EURASIP Journal on Applied Signal Processing 2004:6, 814–832

Automatic Video Object Segmentation Using Volume Growing and Hierarchical Clustering

Fatih Porikli
Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA
Email: fatih@merl.com

Yao Wang
Department of Electrical Engineering, Polytechnic University, Brooklyn, NY 11201, USA
Email: yao@vision.poly.edu

Received 4 February 2003; Revised 26 December 2003

We introduce an automatic segmentation framework that blends the advantages of color-, texture-, shape-, and motion-based segmentation methods in a computationally feasible way. A spatiotemporal data structure is first constructed for each group of video frames, in which each pixel is assigned a feature vector based on low-level visual information. Then, the smallest homogeneous components, so-called volumes, are expanded from selected marker points using an adaptive, three-dimensional, centroid-linkage method. Self descriptors that characterize each volume and relational descriptors that capture the mutual properties between pairs of volumes are determined by evaluating the boundary, trajectory, and motion of the volumes. These descriptors are used to measure the similarity between volumes based on which volumes are further grouped into objects. A fine-to-coarse clustering algorithm yields a multiresolution object tree representation as an output of the segmentation.

Keywords and phrases: video segmentation, object detection, centroid linkage, color similarity.

1. INTRODUCTION

Object segmentation is important for video compression standards as well as recognition, event analysis, understanding, and video manipulation. By object we refer to a collection of image regions grouped under some homogeneity criteria where a region is defined as a contiguous set of pixels.

Basically, segmentation techniques can be grouped into three classes: region-based methods using a homogeneous color or texture criterion, motion-based approaches utilizing a homogeneous motion criterion, and object tracking. Approaches in the region-oriented domain range from empirical evaluation of various color spaces [1], to clustering in feature space [2], to nearest-neighbor algorithm, to pyramid linking [3], to morphological methods [4], to split-and-merge [5], to hierarchical clustering [6]. Color-clustering-based methods often utilize histograms and they are computationally simple. Histogram analysis delivers satisfactory segmentation result especially for multimodal color distributions, and where the input data set is relatively simple, clean, and fits the model well. However, this method lacks generality and robustness. Besides, histogram methods fail to establish spatial connectivity. Region-growing-based techniques provide better performance in terms of spatial connectivity and boundary accuracy than histogram-based methods. However, extracted regions may not correspond to actual physical objects unless the intensity or color of each pixel in objects differs from the background. A common problem of histogram and region-based methods arises from the fact that a video object can contain several totally different colors.

On the other hand, works in the motion-oriented domain start with an assumption that a semantic video object has a coherent motion that can be modeled by the same set of motion parameters. This type of motion segmentation works can be separated into two broad classes: boundary-placement schemes [7] and region-extraction schemes [8, 9, 10, 11, 12]. Most of these techniques are based on rough optical flow estimation or unreliable spatiotemporal segmentation, and may suffer from the inaccuracy of motion boundaries. The estimation of dense motion field tends to be extremely slow, hence not suitable for processing of large volumes of video and real-time data. Blockwise or higher-order motion models may be used instead of dense motion fields. However, a chicken-egg problem exists in modeling motion: should the region where a motion model is to be fitted be determined first, or should the motion field to be used to obtain the region be calculated first? Stochastic methods may overcome this priority problem by simultaneously modeling flow field

and spatial connectivity, but they require that the number of objects be supplied as a priori information before the segmentation. Small and nonrigid motion gives rise to additional model fitting difficulties. Furthermore, modeling may fail when a semantic video object has different motions in different parts of the object. Briefly, computational complexity, region-motion priority, and modeling issues are to be considered in utilizing dense motion fields for segmentation.

The last class is "tracking" [13]. A tracking process can be interpreted as the search for a target. It is the trajectories of the dynamic parameters that are linked in a time. This process is usually embodied through model matching. Many types of features, for example, points [14], intensity edges [15], textures [16], and regions [17] can be utilized for tracking. Three main approaches have been developed to track objects depending on their type: whether they are rigid, nonrigid, or have no regular shape. For the first two approaches, the goal is to compute the correspondences between objects already tracked and the newly detected moving regions, whereas the goal of the last approach is handling the situations where correspondences are ambiguous. The major difficulty in tracking is to deal with the interframe changes of moving objects. It is clear that the image shape of a moving object may undergo deformation, since a new aspect of the object may become visible or an actual shape of an object may change. Thus a model needs to evolve from one frame to the next, capturing the changes in the image shape of an object as it moves. Although for most of the cases, more than two video frames are already available before segmentation, existing techniques usually view tracking as a unidirectional propagation problem.

Semiautomatic segmentation methods have the power of correlating semantic information with extracted regions using human assistance. However, such assistance often obligates training of users to understand the behaviour of the segmentation method. Besides, real-time video systems require user-independent processing tools. The vast amount of video data demands for automatic segmentation since entering object boundaries by hand is cumbersome.

In summary, a single homogeneous color or motion criterion does not lead to satisfactory extraction of object information because each homogeneous criterion can only deal with a limited set of scenarios, and a video object may contain multiple colors and complex motions.

2. PROPOSED SEGMENTATION FRAMEWORK

Each of the segmentation algorithms summarized before has its own advantages. It would be desirable to have a general segmentation framework that combines distinct qualities of separate methods without getting hampered into their pitfalls. Such a system is expected to be made up by compatible processing modules that can be easily modified with respect to the application parameters. Even user-assistance and system-specific a priori information should be easily embedded into the segmentation framework without reconstructing the overall system architecture. Thus, we designed our segmentation framework to meet the following targets:

(i) automaticity,
(ii) adaptability,
(iii) accuracy,
(iv) computational complexity.

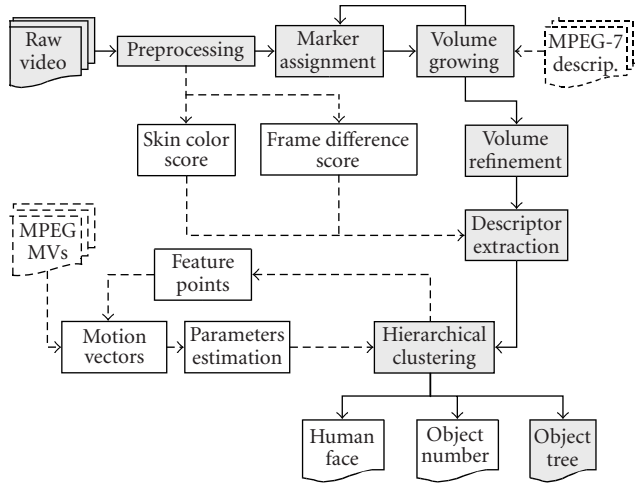


Figure 1: Flow diagram of the video segmentation algorithm showing all the major modular stages.

A general flow diagram of the framework is given in Figure 1. In the diagram, the main algorithm is shown in gray, and its modular extensions that include application-specific modules, that is, skin color detection, frame difference, and motion vector processing, are shown by the dashed lines. When MPEG-7 dominant color descriptors are available, they can be utilized in the volume-growing stage to adapt the color similarity function parameters. Frame difference score becomes useful where the camera system is stationary. Skin color can be incorporated as an additional feature for human detection. For MPEG encoded sequences, motion vectors can be used at the hierarchical clustering stage.

Before segmentation, the input video sequence is sliced into video shots that are defined as groups of consecutive frames having similar attributes between two scene cuts. The segmentation algorithm takes a certain number of consecutive frames within the same video shot, and processes all of these frames at the same time. The number of frames chosen can be the same as the length of the corresponding shot, or a number that is sufficient to have discriminatory object motion within the chosen frames. A limiting factor may be the memory requirement due to the large data size. After filtering, a spatiotemporal data structure is formed by computing pointwise features of frames. These features include color values, frame difference score, skin colors, and so forth as illustrated in Figure 2.

We acquire homogeneous parts of the spatiotemporal data by growing volumes around selected marker points. By volume growing, all the frames of an input video shot

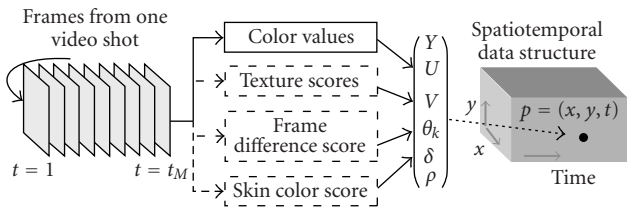


Figure 2: Construction of spatiotemporal data from the video.

Figure 3: Original and filtered images using the simplification filter.

are segmented simultaneously. Such an approach solves the problem of tracking objects and correlating the segmented regions between the consecutive frames since no account of the quantitative information about the regions and boundaries need to be kept. Volume-growing approach solves the problem of "should the region of support be obtained first by color segmentation followed by motion estimation, or should the motion field be obtained first followed by segmentation based on motion consistency?" by supplying the region of support and an initial estimation of motion at the same time. In addition, volume growing is computationally simple.

The grown volumes are refined to remove small and erroneous volumes. Then, motion trajectories of individual volumes are determined. Thus, without explicit motion estimation, a functional approximation of motion is obtained. Self descriptors for each volume and mutual descriptors for a pair of volumes are computed from volume trajectories and also from other volume statistics. These volumewise descriptors are designed to capture motion, shape, color, and other characteristics of the grown volumes. At this stage, we have the smallest homogeneous parts of a video shot and their relations in terms of mutual descriptors. Application-specific information can be incorporated as separate descriptors such as skin color.

In a following clustering stage, volumes are merged into objects by evaluating their descriptors. An iterative, hierarchical fine-to-coarse clustering is carried out until the motion similarity of merged objects becomes small. After clustering, an object partition tree that gives the video object planes for successively smaller number of objects is generated. The object partition tree can be appended to the input video for further recognition, data mining, and event analysis purposes. Note that this framework does not claim to obtain semantic information automatically, but it aims to provide tools for efficient extraction and integration of explicit visual features to improve the object detection. Thus, a user can easily change the visual definition of semantic object at the clustering stage, which has an insignificant computational load, without segmenting the video over again.

3. FORMATION OF SPATIOTEMPORAL DATA

3.1. Filtering

In the preprocessing stage, the input frames are filtered first. Two main objectives of filtering are noise removal and simplification of color components. Noisy or highly textured frames can cause oversegmentation by producing excessive number of segments. This not only slows down the algorithm, but also increases the memory requirements and degrades the stability of the segmentation. However, most noise filtering techniques demand intensive operations. Thus, we have developed a computationally efficient simplification filter which can retain the edge structure, and yet smooth the texture between edges. Simply stated, color value of a point is compared with its neighbors for each color channel. If the distance is less than a threshold, the point's color value is updated by the average of its neighbors within a local window. For the performance comparison of this filter with other methods including Gaussian, median, morphological filtering, and so forth, see [18]. A sample filtering result is given in Figure 3.

3.2. Quantization and color space

To further simplify input images, color quantization is applied by estimating a certain number of dominant colors. Quantization also decreases the total processing time by allowing use of smaller data structures in the implementation of the code. The dominant colors are determined by a hierarchical clustering approach incorporating the generalized Lloyd algorithm (GLA) at each level. Suppose we already have an optimal partitioning of all color vectors in the input image into 2^k level. At the $(k+1)$th level, we perturb each cluster center into two vectors, and use the resulting 2^{k+1} cluster centers as the initial cluster centers at this level. We then run the GLA to obtain an optimal partition with 2^{k+1} levels. Specifically, starting with the initial cluster centers, we group each input color vector to its closest cluster center. The cluster centers are then updated based on the new grouping. A distortion score is calculated which is the sum of the distances of the color vectors to the cluster centers. The grouping and the recalculation of the cluster centers are repeated until the distortion does not reduce significantly anymore. Initially at level $k = 0$, we have one cluster only, including all the color vectors of the input image. As a final stage, the clusters that have close color centers are grouped to decide on a final number of dominant colors.

