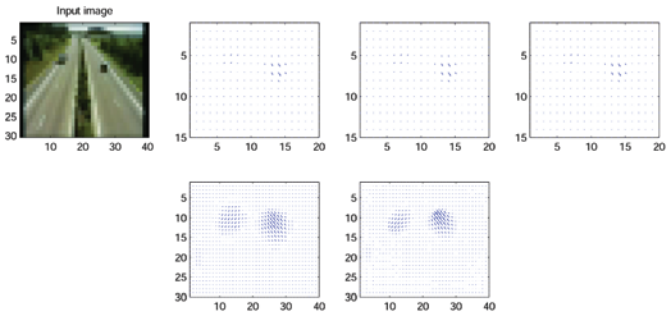

Fig. 5. **Top:** Input I-frame image along with motion vectors of individual P-frames. **Bottom:** Aggregated motion vectors for the GOP using forward projection and motion accumulation.

destination macroblock may not necessarily correspond to an original grid macroblock, its MV should be interpolated from the macroblocks that it overlaps. The MVs of the overlapped I-frame blocks are weighted by the ratio of the overlap to the total coverage. It is true that for an I-frame block that is entirely covered by the projected P-frame blocks, this process is more accurate than another frame that is partially covered. Besides, the tracking approach can only be applied to the adjoining frames as the projected motion degrades rapidly [Fig. 4(b)].

As an alternative approach, we interpolate the filtered MVs $\mathbf{m}^k_{mn}$ of the kth P-frame such that we get a pixel-resolution dense vector field $\mathbf{m}^k_{xy} = [mx^k_{xy}, my^k_{xy}]^T$. We repeat the interpolation for all the P-frames $k = 1, \ldots, K$ of the GOP. Then, by starting from the last P-frame $k = K$, we trace back the movement of the pixel back to first P-frame $k = 1$. The MVs $\mathbf{m}^*_{xy}$ connect the first and last P-frames. To find them we first initialize

$$mx^{*K}_{xy} = mx^K_{xy} \tag{5}$$
$$my^{*K}_{xy} = mx^K_{xy} \tag{6}$$

and starting from the last P-frame, we iterate the following equations

$$mx^{*k-1}_{xy} = mx^{*k}_{xy} + mx^{k-1}_{xy} \tag{7}$$
$$my^{*k-1}_{xy} = my^{*k}_{xy} + my^{k-1}_{xy} \tag{8}$$

for $k = K-1, \ldots, 2$ to assign the pixel-wise aggregated motion

$$\mathbf{m}^*_{xy} = \left[mx^{*1}_{xy} + mx^{*1}_{xy}\right]^T. \tag{9}$$

As a last step, we compute the mean of the aggregated pixelwise MVs to determine the aggregated motion information embedded in the feature vectors

$$\mathbf{a} = [ax_{mn}, ay_{mn}]^T = \frac{1}{z} \sum_{x,y \in block_{mn}} \mathbf{m}^*_{xy} \tag{10}$$

where z is the number of pixels in a block e.g., $z = 64$ or $z = 256$ depending on the granularity. We give examples of forward projection and aggregation in Fig. 5.

III. Segmentation

A. Volume Growth

We iteratively expand the volumes one by one within the FT data starting from the seed points.

These volumes may extend between multiple GOPs. Volume growth arranges the featurewise similar FT points into a consistent segment. Since expansion is based on local similarity, the seed points should properly represent their 3-D local neighborhood. A point that has low local variance is a suitable candidate. We approximate local variance as the amount of the energy in temporal and spatial neighborhood

$$\sigma^2_{t,m,n} = \sum_{k=-h}^{h} \sum_{i=-h}^{h} \sum_{j=-h}^{h} \left(\beta_{t+k,m+i,n+j} - \hat{\beta}\right)^2 \tag{11}$$

where $\hat{\beta}$ is local mean energy, and $h \approx 1$ is a small window kernel size. Again, we drop the scaling constant in the variance formulation since its ineffective when we determine the minimum value. We select the current seed point p_{seed} as the minimum variance in the remaining set of unsegmented points

$$p_{\text{seed}} = (t, m, n)_{\text{seed}} = \arg\min\left(\sigma^2_{t,m,n}\right). \tag{12}$$

The seed selection is a relatively intensive task since it involves searching for the minimum. This process can be speeded up by performing search in separate GOPs. In other words, the local minimum in the current GOP is searched and a volume in 3-D FT data is grown. Then, the next seed is searched from another GOP.

After we select the current seed point, we initialize a volume descriptor $\mathbf{v}$ using the feature vector $\mathbf{f}(p_{\text{seed}})$ of the seed point p_{seed}. We define a set of active boundaries that keeps most recently added but not processed points to the current volume. At the first iteration, the active boundaries set has only the seed point inside.

Then, we compare the adjacent points of the each active boundary point to the current volume descriptor $\mathbf{v}$. We evaluate six neighbor points, i.e., $(t-1, m, n)$, $(t+1, m, n)$, $(t, m-1, n)$, $(t, m+1, n)$, $(t, m, n-1)$, and $(t, m, n+1)$, in all three dimensions. Inclusion of the points in temporal dimension enables imposing the assumption that a segment overlaps between the consecutive GOPs. This assumption may be violated in case the GOPs have a large number of P-frames. This implies that the temporal inter-I-frame disparity of two consecutive GOPs will be large. This effect is particularly exacerbated in the presence of fast moving objects. In our observations, the likelihood of having overlaps is statistically high since MPEG GOPs usually consist of 2 to 15 P-frames. We compute the distance of a feature vector $\mathbf{f}$ of the adjacent point to the volume descriptor $\mathbf{v}$ as

$$d(\mathbf{v},\mathbf{f}) = \omega_{00} \sum_{l=Y,U,V} \left|\mathbf{v}_{C_{00}^{l}} - \mathbf{f}_{C_{00}^{l}}\right| + \omega_{*} \sum_{l=x,y} \left|\mathbf{v}_{C_{l*}^{Y}} - \mathbf{f}_{C_{l*}^{Y}}\right| + \omega_{\beta}|\mathbf{v}_{\beta} - \mathbf{f}_{\beta}| + \omega_{a}|\mathbf{v}_{\mathbf{a}} - \mathbf{f}_{\mathbf{a}}| \tag{13}$$