The complexity of the metric used for computing color distances is a major factor in selecting a color space since most of the processing time is spent while computing the color distances between the points. We preferred the YUV color space since the color distance can be computed using simpler norms. In addition, the YUV space separates illuminance from luminance components, and represents color

Figure 4: Quantization by 32, 16, and 8 dominant colors, which are shown next to each image. As visible, very low quantization levels may disturb the color properties, that is, skin colors and edges.

in more accordance with human perception than the *RGB* [19]. Thus, the segmentation results are visually more plausible. The above-described dominant colors have minor differences from the MPEG-7 dominant color descriptors. For example, MPEG-7 has a smaller number of color bins, and it is based on *Lab* color space. In the case where MPEG-7 descriptors are available with the input video, the dominant color descriptor can be directly used to quantize the input video after suitable conversion of the color space. In Figure 4, quantized images with different number of dominant colors are given.

3.3. Feature vectors

Frames of the input video shot are then assembled into a spatiotemporal data structure S. Each element of this data structure has a feature vector $\mathbf{w}(p) = [Y, U, V, \delta, \theta_1, \ldots, \theta_K, \rho]$. Here, $p = (x, y, t)$ is a point in S where (x, y) is the spatial coordinate and t is the frame number. We will denote individual attributes of the feature vector, for example, the Y color value of point p, by $Y(p)$. Sometimes we also use $w(p, k)$ to represent feature k at point p, for example, $k = Y, U, V$. Table 1 summarizes the notation. Besides the color values, additional attributes can be included in the feature vector. The frame difference score δ is defined as the pointwise color dissimilarity of two frames with respect to a given set of rules. One such rule is

$$\delta(p) = \left| Y(p) - Y(p_{t^-}) \right|, \tag{1}$$

where $p_{t^-} = (x, y, t-1)$. The texture features $\theta_1, \ldots, \theta_K$ are computed by convolving the luminance channel Y with the Gabor filter kernels as

$$\theta_k(p) = \left| Y(p) \otimes \frac{1}{2\pi\sigma^2} e^{-((x^2+y^2)/2\pi\sigma^2)e^{-2\pi(u_k+v_k)}} \right|. \tag{2}$$

It is sufficient to employ the values for the spatial frequency $\sqrt{u^2+v^2} = 2, 4, 8$ and the direction $\tan^{-1}(u/v) = 0, \pi/4, \pi/2, 3\pi/4$, which leads to a total of 12 texture features. Obtaining texture features is computationally as intensive as estimating motion vectors by phase correlation due to the convolution process. Blending texture and color components into a single similarity measure is usually done by assigning weighting parameters [20]. In this work, we concentrate on the color components.

The skin color score ρ indicates whether a point has high likelihood of corresponding to human skin. We obtained a mapping from the color space to the skin color values by projecting the color values of a large set of manually segmented skin images that include people of various races, genders, and ages. This mapping is used as a lookup table to determine the skin color score. More details on this derivation can be found in [21]. In Figure 5, skin color scores of sample images are shown. In these images, higher intensity values correspond to higher likelihoods.

Table 1: Notation of parameters.

S	Volumetric spatiotemporal data
p	Point in S; $p = (x, y, t)$
$\mathbf{w}(p)$	Feature vector at p
$Y(p), U(p), V(p)$	Color values at p
$\delta(p)$	Frame difference at p
$\theta_k(p)$	Texture features at p
$\rho(p)$	Skin color score at p
$\nabla Y, \nabla U, \nabla V$	Color gradient
m_i	Marker of volume V_i
$\mathbf{c}_i$	Feature vector of volume V_i
V_i	A volume within S
$\gamma(i)$	Self descriptor of volume V_i
$\Gamma(i, j)$	Relational descriptor of pair V_i, V_j

4. VOLUME GROWING

Volumes are the smallest connected components of the spatiotemporal data S with homogeneous color and texture distribution within each volume. Using markers and evaluating various distance criteria, volumes are grown iteratively by grouping neighboring points of similar characteristics.

In principle, volume-growing methods are applicable whenever a distance measure and a linkage strategy can be defined. Several linkage methods were developed in the literature; they differ in the spatial relation of the points for which the distance measure is computed. In single-linkage volume growing, a point is joined to its 3D neighboring points whose properties are similar enough. In hybrid-linkage growing, similarity among the points is established based on the properties within a local neighborhood of the point itself instead of using the immediate neighbors. In the case of centroid-linkage volume growing, a point is joined to a volume by evaluating the distance between the centroid of the volume and the current point. Yet another approach is to provide not only a point that is in the desired volume but also counterexamples that are not in the volume. Two-dimensional versions of these linkage algorithms are explained in [22]. In the following, we first describe the marker selection process, and then the centroid-linkage algorithm in more detail.

4.1. Marker assignment

A marker is the seed of a volume around it. Since a volume's initial properties will be determined by its marker, a marker should be a good representative of its local neighborhood. A

Figure 5: Skin color scores ρ of sample images.

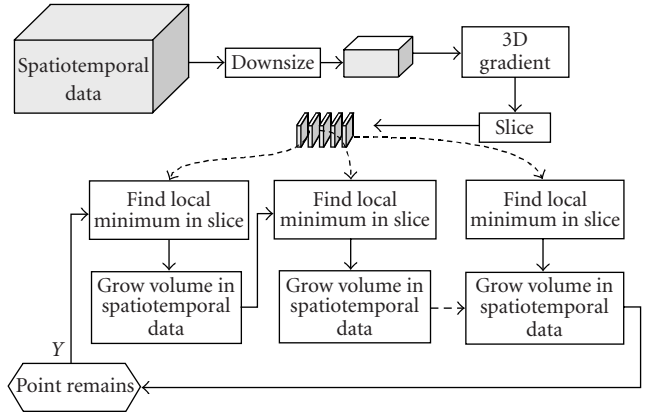


Figure 6: Fast marker selection finds the minimum gradient magnitude points in the current slice of the downsampled data. Then, a volume is grown within the spatiotemporal data, and the process is repeated until no point remains as unclassified.

point that has a low color gradient magnitude satisfies this criterion. Let m_i be a marker for volume V_i, and Q the set of all available points, that is, it is all the points of S initially. The color gradient magnitude is defined as follows:

$$|\nabla S(p)| = |\nabla Y(p)| + |\nabla U(p)| + |\nabla V(p)| \quad (3)$$

such that the gradient magnitude of a channel is

$$|\nabla Y(p)| = |Y(p_{x^+}) - Y(p_{x^-})| + |Y(p_{y^+}) - Y(p_{y^-})| + |Y(p_{t^+}) - Y(p_{t^-})|, \quad (4)$$

where p_{x^+} and p_{x^-} represent equal distances on the x-direction from the center point p, that is, $(x-1, y, t)$, $(x+1, y, t)$, and so forth. We observed that using L_2 norm instead of L_1 norm does not improve the results. The point having local minimum gradient magnitude is chosen as marker. A volume V_i is grown as will be explained in the following section, and all the points of the volume are removed from the set Q. The next minimum in the remaining set is chosen, and the selection process is repeated until no more available points remain in S. Rather than searching the full-resolution spatiotemporal data, a subsampled version of it is used to find the minima since searching in full resolution is computationally costly.

More computational reduction is achieved by dividing subsampled S into slices. A minimum gradient magnitude point is found for the first slice, and a volume is grown, then the next minimum is searched in the next slice as illustrated in Figure 6. The temporal continuity is preserved by growing a volume in the whole spatiotemporal data S after selecting a marker in the current slice. In case the markers are limited only within the first frame, the algorithm becomes a forward volume growing.

Generally, the marker points are uniformly distributed among the frames of a video shot in which objects are consistent and motion is uniform. For such video shots, a single frame of S can be used for selection of all markers instead of using the whole S. However, the presence of fast moving small objects, highly textured objects, and illumination changes may deteriorate the segmentation performance if a single frame is used. Besides, objects that are not visible in the single frame may not be detected at all. The iterative slice approach overcomes these difficulties.

4.2. Centroid-linkage algorithm

For each new volume V_i, a volume feature vector $\mathbf{c}_i$, the so called "centroid," is assigned. Centroid-linkage algorithm compares the features of a candidate point to the current volume's feature vector. This vector is composed of the color statistics of the volume, and initially it is equal to the feature vector of the point chosen as marker $\mathbf{c}_i(k) = \mathbf{w}(m_i, k)$. In a 6-point neighborhood, two in each of the x, y, t direction, the color distances of the adjoint points are calculated. If the distance $d(\mathbf{c}_i, \mathbf{w}(q))$ is less than a volume-specific threshold ϵ_i, the point q is included in the volume, and the centroid vector is updated as

$$\mathbf{c}_i^n(k) = \frac{1}{N}\left[(N-1)\mathbf{c}_i^{n-1}(k) + \mathbf{w}(q,k)\right], \quad (5)$$

where N is the number of points in the volume after the inclusion of q. If the point q has a neighbor that is not included in the current volume, it is assigned as an "active-shell" point. Thus, active-shell points constitute the boundary of the volume. In the next cycle, the unclassified neighbors of the active-shell points are probed. Linkage is repeated if either no point remains in the active shell or in the spatiotemporal data.

There are two other possible linkage techniques: single-linkage, which compares a point with only its immediate neighbors, and dual-linkage, which compares with the current object boundary. We observed that these two techniques are prone to segmentation errors such as leakage and color inconsistent segments. The sample results for the various linkage algorithms are given in Figure 7.

(a) (b) (c)

Figure 7: Segmentation by (a) single linkage, (b) dual linkage, and (c) centroid linkage. Single linkage is prone to errors.

4.3. Distance calculation and threshold determination

The aim of the linkage algorithm is to generate homogeneous volumes. Here we define homogeneity as the quality of being uniform in color composition. In other words, it is the amount of color variation. For a moment, let us assume a color density function of the data is available. Modality of this density function refers to the number of its principal components, that is, the number of separate models for a mixture of models representation. A high modality indicates larger number of distinct color clusters of the density function. Our key hypothesis is that points of a color homogeneous volume are more likely to be in the same color cluster rather than being in different color clusters. Thus, we can establish a relationship between the number of clusters and the homogeneity specifications of volumes. If we know the color cluster that a volume corresponds to, we can determine the specifications of homogeneity for that volume, that is, parameters of the color distance function and its threshold.

Before volume growing, we approximate the color density function by deriving a 3D color histogram of the slice. We find cluster centers within the color space either by assigning the dominant colors as centers or using the described GLA clustering algorithm. We group each color vector $\mathbf{w}(p)$ to the closest cluster center, and for each cluster we compute a within-cluster distance variance σ^2.

After choosing a marker and initializing a volume feature vector $\mathbf{c}_i$, we determine the closest cluster center to the $\mathbf{c}_i$ in the color space. Using the variance of this cluster, we define the color distance and its threshold as follows:

$$d(\mathbf{c}_i, q) = \sqrt{\sum_k \left(\mathbf{c}_i(k) - w(q,k)\right)^2}, \tag{6}$$

where $k : Y, U, V$ and the threshold is $\epsilon_i = 2.5\sigma$ to let the inclusion of the 95% of colors within the same color cluster. The above-formulation assumes that the color channels are equally contributing (due to the Euclidean distance norm), and the 3D color histogram is densely populated (for effective application of clustering). However, a dense histogram may not be available in case of small slice sizes, and color components may not be equally important in case of the YUV space.

We also developed an alternative approach that uses separate 1D histograms. Local maxima $h_n(k)$ of the histograms are obtained for each channel such that $h_n(k) < h_{n+1}(k)$ and $n = 1, \ldots, H_k$. Note that number of maxima H_k for different channels may be different. Histograms are clustered, and within-cluster distance variance is computed for each cluster similarly. Using the current marker point m_i, three coefficients $\tau_i(k)$, $k : Y, U, V$ (one for each histogram) are determined as

$$\begin{aligned} \tau_i(k) &= 2.5\sigma_j(k), \\ j &= \arg\min_n \left|\, \mathbf{c}_i(k) - h_n(k) \,\right|, \end{aligned} \tag{7}$$

where $h_n(k)$ is the closest center. These coefficients specify the cluster ranges. A logarithmic distance function is formulated as follows:

$$d(\mathbf{c}_i, q) = \sum_k H_k \log_2 \left(1 + \frac{\left|\, \mathbf{c}_i(k) - w(q,k) \,\right|}{\tau_i(k)}\right). \tag{8}$$

We normalized the channel differences with the cluster ranges to equalize the contribution of a wide cluster in a histogram to a narrow cluster in another histogram. The logarithmic term intended to suppress the large color mismatches of a single histogram. Considering that a channel that has more distinctive colors should provide more information for segmentation, the channel distances are weighted by the corresponding H_k's. Then, the distance threshold for volume V_i is derived as

$$\epsilon_i = \sum_k H_k. \tag{9}$$

4.4. Modes of volume growing

Volume growing can be carried out either by growing multiple volumes simultaneously, or expanding only one volume at a time. Furthermore, the expansion itself can be done either in an intraframe-interframe switching fashion, or a recursive outward growing style.

(i) Simultaneous growing. After certain number of marker points are determined, volumes are grown simultaneously from each marker. At a growing cycle, all the existing volumes are updated by examining the neighboring points to the active shell of the current volume. In case a volume stops growing, an additional marker that is an adjoint point to the boundary of the stopped volume is selected. Although simultaneous growing is fast, it may divide homogeneous volumes into multiple smaller volumes, thus volume merging becomes is necessary.

(ii) One-at-a-time growing. At each cycle, only a single marker point is chosen, and a volume is grown around this marker. After the volume stops growing, another marker in the remaining portion of the spatiotemporal data is selected. This process continues until no more point remains in S. An advantage of one-at-a-time growing is that it can be implemented by recursive programming. It also generates more homogeneous volumes. However, it demands more memory to keep all the pointers.

(iii) Recursive diffusion. The neighboring points to the active shell are evaluated disregarding whether they are in the same frame with the active shell point or not as illustrated in Figure 8. After a point is included within a volume, the point becomes a point of the active shell as long as it has a neighbor that is not included in the same volume. By updating the active shell as described, the volume is diffused outward from the marker. Instead of using only adjoint points, other points within a local window around the active shell point can be used in diffusion as well. However, in this case the computational complexity increases, and moreover, connectivity may deteriorate.

(iv) Intraframe-interframe switching. A volume grown using recursive diffusion tends to be topologically noncompact by having several holes and ridges within. Such a volume usually generate unconnected regions when it is sliced framewise. In intraframe-interframe switching, the diffusion mechanism is first applied within the same frame to grow a region, then results are propagated to the previous and next frames. The grown region is assigned as the active shell for the neighboring frames. As a result, each framewise projection of a volume will be a single connected region, and volumes will have more compact shapes.

4.5. Volume refinement

After volume growing, some of the volumes may be negligible in size or very elongated due to the fine texture and edges. Such volumes increase the computational load of the later processing. A simple way of removing a small or elongated volume is labeling its points as unclassified and inflating the remaining volumes iteratively to fill up the empty space. First, the unclassified points that are adjoint to other volumes are put into a set of active shell. Then, each active shell point is included in the volume which is adjoint and has the minimum color distance. The point is removed from the active shell, and the inclusion process is iterated until no more unclassified point remains. Alternatively, a small volume can be merged into one of its neighbors as a whole using volumewise similarity. In this case, similarity is defined as a combination of the ratio of the mutual surface, compactness ratio, and color distance. For more details on definition of such a similarity measure, see [21].

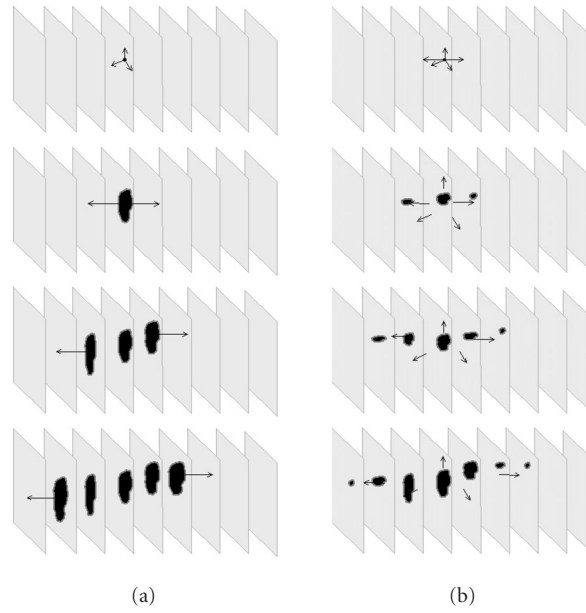


Figure 8: (a) Volume growing by intraframe-interframe switching. (b) Recursive diffusion. As visible, recursive diffusion grows volumes as an inflating balloon, whereas switching method first enlarges a region in a frame then spreads this region to the adjoint frames.

5. DESCRIPTORS OF VOLUMES

Descriptors capture various aspects of the volumes such as motion, shape, and color characteristics of individual volumes, as well as pairwise relations among the volumes.

5.1. Self descriptors

Self descriptors evaluate a volume's properties such as its size $\gamma_{si}(i)$, its total boundary $\gamma_{bo}(i)$, its normalized color histogram $\gamma_h(i)$ $(0 \leq \gamma_h(i) \leq 1)$, and the number of frames $\gamma_{ex}(i)$ that the volume extends in the spatiotemporal data. Compactness $\gamma_{co}(i)$ is defined as

$$\gamma_{co}(i) = \frac{1}{\gamma_{ex}(i)} \sum_t \frac{\gamma_{si}(i,t)}{\gamma_{bo}(i,t)^2}, \tag{10}$$

where the framewise boundary $\gamma_{bo}(i,t)$ is squared to make compactness score independent from the radius of the framewise region $\gamma_{si}(i,t)$ at frame t. (Consider the case of

a disk; $\gamma_{co} = \pi r^2/(2\pi r)^2 = 1/(4\pi)$.) Note that, in the spatiotemporal data, the most compact volume is a cylinder along the time axis, but not a sphere. Elongated, sharp-pointed, shell-like, and thin shapes have lower compactness scores. However, the compactness score is sensitive to the boundary irregularities.

Motion trajectory of a volume is defined as the localization of its framewise representative points. The representative point can be chosen as the center of mass, or it can be the intersection of the longest line within the volumes frame projection and another line that is longest in the perpendicular direction. We used the center of mass since it can be computed easily. Trajectory $\mathbf{T}(i,t) = [T_i^x(t), T_i^y(t)]^T$ is calculated by computing the framewise averages of volume's coordinates along x and y directions. Sample trajectories are shown in Figure 9. Note that, these trajectories do not involve any motion estimation. The trajectory approximates the translational motion in most of the cases. The translational motion is the easiest to be perceived by the human visual system, for much the same reason it is the most discriminative in object recognition. Motion trajectory enables to comprehend the motion of a volume between frames without requiring complex motion vector computation. It can also be used to initialize parameterized motion estimation to improve the accuracy and to accelerate the speed.

The descriptor $\gamma_{tl}(i)$ measures the length of the trajectory. Volumes that are stationary with respect to the camera imaging plane have shorter trajectory lengths. The set of affine motion parameters $A(i,t) = [a_1(i,t), \ldots, a_6(i,t)]$ for a volume models the framewise motion

$$\mathbf{v}(p) = \begin{bmatrix} a_1(i,t) & a_2(i,t) \\ a_4(i,t) & a_5(i,t) \end{bmatrix} p + \begin{bmatrix} a_3(i,t) \\ a_6(i,t) \end{bmatrix} - p, \tag{11}$$

where $\mathbf{v}(p)$ are motion vectors at p. To estimate these parameters, a certain number of feature points p_f are selected for each region $R_i(t)$, and corresponding motion vectors are computed. Feature points are selected among the high spatial energy points. The spatial energy of a point is defined in terms of color variance as

$$\mathbf{w}(p,e) = \sum_p \sum_k \left(\mathbf{w}(p,k) - \mathbf{w}(p,\mu_k)\right)^2. \tag{12}$$

Above, $\mathbf{w}(p,\mu_k)$ is the color mean of points in a small local window centered around p. After $\mathbf{w}(p,e)$'s are computed, the points of $R_i(t)$ are ordered with respect to their spatial energy magnitudes. The highest rank point on the list is assigned as a feature point p_f, and neighboring points of p_f are removed from the list. Then, the next highest rank point is chosen until a certain number of points are selected. To estimate the motion vectors, we used phase correlation in which the search range is constrained around the trajectory $\mathbf{T}(i,t)$. Given motion vectors $\hat{\mathbf{v}}(p_f)$, the affine model is fitted by minimizing

$$A(i,t) = \arg\min \sum_{p_f} \log\left(1 + \left|\mathbf{v}(p_f) - \hat{\mathbf{v}}(p_f)\right|\right), \tag{13}$$

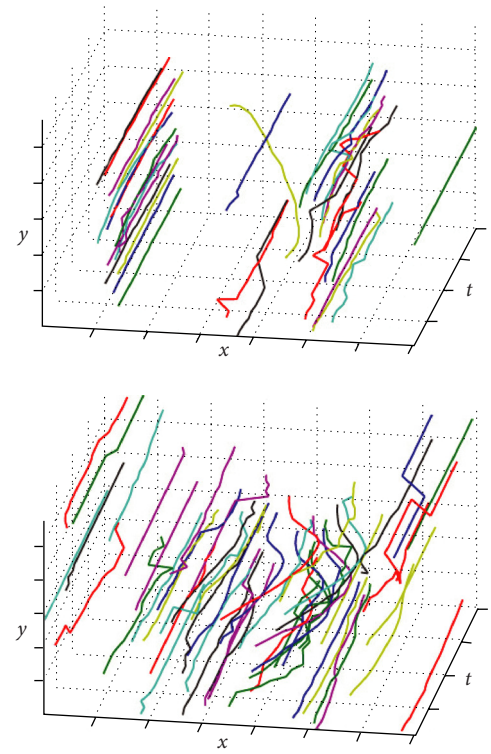


Figure 9: Sample trajectories of *Children* and *Foreman*.

where $\mathbf{v}(p_f)$ are the affine projected motion vectors as given in (11) and $\hat{\mathbf{v}}(p_f)$ are the motion vectors estimated by phase-correlation at feature points p_f. The logarithm term works as a robust estimator which can detect and reject the measurement outliers that violate the motion model. We used downhill simplex method for minimization. To reduce the load of the above computationally intensive motion vector and parameter estimation procedures, we only used up to 20 points to estimate the parameters. Note that the motion parameters are estimated for only a small number of volumes, which is usually between 10 and 100, after the volume refinement stage.