where ω_{00}, ω_{*}, ω_{β}, and ω_{a} regulate the contribution of the DCT coefficients and motion information. Each weight is inversely proportional to the amount of standard deviation of the corresponding entity to normalize its magnitude

$$\omega_{k} = \left(\sum_{\mathbf{f} \in GOP_{0}} (\mathbf{f}_{k} - \hat{\mathbf{f}}_{k}) \right)^{-1} \tag{14}$$

where k indicates a component, e.g., C_{00}^{Y}, β, etc., of the feature vector, GOP_0 is either the first GOP of the data or multiple GOPs of training data to adapt the weights to the given content. We denote the mean of the feature vector by $\hat{\mathbf{f}}$. Using inverse standard deviation enables giving equal importance to each component by suppressing the dominant components that exhibit larger variance. In other words, these weights are adapted to the given input data.

In case the distance $d(\mathbf{v},\mathbf{f})$ is less than a threshold, which is set within the range $1 \pm \epsilon$, the point is included in the active boundary and the volume descriptor $\mathbf{v}$ is updated by the feature vector $\mathbf{f}$ by alpha blending. We observed that the threshold determines the precision of the segmentation process and thus the average size of the volumes. Since we have already scaled the distance with respect to the standard variation, we can choose a threshold value that does not require fine tuning. We select a threshold that prevents from undersegmentation, and so the above range is sufficient. We overcome the oversegmentation problem by fusing the similar volumes in the hierarchical clustering stage. In our experiments, we set the threshold to 1.

The process of volume growth continues until no active boundary point remains, i.e., none of the neighbors of the volume surface points is similar to the volume descriptor. After a volume is grown, all the points included in the volume are marked. The seed selection and volume growth process are iterated until no more point remains in the FT. As a post-processing stage, the volumes that have negligible size are removed, and the remaining volumes are inflated to fill up these removed empty spaces. The seed selection and volume growth take 0.8–1.5 ms for a GOP on average.

B. Multikernel Mean-Shift Segmentation

Recently, iterative gradient maximization methods, particularly mean-shift filtering, have become popular in color image segmentation [24], [25]. Here, we extend the previous work for compressed domain segmentation by integration motion cues and multiple kernels.

Basically, mean-shift segmentation assigns each point in the data space to a sink point. By iteratively evaluating the local gradient direction within a local search region, called kernel, the *mean* of the kernel is shifted in the space. The point where this shift operation moves and converges to the kernel is called its sink point. Depending on the kernel size, multiple data points can be assigned to a single sink point.

We use not only a single spatial kernel but also multiple kernels in different spaces. One additional multidimensional kernel is defined in DCT coefficient space. Another 2-D kernel is defined in the MV space. The original spatial kernel is in three dimensions since the FT data has both spatial and temporal dimensions. At each mean-shift iteration, we shift these kernels in the corresponding spaces while updating the properties, i.e., coverage of the gradient computation, simultaneously.

Let $\mathbf{f}_i$ and $\mathbf{f}_j^{*}$ be the original and sink points in the spatial, DCT, and motion domains as feature vector coefficients exist in all of these domains. For the sake of clarity, we replace the t, m, n indices by j, where $j = 1, \ldots, P$. The superscripts s, r, v will denote the spatial, DCT, and motion parts of the vectors, respectively. Furthermore, let the FT data be already normalized with the standard deviations of the corresponding features. To find the sink points, we apply the following algorithm for each point in the FT:

1) initialize $k = 1$ and $\mathbf{h}_k = \mathbf{f}_j$;
2) compute $\mathbf{h}_{k+1} = \frac{1}{P_k} \sum_{\mathbf{f}_i \in K(\mathbf{h}_k)} \mathbf{f}_i$; $k \leftarrow k+1$ until convergence, i.e., $|\mathbf{h}_k - \mathbf{h}_{k+1}| < \epsilon$;
3) assign $\mathbf{f}_j^{*} = (\mathbf{f}_j^{s}, \mathbf{h}_{conv}^{r,v})$.

The assignment specifies that the sink point at the spatial location of $\mathbf{f}_j$ will have the DCT and motion components of the point of convergence $\mathbf{h}_{conv}$. The number of points in the window $S(\mathbf{h}_k)$ is P_k.

For segmentation, we find the sink points first using the above algorithm. Then, we identify clusters of sink points by linking together all $\mathbf{f}_j^{*}$ which are closer than a preset value from each other in the joint domain. Since we assume we normalized all the coefficients, this value can be set to 0.5.

We observed that using all the DCT coefficients does not provide stable segmentation results and causes excessive over segmentation. The described method becomes sensitive to the parameters when we include the ac components and the energy term. This is expected considering the fact that the mean shift

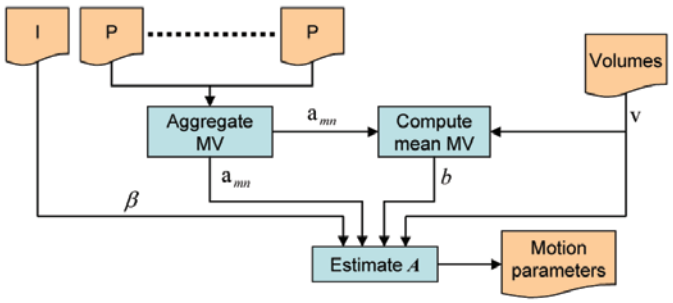

Fig. 6. Affine motion parameters estimated using translational motion as prior.

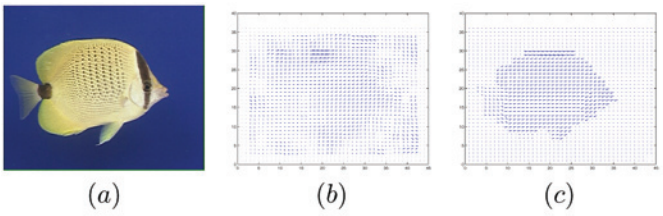

Fig. 7. (a) A frame from *Bream*, (b) original MPEG motion vectors interpolated for 8×8 blocks, and (c) motion vectors after parameter estimation.