The frame difference descriptor $\gamma_\delta(i)$ is proportional to the amount of color change in the volume after trajectory motion compensation:

$$\gamma_\delta(i) = \frac{1}{\gamma_{si}(i)} \sum_{p \in V_i} \delta\left(x - T_i^x(t), y - T_i^y(t), t\right), \tag{14}$$

where the frame difference score δ is given as in (1). We present truncated frame difference scores in Figure 10. The skin color descriptor $\gamma_\rho(i)$ is computed similarly

$$\gamma_\rho(i) = \frac{1}{\gamma_{si}(i)} \sum_{p \in V_i} \rho(p), \tag{15}$$

where $\rho(p)$ is the skin color score as explained in Section 3.3 and $\gamma_{si}(i)$ is the size of the volume.

5.2. Relational descriptors

These descriptors evaluate correlation between a pair of volumes V_i and V_j. The mutual trajectory distance $\Delta(i, j, t)$ is one of the motion-based relative descriptors. It is calculated by

$$\Delta(i, j, t) = \left| \mathbf{T}(i, t) - \mathbf{T}(j, t) \right|. \tag{16}$$

The mean of the trajectory distance $\Gamma_\mu(i, j)$ measures average distance between the trajectories, and $\Gamma_\sigma(i, j)$ is the variance of the distance $\Delta(i, j, t)$. A small variance means two volumes have similar translational motion, and a big variance reveals volumes having different motion, that is, getting away from each other or moving in the opposite directions, etc. One exception happens in case of a large background, since its trajectory usually falls on the center of the frames. To distinguish volumes that have small motion variances but opposite motion directions, for example, two volumes turning around a mutual axis, the directional difference $\Gamma_{dd}(i, j)$ can also be defined. The parameterized motion similarity is measured by $\Gamma_{pm}(i, j)$:

$$\Gamma_{pm}(i, j) = \sum_t \left[c_R \sum_{n=1,2,4,5} \left| a_n(i, t) a_n(j, t) \right| + c_T \sum_{n=3,6} \left| a_n(i, t) - a_n(j, t) \right| \right], \tag{17}$$

where the constants are set as $c_T \gg c_R$ to take into account of the fact that a small change in the parameters a_n, $n = 1, 2, 3, 4$, can lead to much larger difference in the modeled motion field than the translation parameters a_5, a_6. The compactness ratio $\Gamma_{cr}(i, j)$ of a pair of volumes is the amount of the change on the total compactness before and after the two volumes merge:

$$\Gamma_{cr}(i, j) = \frac{\gamma_{co}(V_i \cup V_j)}{\gamma_{co}(i) + \gamma_{co}(j)}, \tag{18}$$

where a small $\Gamma_{cr}(i, j)$ means the merging of V_i and V_j will generate a less compact volume. Another shape-related descriptor $\Gamma_{br}(i, j)$ is the ratio of mutual boundary of two volumes V_i and V_j to the boundary of volume V_i. The color difference descriptor $\Gamma_{cd}(i, j)$ gives the sum of the difference between the color histograms, the mutual existence $\Gamma_{ex}(i, j)$ counts the number of frames in which both volumes exist, and $\Gamma_{ne}(i, j)$ shows whether volumes are adjoint. Similarly, $\Gamma_\rho(i, j)$ shows the difference in the skin color scores between the volumes, and $\Gamma_{fd}(i, j)$ gives the difference in the change detection scores.

6. FINE-TO-COARSE CLUSTERING

As described in the general framework, the volumes are clustered into objects using their descriptors. 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. Grouping volumes using adaptive k-means method in a coarse-to-fine manner is an example of the partitional approaches as illustrated in Figure 11.

Figure 10: Frame difference score $\delta(p)$ for *Foreman*, *Akiyo*, and *Head*. Frame difference indicates the amount of motion for certain cases.

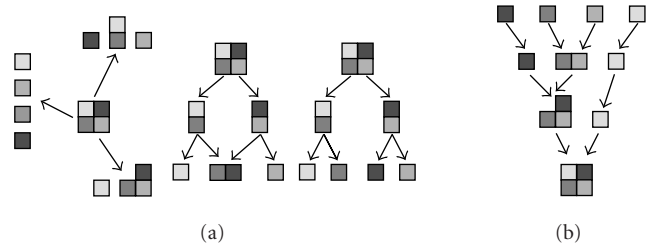


Figure 11: (a) Coarse-to-fine (k-means, GLA, quad tree) and (b) fine-to-coarse clustering. The first approach divides the volumes into certain number of clusters at each time, the second merges a pair of volumes at each level.

In the fine-to-coarse merging method, determination of most similar volumes is done iteratively. At each iteration, all the possible volume combinations are evaluated. The pair having the highest similarity score are merged and affected descriptors are updated. A similar morphological image segmentation approach using such hierarchical clustering is presented in [6].

Detection of a semantic object requires explicit knowledge of specific object characteristics. Therefore, user has to decide which criteria dictate the similarity of volumes. It is the semantic information that is being incorporated at this stage of the segmentation. We designed the segmentation framework such that most of the important object characteristics will be available for user in terms of the self and relational descriptors. Other characteristics can be included easily without changing the overall architecture. Furthermore, the computational load of building objects from the volumes is minimized significantly by transferring the descriptor extraction in the previous automatic stages.

The following observations are made on the similarity of two volumes.

(1) Two volumes are similar if their motion is similar. In other words, volumes having similar motion construct the same object. A stationary region has high probability of being in the same object with another region

that is stationary, that is, a tree and a house in the same scene. We already measured the motion similarity of two volumes in terms of motion-based relational descriptors $\Gamma_\sigma(i,j)$, $\Gamma_{dd}(i,j)$, and $\Gamma_{pm}(i,j)$. These descriptors can be incorporated in the similarity definition. However, without using further intelligent models, it is not straightforward to distinguish objects with similar motion.

(2) Objects tend to be compact. A human face, a car, a flag, a soccer ball are all compact objects. For instance, a car in a surveillance video is formed by separate elongated smaller regions. Shape of a volume gives clues about its identity. We captured shape information in the descriptors $\Gamma_{cr}(i,j)$ and $\Gamma_{br}(i,j)$ and also volume boundary itself. Note that, compactness ratio must be used with caution in merging volumes. If a volume is enclosing another volume, their merge will increase compactness whether these two volumes correspond to same object or not. Furthermore, many objects such as cloud formations, walking people, and so forth are not compact. To improve the success of shape-based object clustering, application-specific criteria should be used, for example, a human model for videoconferencing.

(3) Objects have connected parts. This is obvious for most of the cases, an animal, a car, a plane, a human, and so forth, unless an object is visible only partially. We begin evaluation of similarity with the volumes that are neighbors to each other. Neighborhood constraint is useful, and yet, can easily deteriorate the segmentation accuracy in case of an under segmentation, that is, background encloses most of the volumes.

(4) An object moves as a whole. Although this statement is not always true for human objects, for rigid bodies, it is useful. The change detection descriptor becomes very useful in constructing objects that are moving in front of a stationary background.

(5) Each volume already has a consistent color by construction, therefore there is little room for utilization of color information to determine a neighbor to merge in. In fact, most objects are made from small volumes that have different colors, that is, human body constituents face, hair, dress, and so forth. When forming the similarity measure, color should not be a key factor. However, for specific video sequences featuring people, human skin color is an important factor.

(6) Important objects tend to be at the center. We can find good examples as in head-and-shoulder sequences, sports, and so forth.

To blend all the above observations and statements, we evaluate the likelihood of a volume merge given the relevant descriptors. For this purpose, we define a similarity score

$$P_*(V_{i,j}) \equiv \frac{\Gamma_*(i,j)}{\sum_{m,n}\Gamma_*(m,n)}. \tag{19}$$

Alternatively, $P_*(V_{i,j})$ can be defined using a ranking-based similarity measure. For all possible neighboring volume pairs, the relevant relative descriptors are ordered in separate lists in either descending or ascending order. For example, $L_\sigma(i,j)$ returns a number indicating the rank of the descriptor $\Gamma_\sigma(i,j)$ in its ordered list. Using the ranks in the corresponding lists, the likelihood is computed as

$$P_*(V_{i,j}) \equiv 1 - \frac{2L_*(i,j)}{l_*(l_*+1)}, \tag{20}$$

where the length of the list L_* is l_*. The similarity based on all descriptors is defined as

$$P(V_{i,j}) = \sum_{*:\sigma,dd,\ldots} \lambda_* P_*(V_{i,j}), \tag{21}$$

where constant multipliers λ's are used to normalize and adjust the contribution of each descriptor. These multipliers can be adapted to the specific applications as well. To detect human face, skin color descriptor $\Gamma_\rho(i,j)$ can be included in the above formula. Similarly, if we are interested in finding moving objects in a stationary camera setup but trajectory or parametric modeling are not sufficient enough to obtain an accurate motion representation, the frame difference descriptor $\gamma_\delta(i)$ becomes an adequate source.

The pair having the highest similarity score are merged, and the descriptors of the volumes are updated accordingly. Clustering is performed until there are only two volumes remaining. 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 similarity score of the selected merge. If this score gets small or shows a sudden drop, the merge is likely to be not a valid merge although it is the best available merge.

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 other decision mechanism for further analysis. We use an object tree structure to represent segmentation results as demonstrated in Figure 12. In this representation, the video is divided into objects, and objects into volumes. At the lowest volume level, the descriptors and boundaries are available. Volumes are homogeneous in color and texture, and they are connected within. The clustering step generates higher levels that are consistent in motion. The user can choose the segmentation result at different levels based on the desired level of details. In case a user wants to change the criteria used to cluster volumes, only the clustering stage needs to be executed with new criteria, for example, weights in different descriptors, which is computationally simple.

The corresponding objects at various object levels of the multiresolution object tree are presented in Figures 13 and 14. The descriptor multipliers are set as $\lambda_{fd} = \lambda_\rho = \lambda_{cr} = \lambda_{br} = 1$, $\lambda_{\text{others}} = 0$ for *Akiyo* since we intended to find a human head having very slow nonrigid motion, $\lambda_\mu = \lambda_{cr} = \lambda_{br} = 1$, $\lambda_{\text{others}} = 0$ for *Bream* since motion is the most

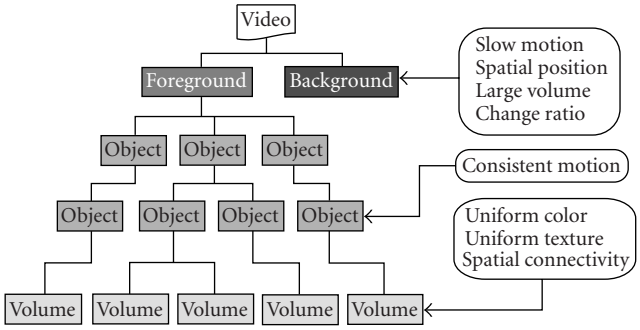


FIGURE 12: Multiresolution partition of objects in a hierarchical tree representation.

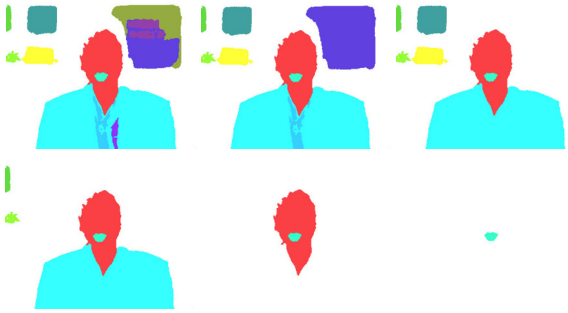


FIGURE 13: Results at object levels 13, 10, 8, 6, 3, and 2 for *Akiyo* using frame difference descriptor.

discriminating visual feature for the fish, and $\lambda_\mu = \lambda_\rho = \lambda_{cr} = \lambda_{br} = 1$, $\lambda_{\text{others}} = 0$ for *Children* since objects are defined as moving regions that have human skin colors. Hierarchical clustering finds the mouth of the speaker in *Akiyo* as the most different object since it has the highest frame difference and skin color score. At the consequent levels of the multiresolution tree, the face and suit comes because of the same reason. For *Children*, the red ball has the most discriminating motion among all the objects, and the proposed video object segmentation (VOS) method correctly put it on the top level of the multiresolution tree (Figure 14). As visible, volume growing accurately detects the objects boundaries as a result of adaptive color distance threshold assignment.

7. EXPERIMENTAL RESULTS

We selected a version of the proposed VOS framework to be used as a reference considering the computational simplicity, that is, texture features and motion parameters are omitted. Centroid linkage is used to grow volumes, and 1D histogram-based formulation (8) is applied to compute color distance. Intra-inter switching method is preferred to prevent a volume from having disconnected regions.

We also implemented two other state-of-art semiautomatic tracker to provide a detailed comparison of the proposed method with others.

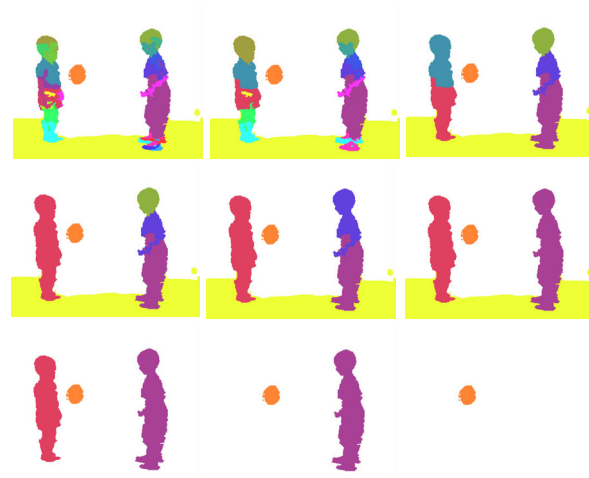


FIGURE 14: Results at object levels 12, 10, 8, 7, 6, 5, 4, 3, and 2 for sequence *Children* using trajectory distance variance descriptor.

7.1. Reference methods

Active MPEG-4 object segmentation (AMOS)

We used a semiautomatic video object segmentation algorithm [23, 24] to compare our results. This algorithm requires the initial object definition, that is, object boundary to be provided by users by mouse-selected points around the target object. Then a snake algorithm refines the user input to fit a smooth boundary. The initial object is generated through a region segmentation and aggregation process. To extract homogeneous regions in both color and motion, motion segmentation based on a dense motion field is used to further split the color regions. Homogeneous regions are classified as either foreground or background to form the object. Region aggregation is based on the coverage of each region by the initial object mask: regions that are covered more than a certain percentage are grouped into the foreground object. The final contour of the semantic object is computed from foreground regions. Tracking is done at both the region and object levels. Segmented regions from the previous frame are first projected to the current frame using their individual 2D affine models with 6 parameters. An expanded bounding box including all projected foreground regions is computed. Then the area inside the bounding box is split to homogeneous color and motion regions following a region tracking process. Pixels that cannot be tracked from any old regions are labeled as new regions. Thus the resulting homogeneous regions are tagged either foreground (meaning tracked from a foreground region), background (meaning tracked from a background region), or new (meaning not tracked). They are then passed to an aggregation process and classified as belonging to either the foreground object or the background. To handle possible motion estimation errors, the aggregation process is carried out iteratively. Finally, the object contour is computed from foreground regions.

This technique is very similar to the system explained in COST-211 project [25].

Self-affine mapping tracker (SAM)

We also made comparisons with another semiautomatic tracker [26] in which the initial boundary was entered by painting a region instead of mouse clicks. The concept of this method is quite different from that of the snake method. A self-affine mapping system instead of the energy minimization procedure is used to approach and fit the roughly drawn line to the object contour. The object contour is extracted as a self-similar curve instead of a smooth curve. The self-affine map's parameters are detected by analyzing the block-wise self-similarity of an image using a simplified algorithm in fractal encoding.

7.2. Performance measures

As explained in [27], comparative assessment of segmentation algorithms is often based upon subjective judgement, which is qualitative and time consuming. Although several measures can be applied in the presence of a ground-truth mask, the generation of ground truth requires significant effort and is often limited to foreground-background type of object segmentation. Selection of a conventional ground truth for multilevel object extraction algorithms may not be possible. For instance, what should be assigned as ground truth for 3-object level for *Children* sequence: two boys and the background, or one boy, ball and background, or some other possible combination? Should two boys constitute a single object, or should they be considered as separate entities? For two-object case, we hand segmented foreground object using the AMOS method since it is semiautomatic. However, we stopped the tracker whenever it makes an error and corrected the object boundary accordingly. We observed that even for the experienced users and careful initialization, the generation of ground truth is very exhausting and it takes more than 20 seconds for a single frame on average.

Using the binary ground truth $G(p) = (1 : \text{object}, 0 : \text{background})$, we calculate a point misclassification score $E_{\text{pixel}}(t)$ at frame t as follows:

$$E_{\text{pixel}}(t) = \frac{1}{|R(t)|} \sum_{p} |G(p) - R(p)|, \tag{22}$$

where $R(p) = 1$ if the point p is inside the object, and $|R(t)|$ is the number of points inside the object. This measure computes the ratio of the misclassified points to the total number of object points in the current frame.

In addition to ground-truth-based measure, we use three other color- and motion-based performance measures (spatial color distance, temporal histogram distance, and motion distance). These measures do not require a ground truth, and depend on these assumptions: object boundaries coincide with both the color and motion boundaries, and the color histogram of the object is stationary from frame to frame. In order to measure the spatial color difference, a set of probe points just inside and just outside of the objects are selected. For the points p_{out}, p_{in} that are at the opposite sides of the object boundary and at an equal distance, the averaged color $\bar{I}(p_{\text{out}})$ and $\bar{I}(p_{\text{in}})$ are computed in the $M \times M$ neighborhood of the corresponding points. The color difference measure along the boundary is calculated as

$$E_{\text{spacol}}(t) = 1 - \frac{1}{|B(t)|} \sum_{p \in B(t)} |\bar{I}(p_{\text{out}}) - \bar{I}(p_{\text{in}})|, \tag{23}$$

where $|B(t)|$ is the total length of the object boundary $B(t)$ in frame t. When the location of the object boundary is estimated correctly, we expect the spatial color measure E_{spacol} to take a small value. However, the converse of this statement is not necessarily true. That is, if the spatial color measure has a small value, this does not imply that the object boundary is located correctly. This color measure is expected to be reliable when the object and background textures are not cluttered and when the color contrast across the boundary is high.

A straightforward way to assess the temporal changes in the segmented object is to calculate the pairwise color histogram differences of the objects at time t and $t-1$. However, a drawback of this approach is that it may not catch a gradual deterioration. Therefore, we can alternatively check the histogram differences between the first and current object regions. This method penalizes the cumulative difference effect of the previous approach. The temporal histogram difference measure is defined as

$$E_{\text{hist}}(t) = 1 - |\gamma_h(i,t) - \gamma_h(i,1)|, \tag{24}$$

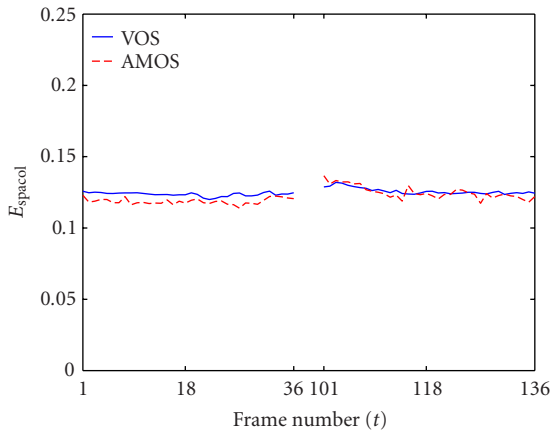
where $\gamma_h(i,1)$ is the normalized framewise color histogram of the object i at the first frame $t = 1$. We used the foreground object for the presented results in Figures 15, 16 and 17. In order to quantify how well the estimated object boundary coincides with actual motion boundaries, we adopt the geometry of the probes used for spatial color difference and consider the difference of the average motion vectors in the neighborhood the points. The motion measure for frame t is estimated as follows:

$$E_{\text{motion}}(t) = 1 - \frac{1}{|B(t)|} \sum_{p \in B(t)} 1 - e^{-|\mathbf{v}(p_{\text{out}}) - \mathbf{v}(p_{\text{in}})|}. \tag{25}$$

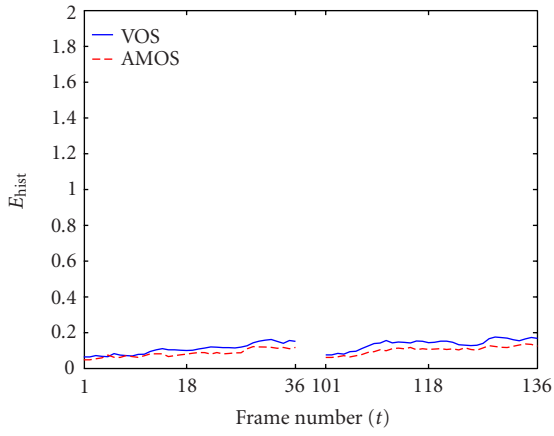
The motion difference can sometimes be large, not because of errors in segmentation, but as a consequence of the fact that not all parts of the object is moving or having a uniform translational motion.