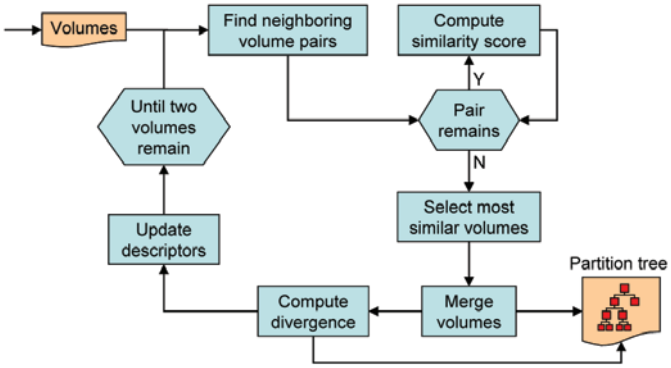

Fig. 8. Two most similar volumes merged at each iteration of hierarchical clustering.

will assign a different sink anywhere a point has different ac coefficients than its neighbors. Thus, we used only the three DCT coefficients in our reported results.

In the final stage of the mean-shift segmentation, we group together the points that are assigned to same cluster of sink points. This group of points constitute a volume. As a postprocessing step, the small volumes are eliminated and the adjacent volumes are inflated to regroup the unassigned points. The described multikernel mean-shift segmentation takes $3 \sim 5$ ms for a GOP.

IV. Hierarchical Clustering

Segmentation algorithm generates volumes as well as their attributes and information about how these volumes should be merged. Since the use of its results is ultimately determined by the specific application, we generate a partition tree by hierarchical clustering and leave the selection of the level in this tree to the application.

After volume growth, we obtain the parts of the FT data that are consistent in DCT coefficients and translational motion. To find a more refined motion model, we fit a motion model to each volume.

A. Motion Model

We estimate affine motion parameters at each GOP of a volume and then average the set of individual parameters over all of the GOP layers. In this way, we solve the region of support problem of motion segmentation by using the segmented regions of the temporal layers. Motion parameter estimation is illustrated in Fig. 6.

We model the layerwise motion $p \rightarrow p'$ by a set of affine motion parameters A, b between the points of a volume in one GOP layer to next layer as

$$p' = p + \begin{bmatrix} ax_{mn} \\ ay_{mn} \end{bmatrix}_p = Ap + b = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} p + \begin{bmatrix} b_x \\ b_y \end{bmatrix} \quad (15)$$

where $b = [b_x, b_y]^T$ is the translational motion. We dropped the macroblock size multiplier from above for clarity. There are two possible translational motion: the average of the motion vectors $\mathbf{a} = [ax_{mn}, ay_{mn}]^T$, and the trajectory displacement. Trajectory is defined as the string of layerwise (i.e., GOP-wise) center of the corresponding regions of a volume. It is calculated by averaging the coordinates of the points belonging to the volume in a GOP. After finding the trajectory, we use trajectory displacement layers as translational motion. We observed that using the mean is a more accurate approach since trajectory of a large background object tends to be zero.

For a region that consists of K points, we rewrite the above equation as

$$\begin{aligned} A\,[p_1 \,|\, \ldots \,|\, p_K] &= [p_1 + \mathbf{a}_1 - b \,|\, \ldots \,|\, p_K + \mathbf{a}_K - b] \\ A_{(2\times 2)} Q_{(2\times K)} &= \Lambda_{(2\times K)} \end{aligned} \quad (16)$$

where the only unknown is the matrix A. Since P is an $2 \times K$ matrix, $A = \Lambda Q^{-1}$ is the solution in the least-squares sense to the overdetermined system of equations. The effective rank and pseudo inverse is determined from the QR decomposition with pivoting. This process takes 5–8 ms for a GOP. Sample MVs after parameter estimation are shown in Fig. 7.

B. Construction of Partition Tree

We cluster the segmented volumes into objects using their descriptors and motion models as shown in Fig. 8. Clustering can be done either hierarchically or in a partitional manner. Hierarchical approach produces a nested series of partitions, while a partitional approach obtains a single partition of the data. We adapt a hierarchical clustering technique by merging the volumes in a fine to coarse manner.

At each iteration of the hierarchical clustering, we merge the pair having the maximum similar score. We define the similarity between the volumes as

$$s(\mathbf{v}_i, \mathbf{v}_j) = \left[1 + \sum_t \left(c_A |A_{i_t} - A_{j,t}| + c_b |b_{i,t} - b_{j,t}| \right) \right]^{-1} \quad (17)$$

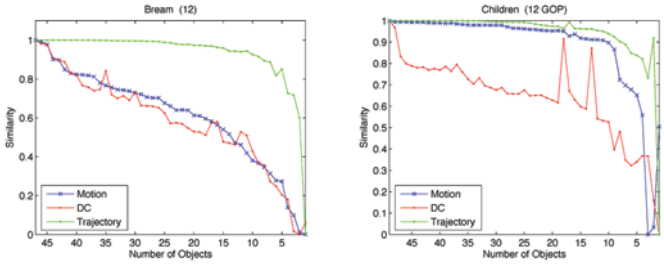

Fig. 9. Motion, trajectory, and DCT coefficient-based similarity scores of the merged pairs for sample sequences.

where t is the GOP layers on which both volumes are visible. The mixture constants are set as $c_A \ll c_b$ to take into account of the fact that a small change in the rotation and scaling parameters can lead to much greater disparity than the translation parameters. We update the descriptors, i.e., the motion parameters of the volumes, accordingly after each merger. The hierarchical clustering is iterated until two volumes remain.

At each level of the clustering algorithm, we evaluate whether the chosen volume pair is a valid merger. We keep track of the change in the similarity term. The sudden drops and small values of the similarity score indicate an invalid merger. We should note that the consistency of this score depends the definition of the similarity. We observed that the motion parameter-based similarity scores are robust, i.e., they give smooth and monotonically decreasing scores while providing accurate clustering results as shown in Fig. 9. The trajectory-based motion similarity score is found to be not as consistent as the motion parameter metric score since it disregards the rotation and is sensitive to the shape of the object, e.g., larger objects tend to have less descriptive trajectories due to the averaging of the positions of its constituent blocks. Similarly, the DCT coefficient-based similarity causes wrong merges since it disregards the motion information.