7.3. Ground truth for motion field

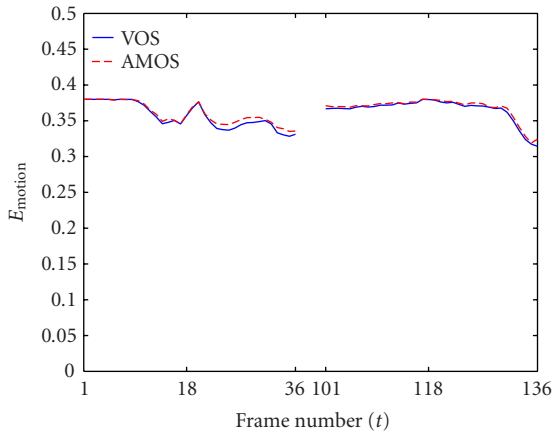
We implemented a dense optical flow estimation method [28, 29] to generate the ground-truth motion vectors as illustrated in Figure 18. This is done only for comparison, and this dense field is not a part of the proposed segmentation framework. Instead of the simple block matching, we used phase correlation which is a frequency domain motion measurement method that makes use of the shift property of the Fourier transform. Phase correlation takes the advantage of the fact that a shift in the spatial domain is equivalent to a phase shift in the frequency domain. Using the rotation



(a)

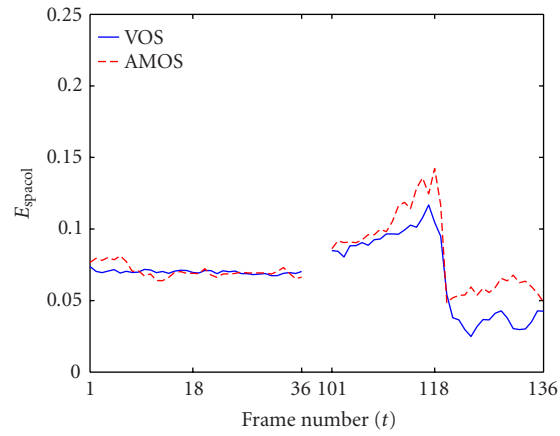


(b)

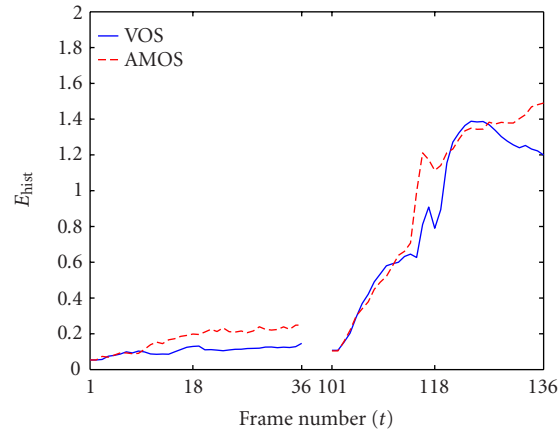


(c)

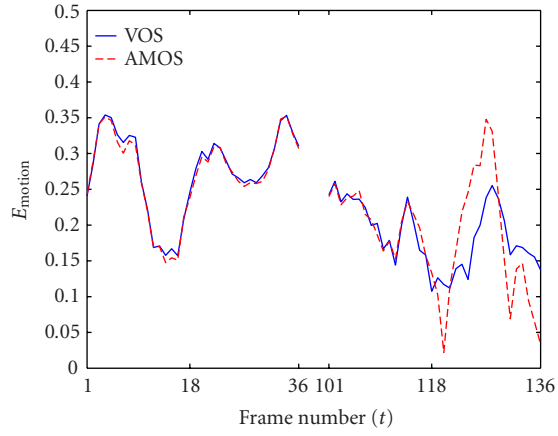
Figure 15: Comparison of (a) the spatial color distance, (b) temporal histogram distance, and (c) motion distance measures for *Akiyo*.



(a)

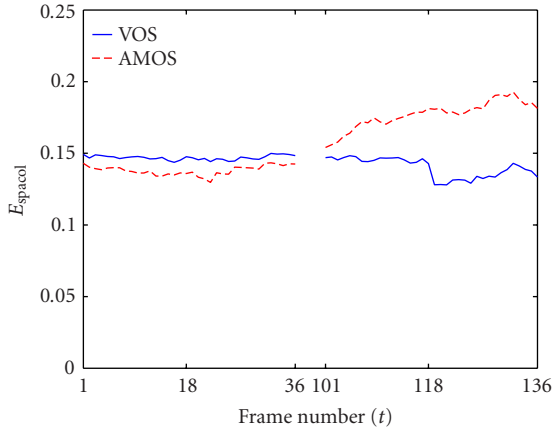


(b)

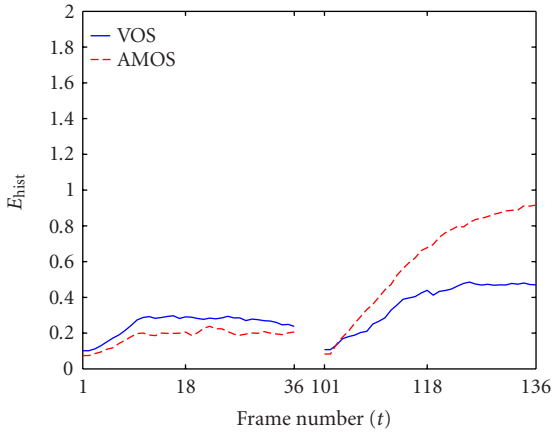


(c)

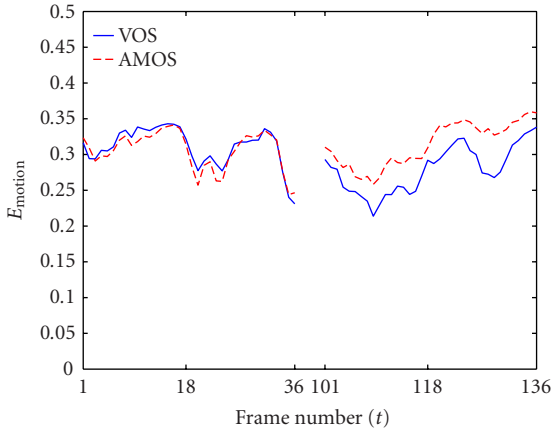
Figure 16: Comparison of (a) the spatial color distance, (b) temporal histogram distance, and (c) motion distance measures for *Bream*. When the AMOS cut most of the fish, its spatial color and temporal histogram errors became very large in comparison to VOS.



(a)

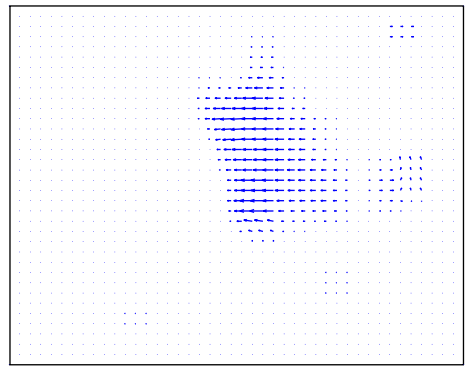


(b)

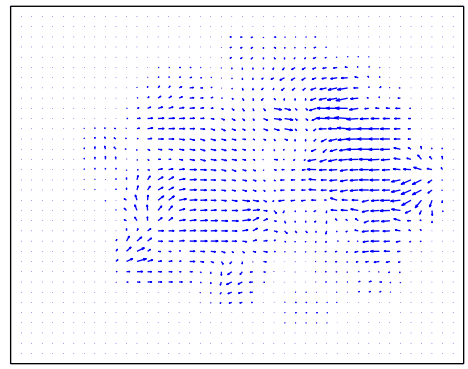


(c)

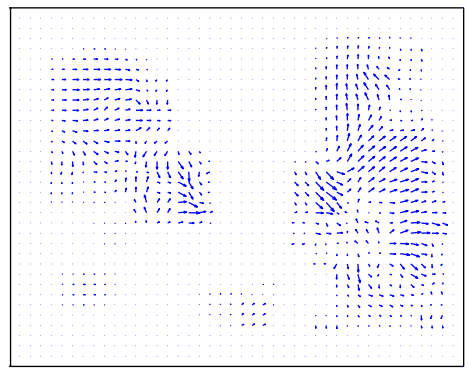
Figure 17: Comparison of (a) the spatial color distance, (b) temporal histogram distance, and (c) motion distance measures for *Children*.



(a)



(b)



(c)

Figure 18: Estimated ground-truth motion vectors using phase correlation for the frame 101 of (a) *Akiyo*, (b) *Bream*, and (c) *Children*.

and scale properties of the Fourier transform, it is possible to find the rotation and scale as a shift in the frequency domain invariant to any translation. We first window both images due to repeating nature of the frequency spectrum, and calculate its Fourier transform. We filter out the DC component and any high-frequency noise. We then calculate the normalized cross power spectrum above. We take the inverse Fourier transform, and find peak on correlation surface. An interpolation is done finally on the surface to achieve subpixel accuracy. Phase correlation is limited by the number of samples that the Fourier transform can use, thus limiting the resolution in the frequency domain. Therefore, the block size is chosen as 32×32.

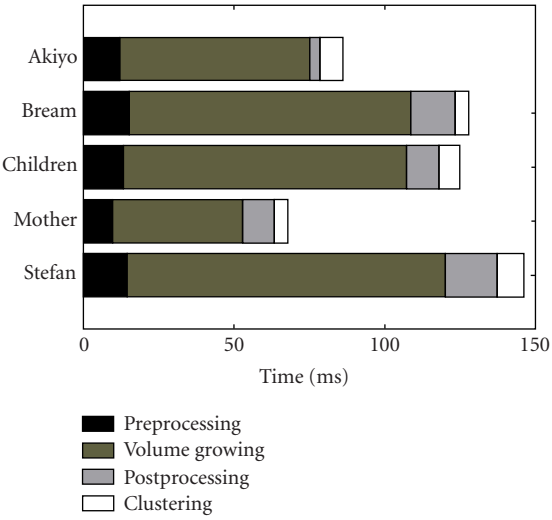


Figure 19: Average processing times of different components for a single frame. Preprocessing includes filtering and threshold adaptation. Volume growing includes marker selection and one-at-a-time growing. Postprocessing includes volume refinement and descriptor extraction.

7.4. Discussion on results

We extensively tested the proposed algorithm and the reference methods. For the AMOS method, we carefully marked the initial boundary by mouse clicking on more than 50 boundary points. The initial boundary is aligned on the object as close as possible. Then we segmented the sequence for a total of 136 frames. We generated spatiotemporal data for the same-sized video and run the automatic segmentation as mentioned before.

For the VOS method results presented in this section, we did not fine-tune any parameters but only modified the multipliers in the clustering stage since they are related with the semantic definition which differs for each sequence. We set the multipliers of the hierarchical clustering stage as $\lambda_\rho = 1$, $\lambda_{\text{others}} = 0$ for *Akiyo*, $\lambda_\mu = 1$, $\lambda_{\text{others}} = 0$ for *Bream*, and $\lambda_\mu = 1$, $\lambda_{\text{others}} = 0$ for *Children* to be able to extract semantically similar objects as the hand-generated ground truth, that is, face, fish, children, and ball.

We performed experiments for 320×240 of YUV video on a P4 1.8 Ghz CPU. In Figure 19, we show the average processing time for each module of the proposed method for various test sequences. The differences among the processing times are a result of the spatial color distribution and the number of small volumes going into the volume refinement. For instance, for the *Bream* sequence, the fine texture on the fish causes several small volumes to be removed. On the other hand, the smooth background and the relatively larger volumes after the volume growing keep the computational time low for *Akiyo*. Table 2 shows the averaged CPU processing time of a frame and preparation time required before the segmentation for the semiautomatic methods. For a small number of experienced users, we counted the initial boundary marking time for the reference methods. As presented in the table, most users spend more than 30 seconds to enter the initial object boundary for the AMOS and SAM methods. The preparation time for the VOS method indicates the time required for threshold adaptation and memory handling before the segmentation. We observed that both of the SAM and VOS methods have close speeds (100 miliseconds/frame) although the SAM algorithm requires additional 30 seconds for boundary initialization. Moreover, we observed that the segmentation results of the SAM deteriorated after only a small number of frames (around 10 frames) and requires halting the tracking process and correcting the boundary. The AMOS method needs more time to process a frame (more than 2 seconds) but is more stable. Thus, we compared the segmentation accuracy with the better-performing AMOS method.

Table 2: Processing times of single frame.

	Processing			Preparation		
	VOS	AMOS	SAM	VOS	AMOS	SAM
Akiyo	86 ms	2.1 s	70 ms	27 ms	36 s	25 s
Bream	128 ms	6.3 s	113 ms	25 ms	45 s	35 s
Children	125 ms	5.5 s	120 ms	25 ms	55 s	35 s
Mother	68 ms	7.2 s	123 ms	25 ms	40 s	23 s
Stefan	157 ms	2.2 s	87 ms	28 ms	30 s	25 s

We present the segmentation results in Figure 20. The proposed method consistently produces both visually and quantitatively better results. In Figure 21, the misclassification errors are plotted for the VOS (blue) and AMOS (red). For *Akiyo*, the error scores are similar due to the minor differences between the extracted boundaries. However, the semiautomatic method (AMOS) fails to maintain the correct boundary on the left side of the head and starts expanding after certain number of frames. This also happens whenever object moves fast, which causes the tracker to miss the part of the object, as in the case of *Children* when the boy on the left suddenly kneels down. For *Bream*, the proposed method manages to detect the correct boundary even when the fish changes its direction. On the other hand, the motion estimation and boundary fitting mechanisms of the AMOS cannot compensate for this movement as a result object boundary is significantly deformed. One shortcoming of the proposed method is that the volume refinement process may drop a grown volume if it fails to satisfy size criterion. For instance, the size of the volume corresponding to the feet of the boy on the left in *Children* is less than the threshold, thus it is not included among the volumes send to the clustering stage. Still, it is evident that the proposed algorithm has results superior to those of than the reference one.

The computed point discrepancy measure (given in Figure 21) also confirms these observations. We used (22) to find the misclassification scores.

In Figures 15, 16, and 17, we present the nonground-based performance measure results. These graphs confirm the ground-truth results, although for some certain conditions the sensitivity of the motion and temporal color

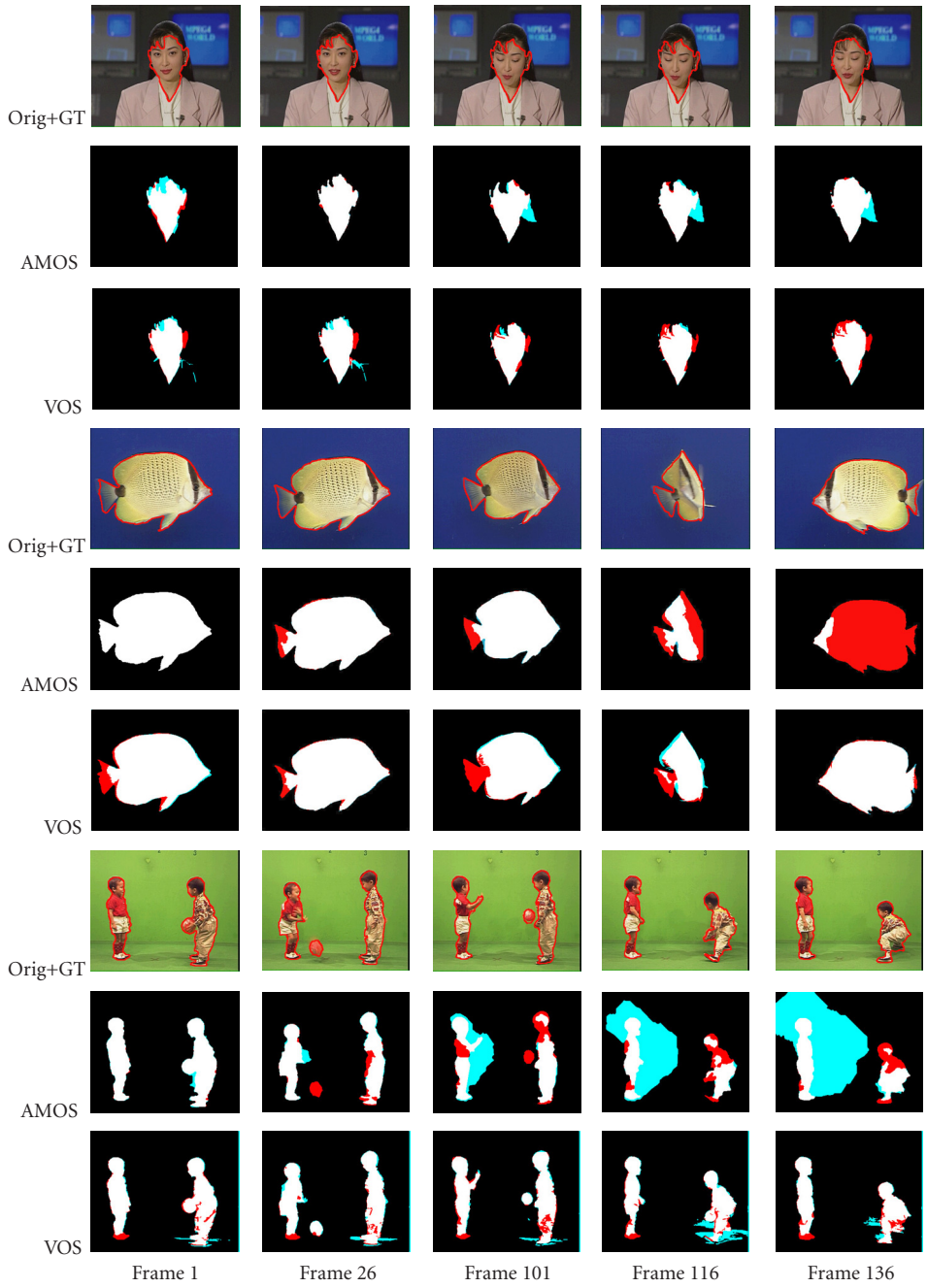
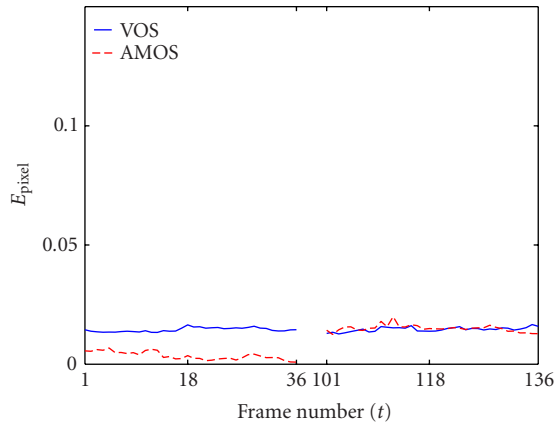


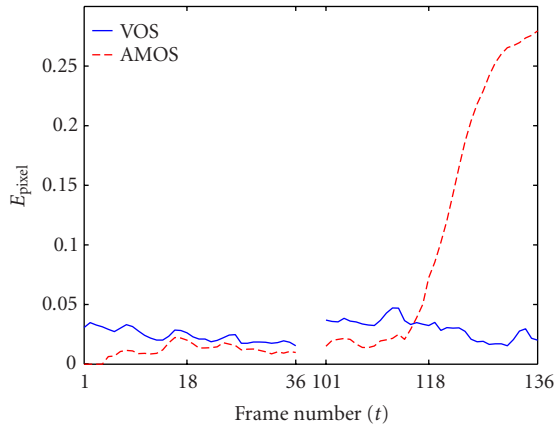
FIGURE 20: Segmented objects for frames 1, 26, 101, 116, and 136 of test sequences. Ground truth is marked by a red boundary in the original images. The red areas in the segmented images show undersegmented pixels which are missed. The cyan areas correspond the oversegmented regions where the algorithm exceeded object boundary. White + cyan areas show what segmentation generates.

distance are limited. In Figure 22, we give a plot of performance measures versus object levels and frame numbers. As visible, the errors decrease for most frames as the object number gets smaller until it reaches 2, which was the intention of foreground/background segmentation. However, we observed that the measures do not always comply with this observation since they depend on the previously described assumptions.

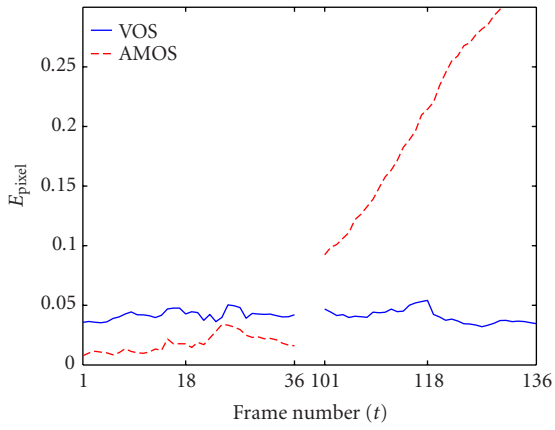
In Figure 23, the highest-similarity scores $P(V_{i,j})$ at each object level are plotted for different test sequences. One hypothesis is that if the clustering stage "accurately" merges two volumes at the current level (k), in the highest likelihood in the next object level $(k-1)$ will be less than the highest value at the current level (k). Otherwise, a possible merge, which has higher likelihood value, would be missed since it is encountered in the following level (that is how we



(a)

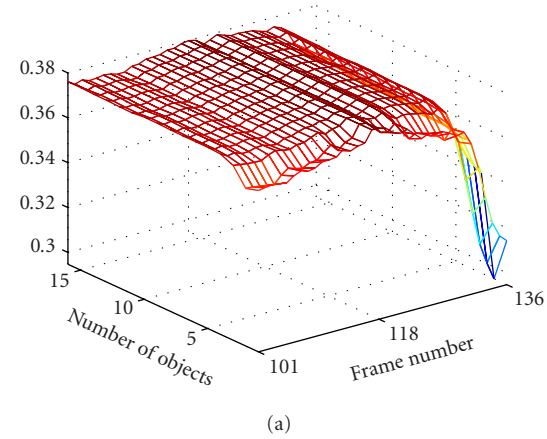


(b)

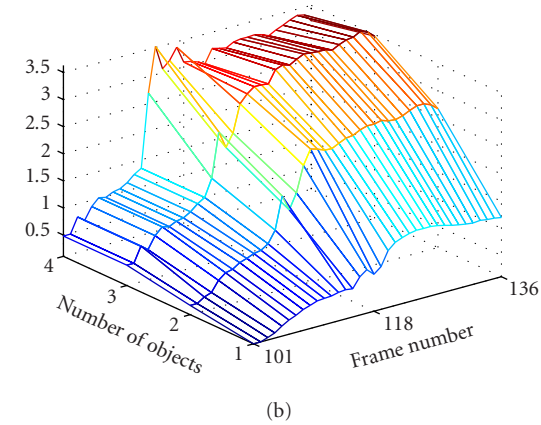


(c)

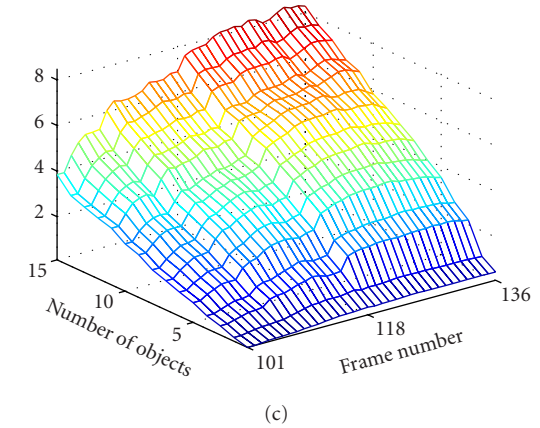
Figure 21: Misclassification errors (22) of the proposed segmentation framework (VOS) and a semiautomatic method (AMOS) using manually extracted ground truths for (a) *Akiyo*, (b) *Bream*, and (c) *Children*.



(a)



(b)



(c)

Figure 22: Performance measures for different object levels in the hierarchical clustering: (a) the motion distance for *Akiyo*, (b) the spatial color distance for *Bream*, and (c) the temporal histogram distance for *Children*. As expected, the temporal histogram errors consistently dropped for the smaller object numbers.

find the existence of such a merge). This hypothesis is justified by the object level versus performance measure plot as shown in Figure 23. These plots show that the highest likelihood drops as the object level decreases, which also indicates that the merging process works accurately.