Based on the above observations, we define a divergence score α_L to estimate the optimum number of segments. This score indicates how the consistency of the motion information changes after the merge of the selected volumes as

$$\alpha_L = \sum_{l=1}^{L} N_l \sigma^2_{\mathbf{a},l} \tag{18}$$

where $\sigma^2_{\mathbf{a},l} = \sigma^2_{ax,l} + \sigma^2_{ay,l}$ is the variance of the aggregated motion vectors of the volume l. N_l is the number of the points in the volume. At level L of the partition tree, there are L volumes. The divergence is computed over all volumes. Since we use variance of the MVs, the divergence score corresponds to the total weighted variance of the MVs among all segments. We used the aggregated MVs instead of the MVs of the P-frames. Using all P-frame MVs may cause an inflated variance even though the motion at a P-frame is consistent.

The divergence score yields small values for valid mergers, and high values for invalid associations. Thus, by evaluating the minima of the divergence score, we can determine the correct cluster number automatically.

C. Segmentation of P- and B-Frames

After we obtain the partition tree, we propagate the object boundaries determined for the I-frames to the P- and B-frames. We utilize the already computed dense and filtered MVs $\mathbf{m}^k_{xy} = [mx^k_{xy}, my^k_{xy}]^T$ of the corresponding P-frames for this purpose. Let R^0 be the region of a volume in I-frame. The corresponding region in a P-frame is obtained by

$$R^{k+1}(x, y) = R^k\left(x + mx^k_{xy}, y + my^k_{xy}\right). \tag{19}$$

For block resolution segmentation, we simply assign the most common label in the block. Labels of the uncovered blocks are assigned from their neighbors. We preferred to use the dense vectors instead of the block vectors to improve the quality of the boundaries.

The labels of the blocks in B-frames are assigned from the neighboring P-frame labels as no B-frame residual information is decoded in the parsing stage. To properly segment B-frames, the B-frame block labels should be estimated from the neighboring frames using the corresponding forward and backward MVs.

V. Experiments

For our experiments, we used video shots from standard test sequences with varying number of GOPs ranging from 1 to 40, and with different number of P-frames from 1 (IPIP..) to 15 (IBPBP..). We constructed the FT data structure using the 12 GOPs that contain one I-frame, four P-frames, and three B-frames. This syntax is a common configuration and corresponds to a synchronous segmentation of 96 frames of the original raw video. The pixel resolution of the raw data was 352×288, and block resolution in luminance was 44×36.

In addition to standard configuration, we tested constructing the FT data with different number of P-frames, e.g., employing only the first P-frame. In other words, motion aggregation had only one stage. We observed that the inclusion of the following P-frames increases the computational load as expected; however, it does not necessarily improve the segmentation results.

We set the volume growth threshold to 1 in all tests. As we explained before, any threshold value that prevents undersegmentation would suffice for our purpose since the clustering stage is designed to merge oversegmented volumes.

We tested different variance-based weights ω_{00}, ω_*, ω_β, and ω_a, when we computed the distance between the volume descriptor and the candidate feature vectors. We set certain weights to zero for the volume growth using DCT coefficients ($\omega_\beta = \omega_a = 0$), dc values ($\omega_* = \omega_\beta = \omega_a = 0$, and MVs ($\omega_{00} = \omega_* = \omega_\beta = 0$). We observed that higher values of the DCT distance provide more accurate segmentation results in case of the slow motion, e.g., for *Akiyo* and other head-and-shoulder sequences. We verified that the motion information provides more discriminating information for the sequences that contain fast moving objects. In favor of interlayer expansion, we locally adapted the feature weights depending on whether the candidate point is an interlayer point or an intralayer point. However, we noticed occasional volume leakage problems especially around the inaccurate MVs. As a postprocessing step, we removed tiny volumes.

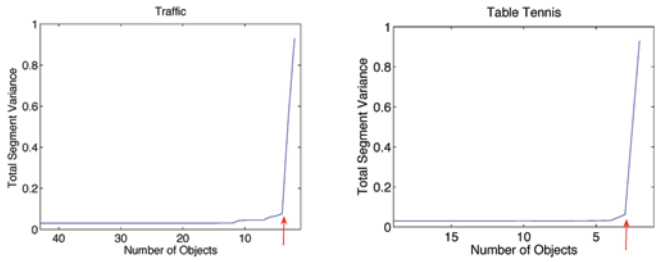

Fig. 10. Divergence score α_L for two sequences. Arrows indicate the optimum object number before an invalid merge.

Segmentation	Clustering	Weighted Error
Mean-Shift DC	MV	38
Mean-Shift DC	DCT+MV	57
Mean-Shift MV	DCT	93
Mean-Shift MV	DCT+MV	100
Mean-Shift DC+MV	DCT+MV	78
Volume Growing DCT	MV	41
Volume Growing DCT	DCT+MV	66
Volume Growing MV	DCT	72
Volume Growing MV	DCT+MV	77
Volume Growing DCT+MV	DCT+MV	84

Fig. 11. Weighted average error rates for different combinations of features in segmentation and hierarchical clustering. Table shows the average errors for six sequences. Individual errors are weighted according to the GOP number of the corresponding sequence.

Figs. 12–17 show the initial segmentation results (marked as *initial*) for an I-frame of the first GOP layer for each test sequence after the volume growth stage. As can be seen, the objects that have similar DCT coefficients and MVs are accurately detected even at the coarse block resolution, which shows the effectiveness of the compressed domain segmentation process.

Fig. 12 shows an I-frame, the initial segmentation and the partition tree for the *Akiyo* sequence, where the divergence score indicates that the optimum number of clusters is 2. These objects are segmented as the head and the background, since the head of the speaker has the most discriminating movement in that group of GOPs.