We also analyzed the effects of the color quantization as shown in Figure 24. By quantizing the 3D space into 256 levels, we are able to decrease the computational load by 15% without causing a degradation of the segmentation

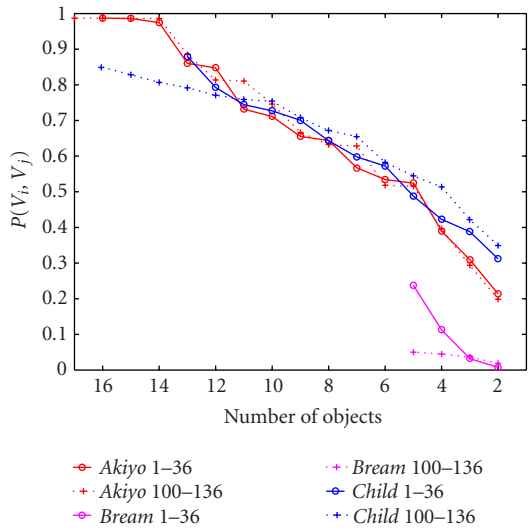


FIGURE 23: Highest similarity score monotonically decreases as the volumes are merged. Note that after volume growing, the number of volumes are different for each sequence. Large decreases indicate potentially weak merges.

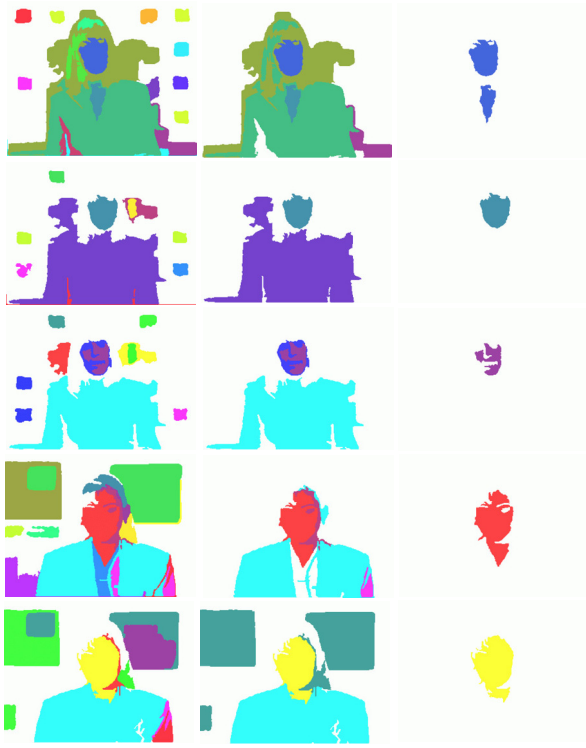


FIGURE 24: Effects of quantization by 256, 64, and 16 dominant colors. Quantization decreases the computational load. However, with the decreasing number of quantization levels, the extracted volume boundaries become more sensitive to the quantization errors. First row: 256 color levels *Head* sequence for 17, 6, and 2 objects after clustering. Second row: 64 levels for 10, 3, and 2 objects. Third row: 16 levels for 11, 4, and 2 objects. Fourth row: 32 levels *Akiyo* for 18, 6, and 2 objects. Last row: 16 levels for 11, 4, and 2 objects.

performance. This gain is a result of using shorter data structures for memory handling in the implementation. Further quantization, that is, into 64 and 32 levels, requires platform-specific data structures. Severe quantization, that is, into 16 and 4 levels, significantly disturbs the volume boundaries and washes out skin colors.

8. SUMMARY

We introduced an automatic segmentation framework. The main stages of the presented automatic segmentation framework are filtering and simplifying color distributions, calculating feature vectors, assigning markers as seeds of volumes, volume growing, removal of volume irregularities, deriving self and relational descriptors of volumes, and clustering volumes into a multiresolution object tree. Several alternatives for each of the preceding stages have been explored.

For volume growing, we discussed several linkage methods: single linkage, dual linkage, and centroid linkage. We proposed threshold adaptation techniques for centroid-linkage method as well. Furthermore, we compared various modes of the volume growing. Out of these, the simultaneous growing and one-at-a-time growing methods basically differ in the number of markers that are active at each iteration. The recursive diffusion and intraframe/interframe switching methods offer different expansion mechanisms. We assigned self descriptors to quantify individual volumes. We also introduced the relational descriptor concept which evaluates the similarity between a pair of volumes. In addition to descriptors that capture general attributes such as motion and shape, we discussed ways to integrate application-specific features, such as skin color and frame difference, into the descriptors. Hierarchical clustering approach was adapted to group volumes into objects. We used a rank-based similarity measure of volumes. We proposed a multiresolution object tree representation as an output of the segmentation. This framework blends the advantages of color-, texture-, shape-, and motion-based segmentation methods in an automatic and computationally feasible way.

Our experiments proved the effectiveness and accuracy of the proposed framework.

As a future work, we plan integrating the previously mentioned texture and available compressed domain features to the automatic segmentation framework.

REFERENCES

[1] Y. Ohta, *A region-oriented image-analysis system by computer*, Ph.D. thesis, Kyoto University, Japan, 1980.

[2] B. Schachter, L. S. Davis, and A. Rosenfeld, "Some experiments in image segmentation by clustering of local feature values," *Pattern Recognition*, vol. 11, no. 1, pp. 19–28, 1979.

[3] P. J. Burt, T. H. Hong, and A. Rosenfeld, "Segmentation and estimation of image region properties through cooperative hierarchical computation," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 11, no. 12, pp. 802–809, 1981.

[4] P. Salembier and M. Pardas, "Hierarchical morphological segmentation for image sequence coding," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 639–651, 1994.

[5] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-generation image coding techniques," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 549–574, 1985.
[6] J. Crespo, R. Schafer, J. Serra, C. Gratin, and F. Meyer, "The flat zone approach: A general low-level region merging segmentation method," *Signal Processing*, vol. 62, no. 1, pp. 37–60, 1998.
[7] W. B. Thompson and T. G. Pong, "Detecting moving objects," *International Journal of Computer Vision*, vol. 4, no. 1, pp. 39–57, 1990.
[8] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 625–638, 1994.
[9] G. Adiv, "Determining three-dimensional motion and structure from optical flow generated by several moving objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 7, no. 4, pp. 384–401, 1985.
[10] P. Bouthemy and E. Francois, "Motion segmentation and qualitative dynamic scene analysis from an image sequence," *International Journal of Computer Vision*, vol. 10, no. 2, pp. 157–182, 1993.
[11] B. Duc, P. Schroeter, and J. Bigun, "Spatio-temporal robust motion estimation and segmentation," in *Proc. 6th Int. Conf. Computer Analysis of Images and Patterns*, pp. 238–245, Springer-Verlag, Prague, September 1995.
[12] D. W. Murray and B. F. Buxton, "Scene segmentation from visual motion using global optimization," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 2, pp. 220–228, 1987.
[13] J. K. Aggarwal, L. S. Davis, and W. N. Martin, "Corresponding processes in dynamic scene analysis," *Proceedings of the IEEE*, vol. 69, no. 5, pp. 562–572, 1981.
[14] I. K. Sethi and R. Jain, "Finding trajectories of feature points in a monocular image sequence," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 56–73, 1987.
[15] R. Deriche and O. Faugeras, "Tracking line segments," in *Proc. European Conference on Computer Vision*, O. Faugeras, Ed., vol. 427 of *Lecture Notes in Computer Science*, pp. 259–268, Springer-Verlag, Antibes, France, April 1990.
[16] M. J. Black, "Combining intensity and motion for incremental segmentation and tracking over long image sequences," in *Proc. European Conf. Computer Vision*, vol. 588 of *Lecture Notes in Computer Science*, pp. 485–493, Santa Margherita Ligure, Italy, May 1992.
[17] F. Meyer and P. Bouthemy, "Region-based tracking using affine motion models in long image sequences," *CVGIP: Image Understanding*, vol. 60, no. 2, pp. 119–140, 1994.
[18] F. Porikli, "Image simplification by robust estimator based reconstruction filter," in *Proc. 16th Int. Symposium on Computer and Information Sciences*, November 2001.
[19] W. Skarbek and A. Koschan, "Colour image segmentation: A survey," Tech. Rep. 94-32, Department of Computer Science, Technical University of Berlin, 1994.
[20] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.
[21] F. Porikli, *Video object segmentation*, Ph.D. thesis, Electrical and Computer Engineering Department, Polytechnic University, New York, 2002.
[22] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, Boston, Mass, USA, 1st edition, 1992.
[23] S.-F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong, "A fully automated content-based video search engine supporting spatiotemporal queries," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 602–615, 1998.
[24] D. Zhong and S. F. Chang, "Long-term moving object segmentation and tracking using spatio-temporal consistency," in *Proc. International Conference on Image Processing*, vol. 3, pp. 57–60, Thessaloniki, Greece, October 2001.
[25] A. Alatan, L. Onural, M. Wollborn, R. Mech, E. Tuncel, and T. Sikora, "Image sequence analysis for emerging interactive multimedia services-the European COST 211 framework," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 802–813, 1998.
[26] T. Ida and Y. Sambonsugi, "Self-affine mapping system and its application to object contour extraction," *IEEE Trans. Image Processing*, vol. 9, no. 11, pp. 1926 –1936, 2000.
[27] C. E. Erdem, A. M. Tekalp, and B. Sankur, "Video object tracking with feedback of performance evaluation measures," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 4, pp. 310–324, 2003.
[28] B. Reddy and B. Chatterji, "An FFT-based technique for translation, rotation and scale-invariant image registration," *IEEE Trans. Image Processing*, vol. 5, no. 8, pp. 1266–1271, 1996.
[29] L. Hill and T. Vlachos, "Shape adaptive phase correlation," *Electronics Letters*, vol. 37, no. 25, pp. 1512–1513, 2000.

Fatih Porikli received the B.S. degree in electronic engineering from Bilkent University, Ankara, Turkey, in 1992, and the M.S. and Ph.D. degrees in electrical and computer engineering from Polytechnic University, Brooklyn, NY, in 1996 and 2002, respectively. He joined the Mitsubishi Electric Research Laboratories, Cambridge, Mass in 2000 after working on stereoscopic depth estimation at AT&T Labs Research in 1997 and developing satellite imagery classification methods at Hughes Research Labs in 1999. Previously, he designed algorithms for post-filtering, network management, and optimal bandwidth allocation. More recently, his research focused on computer vision and data mining, automatic object detection and tracking, unusual event detection, video content analysis, and multicamera surveillance applications. He is serving as an Associate Editor for SPIE Journal of Real-Time Imaging and a Senior Member of IEEE and ACM.

Yao Wang received the B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1983 and 1985, respectively, and the Ph.D. degree in electrical and computer engineering from University of California at Santa Barbara in 1990. Since 1990, she has been with the faculty of Polytechnic University, Brooklyn, NY, and is presently a Professor of electrical and computer engineering. She was on sabbatical leave from the Princeton University in 1998 and was a Visiting Professor at the University of Erlangen, Germany, in the summer of 1998. She was a Consultant with AT&T Labs Research, formerly AT&T Bell Laboratories, from 1992 to 2000. Her research areas include video communications, multimedia signal processing, and medical imaging. She is the leading author of a textbook titled *Video Processing and Communications*, and has published over 100 papers in journals and conference proceedings. She is a Senior Member of IEEE and has served as an Associate Editor for IEEE Transactions on Multimedia and IEEE Transactions on Circuits and Systems for Video Technology. She received New York City Mayor's Award for Excellence in Science and Technology in the Young Investigator Category in 2000.