Fig. 13 presents the segmentation results for the *Lab* sequence. The divergence score suddenly changes at the clustering level 2, which shows there should be two clusters, i.e., person and background.

Fig. 14 gives results for the *Bream* sequence. We observed that the divergence score gives the optimum at the clustering level 3 due to the fact that the upper fin of the fish has different movements. Ideally, the optimum number should be 2; however, not all regions of the fish has the same motion.

We show an I-frame from the *Children* sequence and its partition tree in Fig. 15. The computed divergence score changes suddenly after the clustering level 3, which indicates that optimum segments are the moving head of the boy on the left, the body of the boy on the right, and the background.

The segmentation results for a GOP of *Traffic* is given in Fig. 16. There are three distinct objects: vehicle on the left, vehicle on the right, and the stationary background. As shown

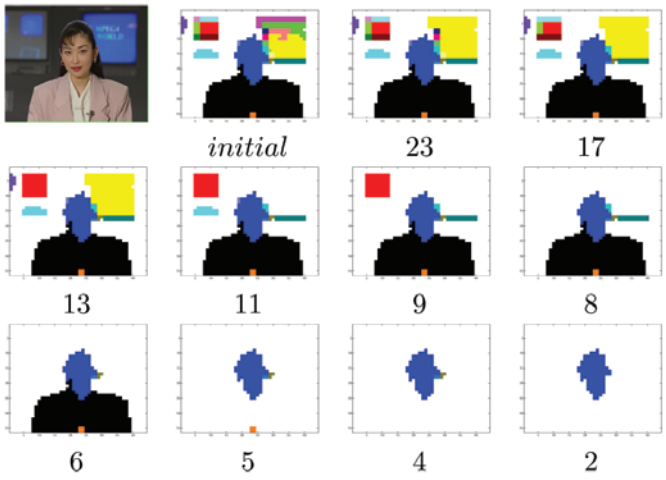

Fig. 12. I-frame from *Akiyo*, and the segmentation results at the corresponding clustering levels including the background. Both divergence score and similarity-based metrics indicate that the optimum number of clusters is 2.

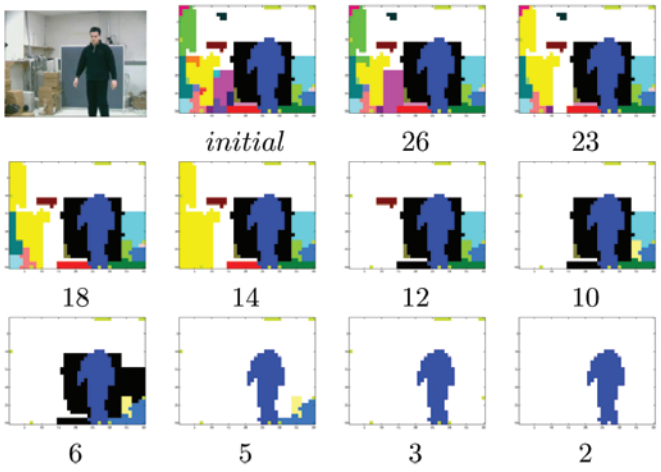

Fig. 13. I-frame from *Lab*, and the segmentation results at the corresponding clustering levels. Validity metric jumped at level 2, which indicates there should be two clusters, i.e., person and background. Different volumes are randomly colored.

in Fig. 10, the divergence score for this video jumped when we tried to merge the three remaining objects. This indicates that the merger is invalid, and objects should not be merged any more.

Similarly, for the *Table Tennis* sequence the divergence score estimates the optimum number of clusters as 2—the ball and the background. This is justifiable since only the ball moves in that GOP. The final partition tree contains the table and the arm on that layer, as shown in Fig. 17.

Overall, we observed that the total segment variance suddenly increases in case of an invalid merger. These results confirm the motion existing in the scene, and proves the effectiveness of the proposed algorithm even if the objects are small in comparison to the frame size.

As seen from these results, the motion parameter-based similarity measure can detect the small motion variances. Although a fast moving single small object may invalidate

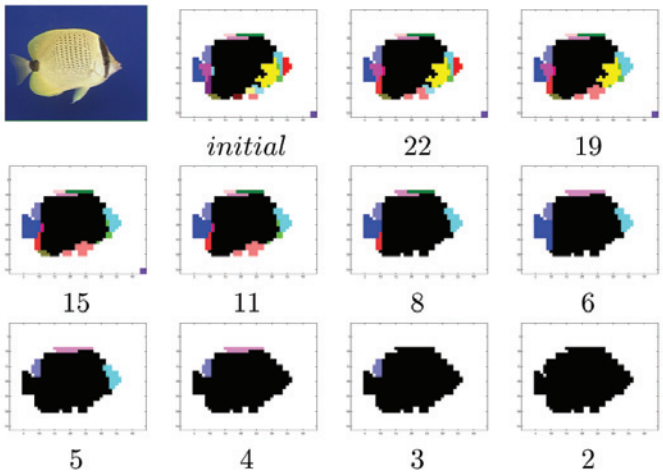

Fig. 14. I-frame from *Bream*, and the segmentation results at the corresponding clustering levels. Divergence score indicates that the segmentation is optimum at the level 3 due to the fact that the upper fin of the fish has a different movement.

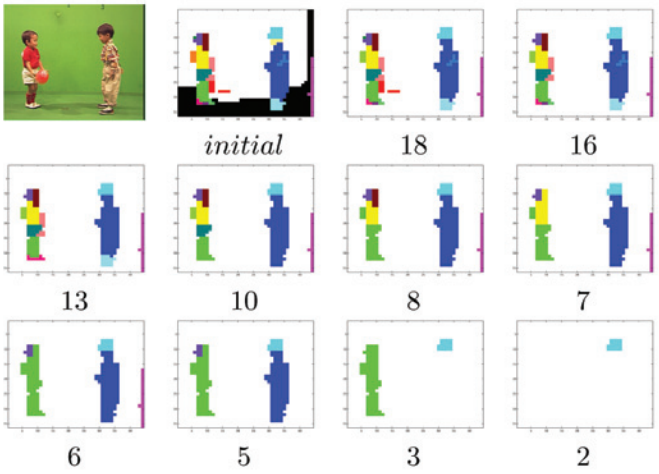

Fig. 15. I-frame from *Children*, and the segmentation results at the corresponding clustering levels. Divergence score indicates the optimum number of clusters should be 3; moving head of the boy on the left, body of the boy on the right, and background. However, our system enables the end user to choose any level in the object tree.

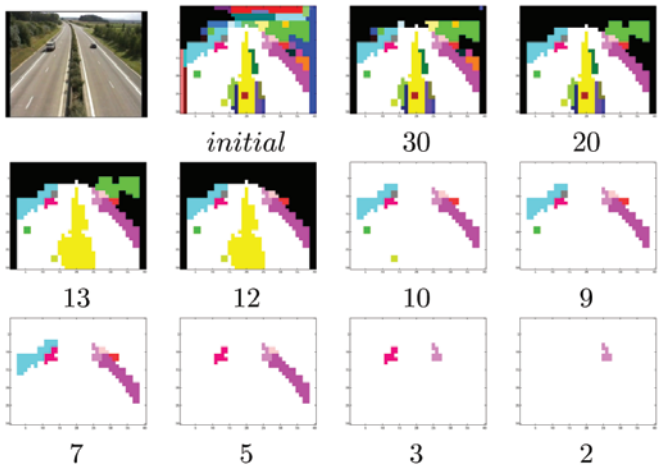

Fig. 16. I-frame from *Traffic*, and the segmentation results at the corresponding clustering levels. The validity score indicates the optimum segment number should be 3 including the background.

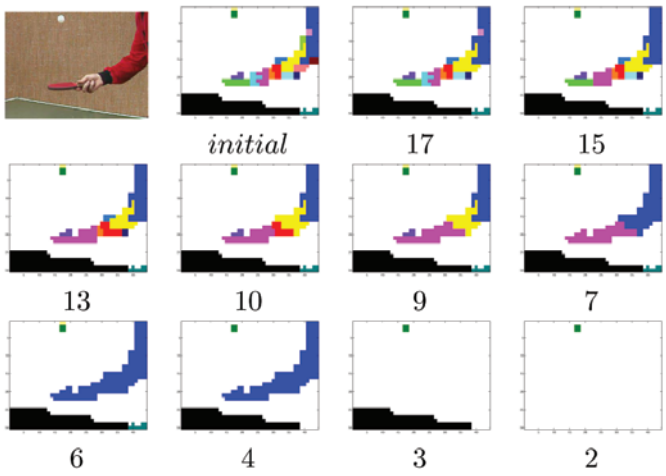

Fig. 17. I-frame from *Table Tennis*, and the segmentation results at the corresponding clustering levels. The Divergence score for this sequence jumped at level 2. There are four distinct motion in this GOP; ball, arm, table (due to camera motion), and wall (static background). We observed that the hand and arm have similar motion vectors. The spurious segment on the lower right has inaccurate motion vectors. The volume growth process could not blend it into other regions since its DCT coefficients were also significantly different. On the other hand, the white ball has the largest motion vector, and thus it stayed as a distinct object until to the final level.

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.

We have also conducted experiments to test the performance of multikernel segmentation using mean shift. For an objective analysis, we have used the same experimental setup and video sequences as explained previously. For the sake of qualitative comparison, we give the detailed results of incremental segmentation steps involved for a subset of algorithms in Fig. 18 for *Traffic*, and in Fig. 19 for *Table Tennis* sequences.

We measured the accuracy of the various combinations of FT segmentation on different feature spaces using either volume growth or mean shift:

1) segmentation by mean shift on dc values; hierarchical clustering by MVs;
2) segmentation by mean shift on dc values; clustering by DCT and MVs;
3) segmentation by mean shift on MVs; clustering by DCT coefficients;
4) segmentation by mean shift on MVs; clustering by DCT coefficients and motion vectors;
5) segmentation by mean shift on dc values and MVs; clustering by DCT coefficients and MVs;
6) segmentation by volume growth on DCT coefficients; clustering by MVs;
7) segmentation by volume growth on DCT coefficients; clustering by DCT coefficients and MVs;

Fig. 18. *Traffic* sequence. **Column 1**: Mean-shift segmentation using DCT coefficients, hierarchical clustering using motion vectors. **Column 2**: Mean-shift segmentation using motion, hierarchical clustering using DCT. **Column 3**: Mean-shift segmentation using DCT and motion vectors, hierarchical clustering using DCT and motion. **Column 4**: Region growth using DCT, hierarchical clustering using motion. **Column 5**: Region growth using motion, hierarchical clustering using DCT. **Column 6**: Region growth using DCT and motion, hierarchical clustering using DCT and motion.

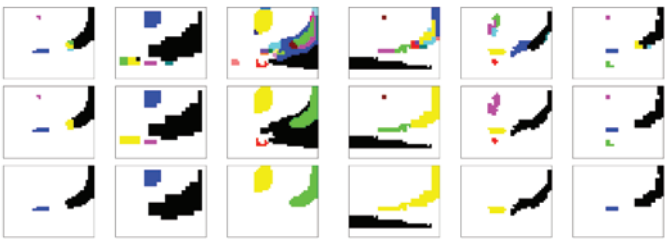

Fig. 19. *Table Tennis* sequence. **Column 1**: Mean-shift segmentation using DCT coefficients, hierarchical clustering using motion vectors. **Column 2**: Mean-shift segmentation using motion, hierarchical clustering using DCT. **Column 3**: Mean-shift segmentation using DCT and motion vectors, hierarchical clustering using DCT and motion. **Column 4**: Region growth using DCT, hierarchical clustering using motion. **Column 5**: Region growth using motion, hierarchical clustering using DCT. **Column 6**: Region growth using DCT and motion, hierarchical clustering using DCT and motion.

8) segmentation by volume growth on MVs; clustering by DCT coefficients;
9) segmentation by volume growth on MVs; clustering by DCT coefficients and MVs;
10) segmentation by volume growth on DCT coefficients and MVs; clustering by DCT coefficients and MVs.

These algorithms are applied on DCT coefficients or MVs for segmentation. Also, the hierarchical clustering for region merging is operated on either feature space, i.e., DCT coefficients and MVs. This way, we have generated a matrix of various segmentation techniques. The results on this dense matrix of algorithms are reported in Fig. 11.

We manually labeled object boundaries for several GOPs from various sequences and computed the segmentation error, which is defined as the amount of mismatch between the labels in ground truth and segmentation result. We assumed that an object in the segmentation results belongs to the object that has the maximum overlap in the ground-truth data. Note that segmentation task is application specific; a universal ground truth does not exist except for very simple examples. This is the main motivation to use a partition tree to let the end user to assess the performance.

Fig. 11 shows the average normalized mean-square errors between binary mask at final (optimal) level of the clustering stage. We weighted errors across the sequences with respect to the number of GOPs and normalized them by assuming that the worst error is 100. It can be inferred from the table that both the presented segmentation methods produce similar results for the same configurations. Using DCT terms in FT segmentation and MV in the consecutive clustering, on average, gives better results.

TABLE I
AVERAGE COMPUTATION TIMES (352×288 VIDEO; 44×36 BLOCKS)

Parsing	0.5 ms
FT generation	0.7 ms
Seed selection and volume growing	2 ms
Multikernel mean shift	3 ms
Motion parameter estimation	6 ms
Hierarchical clustering	3 ms

It is also seen from that table that dividing the segmentation into two separate tasks (FT segmentation and hierarchical clustering) generates more accurate object boundaries than segmenting into a joint space (mean shift DCT + MV and volume growing DCT + MV).

The proposed algorithm is faster than the real-time one. The total segmentation time including the MPEG parsing is approximately 12 ms for a GOP on a P4 3-GHz platform. The load of each stage is shown in Table I. These numbers may vary depending on the number of initial objects. Most computational load is on the motion model estimation stage. Favorably, the speed is not influenced by the complexity of the motion. Since a GOP consists of eight frames in the above tests, the compressed domain segmentation achieves an average of average 1.5 ms processing speed per frame, which is significantly faster than any raw domain segmentation method.

A. Limitations

Compressed video MVs are often computed by minimizing a cross-correlation error. They map to the most similar underlying patterns in the reference frames, not necessarily to the actual motion of the objects. Therefore, performance of any compressed domain algorithm is limited to the accuracy of the MPEG motion vectors. For sequences that contain fast moving objects, this issue becomes more apparent, as most encoders automatically switch into the intra-frame coding for fast moving blocks where block motion estimation fails to provide a valid match.

Volume growth inherently imposes objects to have overlapping regions between the consecutive GOPs. Otherwise, a single object may be divided into multiple objects along the temporal axis. Still, tracking of an object across multiple GOPs is not required for most applications.

Compressed domain segmentation generates low-resolution segmentation masks, which are insufficient for shape-based detection and retrieval tasks. To improve the spatial resolution, a raw domain-boundary refinement should be applied.

VI. CONCLUSION

This paper has presented a compressed domain segmentation method that takes full advantage of the inter-frame

motion and intra-frame spatial frequency information already embedded in MPEG encoded video. The primary contribution is a volumetric segmentation and an iterative clustering-based scheme for generation of a hierarchical partition tree to select the most desired segmentation based on the application specifications.

Our tests revealed that a slight oversegmentation using DCT coefficients followed by aggregated motion-based clustering produces more accurate boundaries than single-stage joint segmentation. This agrees with our intuition; the volume boundaries obtained from DCT coefficients fits into the underlying video more closely than the motion boundaries, which tend to be deformed and erroneous since the original MVs are generated to optimize the compression efficiency.

We observed that using all of the DCT coefficients do not necessarily provide a stable segmentation. For instance, mean-shift segmentation becomes sensitive when we include the ac components and the energy term. Mean shift assigns a different sink, thereby causing severe oversegmentation, to any point that has different coefficients than its neighbors if the kernel size is not correct. For volume growth, this problems is partially alleviated, as the contributions of the coefficients are adjusted according to their variance.

In addition, we show that imposing the segmentation on the compact FT data structure constructed from the parsed video enables superior processing speeds as fast as 1.5 ms per frame, which makes the method a perfect preprocessing stage to provide initial segmentation masks for full-resolution uncompressed video segmentation.

Another contribution is the automatic estimation of the number of objects in the scene using a novel divergence score, which indicates how the consistency of the motion information changes after the merger of the selected volumes. We observed that the motion parameter-based similarity score exhibits robust and consistent responses while producing accurate clustering results unlike the trajectory-based similarity score, which is sensitive to the shape and rotational motion of objects.

REFERENCES

[1] P. Bouthemy, E. Francois, "Motion segmentation and qualitative dynamic scene analysis from an image sequence," *Int. J. Comput. Vision*, no. 10, pp. 157–187, 1993.

[2] T. Meier, K. N. Ngan, "Automatic segmentation of moving objects for video object plane generation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 525–538, Sep. 1998.

[3] F. Moscheni, S. Bhattacharjee, M. Kunt, "Spatiotemporal segmentation based on region merging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 9, pp. 897–915, 1998.

[4] G. Wu, T. Reed, "Image sequence processing using spatiotemporal segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 5, pp. 798–807, Aug. 1999.

[5] F. Porikli, Y. Wang, "Automatic video object segmentation using volume growing and hierarchical clustering," *J. Appl. Signal Process., (Spl. Issue on Object-Based Semantic Image Video Anal.)*, Jul. 2004.

[6] M. Ghanbari, "Cross search algorithm for motion estimation," *IEEE Trans. Commun.*, vol. 38, no. 7, pp. 950–953, Jul. 1990.

[7] L. M. Po, W. C. Ma, "A novel four step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[8] C. Stiller, "Object-based estimation of dense motion fields," *IEEE Trans. Image Process.*, vol. 6, no. 2, pp. 234–250, Feb. 1997.

[9] B. Horn, B. Schunck, "Determining optical flow," *Artificial Intell.*, vol. 17, pp. 185–204, 1981.

[10] B. D. Lucas, T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artificial Intell.*, Vancouver, BC, 1981, pp. 674–679.

[11] H. Wang, S. F. Chang, "Automatic face region detection in MPEG video sequences," in *Proc. SPIE Photonics, Electron Imaging Multimedia Syst.*, China, 1996.

[12] V. Kobla, D. Doermann, A. Rosenfeld, "Compressed domain video segmentation," CfAR Tech. Rep. CAR-TR-839, 1996.

[13] M. Mandal, S. Panchanathan, "Video segmentation in the wavelet domain," in *Proc. SPIE*, vol. 3527. 1998, pp. 262–270.

[14] J. Nang, S. Hong, Y. Ihm, "An efficient video segmentation scheme for MPEG video stream using macroblock information," in *Proc. 7th ACM Int. Conf. Multimedia*, 1999.

[15] R. V. Babu, K. R. Ramakrishnan, H. S. Srinivasan, "Video object segmentation: A compressed domain approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 4, pp. 462–474, Apr. 2004.

[16] R. V. Babu, K. R. Ramakrishnan, *Multimedia Tools Applicat.*, vol. 32, no. 1, pp. 93–113, Jan. 2007.

[17] V. Mezaris, I. Kompatsiaris, N. V. Boulgouris, M. G. Strintzis, "Real-time compressed-domain spatiotemporal segmentation and ontologies for video indexing and retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 5, pp. 606–621, May 2004.

[18] W. Zeng, J. Du, W. Gao, and Q. Huang, "Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model," in *Proc. Real-Time Imaging SPIE*, 2005, pp. 11–14.

[19] R. DeQueiroz, Z. Fan, T. Tran, "Optimizing block thresholding segmentation for multilayer compression of compound images," *IEEE Trans. Image Process.*, 2000.

[20] S. Ji, H. W. Park, "Image segmentation of color image based on region coherency," in *Proc. IEEE Trans. Image Process.*, vol. 1. Chicago, IL, 1998, pp. 80–83.

[21] F. Kossentini, Y. Lee, "Computation constrained fast MPEG-2 video coding," *IEEE Signal Process. Lett.*, vol. 4, no. 8, pp. 224–226, 1997.

[22] O. Sukmarg, K. Rao, "Fast algorithm to detect and segmentation in MPEG compressed domain," in *Proc. IEEE Region 10th Tech. Conf.*, Malaysia, 2000, pp. 364–368.

[23] R. Wang, H. J. Zhang, Y. Q. Zhang, "A confidence measure based moving object extraction system for compressed domain," in *Proc. IEEE Int. Symp. Circuits Syst.*, Switzerland, 2000, pp. 21–24.

[24] D. Comaniciu, P. Meer, "Mean-shift analysis and applications," in *Proc. IEEE Int. Conf. Comput. Vision*, 1999, pp. 1197–1203.

[25] D. Comaniciu, P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

Fatih Porikli (M'01–SM'04) graduated from the Bilkent University, Turkey, in 1992 and received the Ph.D. degree specializing in video object segmentation from the Polytechnic University, Brooklyn, NY.

He is currently a Senior Research Scientist and Project Manager at Mitsubishi Electric Research Labs, Cambridge, MA. Before joining there in 2000, he developed aerial image analysis applications at Hughes Research Labs, Malibu, CA, in 1999, and 3-D-stereoscopic systems at AT&T Research Labs, NJ, in 1997. His research focuses on computer vision, online learning and classification, robust optimization, multimedia processing, and video mining, with many commercial applications ranging from surveillance to medical to intelligent transportation systems. He has authored more than 80 technical publications and applied for over 50 patents. He is an Associate Editor for the *Springer Journal of Machine Vision Applications*, *EURASIP Journal of IVP* and the JOURNAL OF REAL-TIME IMAGING.

Dr. Porikli is a Senior Member of ACM and SPIE. He received the R&D 100 Scientist of the Year Award in 2006, the Best Paper Award at the IEEE International Conference on Computer Vision and Pattern Recognition, and the Most Popular Scientist of the Year Award, Turkey, in 2007. He is also recipient of the Superior Invention Award from MELCO in 2008.

Faisal Bashir received the B.S.E.E. degree from the University of Engineering and Technology, Lahore, Pakistan, and the Ph.D. degree in electrical and computer engineering from the University of Illinois, Chicago (UIC), in 2000 and 2006, respectively.

From 2000 to 2001, he was with Delta Indus Systems, a machine vision software development company specializing in height measurement through structured light patterns. From 2001 to 2005, he was a Research Assistant with the Multimedia Systems Lab, UIC. He was a Computer Vision Consultant at Mitsubishi Electric Research Labs, Cambridge, MA, in 2006. He is now a Senior Scientist at Retica Systems, Waltham, MA. His research interests include content-based multimedia indexing and retrieval, computer vision, machine learning, and statistical signal processing.

Dr. Bashir was a recipient of the National Talent Scholarship from 1995 to 2000 and the Provost Award for Graduate Research in Spring 2005.

Huifang Sun graduated from Harbin Engineering Institute, China, and received the Ph.D. degree from University of Ottawa, Canada.

He joined the Electrical Engineering Department of Fairleigh Dickinson University, Vancouver, Canada, in 1986 and was promoted to an Associate Professor before moving to the Sarnoff Corporation in 1990. He joined Sarnoff Lab as a Member of Technical Staff and was promoted to Technology Leader of Digital Video Communication later. In 1995, he joined Mitsubishi Electric Research Labs, Cambridge, MA, as Senior Principal Technical Staff and was promoted to Vice President and Fellow in 2003. His research interests include digital video/image compression and digital communication. He has coauthored two books and more than 150 journal and conference papers. He holds 48 U.S. patents. He received Technical Achievement Award in 1994 at Sarnoff Lab.

Dr. Sun is now an Associate Editor for IEEE TRANSACTION ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and was the Chair of Visual Processing Technical Committee of IEEE Circuits and System Society. He received the Best Paper Award of the 1992 IEEE Transaction on Consumer Electronics, the Best Paper Award of the 1996 ICCE, and the Best Paper Award of the 2003 IEEE Transaction on CSVT.