Object Segmentation of Color Video Sequences

Fatih Murat Porikli

Mitsubishi Electric Research Labs,
Murray Hill, NJ 07974, USA,
fatih@merl.com

Abstract. We present a video segmentation algorithm that accurately finds object boundaries, and does not require any user assistance. After filtering the input video, markers are selected. Around each marker, a volume is grown by evaluating the local color and texture features. The grown volumes are refined and motion trajectories are extracted. Self-descriptors for each volume, mutual-descriptors for a pair of volumes are computed from trajectories. These descriptors designed to capture motion, shape as well as spatial characteristics of volumes. In the fine-to-coarse clustering stage, volumes are merged into objects by evaluating their descriptors. Clustering is carried out until the motion similarity of merged objects at that iteration becomes small. A multi-resolution object tree that gives the video object planes for every possible number of objects is generated. Test results prove the effectiveness of the algorithm.

1 Introduction

It is possible to segment video objects either under user control, semi-automatic, or unsupervised, i.e., fully automatic. User-controlled segmentation is exceedingly laborious and obviously far from processing amount of nowadays video data. In the semi-automatic case, user can provide segmentation for the first frame of the video. The problem then becomes one of video object tracking. In the fully automatic case, the problem is to first identify the video object, then to track the object through time and space with no user assistance. Methodically, object segmentation techniques can be grouped into three classes: region-based methods using a homogeneous color criterion [1], object-based approaches utilizing a homogeneous motion criterion [2], and object tracking [3]. Although color-oriented techniques work well in some situations where the input data set is relatively simple, clean, and fits the model well, they lack generality and robustness. The main problem arises from the fact that a video object can contain totally different colors. On the other hand, works in the motion oriented segmentation domain start with an assumption that a semantic video object has homogeneous motion [4]. These motion segmentation works can be simply separated into two broad classes: boundary placement schemes and region extraction schemes [5]. Most of them are based on rough optical flow estimation or unreliable spatiotemporal segmentation. As a result, they suffer from the inaccuracy of motion boundaries. The last class of methods that is related to semantic video

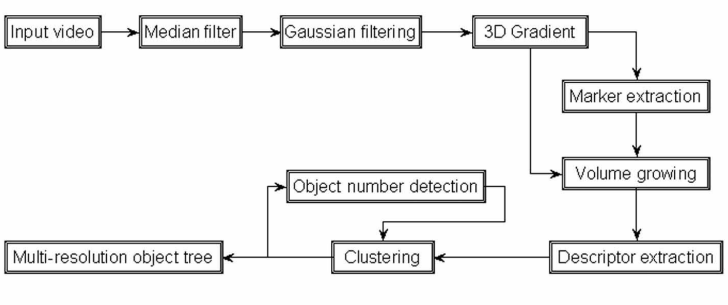


Fig. 1. Flow diagram of the video segmentation algorithm.

object extraction is tracking [6]. However, the tracking algorithms need user interface, and the performance of the tracking algorithms depends extensively on the initial segmentation. In general, most of the object extraction algorithms treat segmentation as a 2-D inter-or-intra frame processing problem with some additional motion model assumptions or smoothing constraints by ignoring the 3-D nature of the video data.

To develop an algorithm that blends intra-frame color and texture based spatial segmentation schemes with inter-frame motion estimation techniques, we consider video sequence as a 3-D volumetric data, that we call it as video-cube, but not a collection of 2-D images. Thus, the semantic object information can be propagated forward and as well as backward in time without saddling into initial segmentation accuracy or tracking limitations.

A general framework of the algorithm is shown in Fig. 1. In Section II, the video-cube concept is introduced. Section III describes the stages of filtering, marker selection, volume growing, refining, and clustering volumes into objects. The test results and discussion are included in the last section.

2 Formation of Video-Cube

By registering the raw and processed image frames along the time axis as shown in Fig. 2, a video-cube $V(x, y, t)$ $1 \leq x \leq x_M$, $1 \leq y \leq y_M$, and $1 \leq t \leq t_M$ is constricted. Each element of video-cube V corresponds to a vector $\mathbf{v}(x, y, t) = [y, u, v, \theta_1, \ldots, \theta_K]^T$ that consists of color and texture features of the spatiotemporal point (x, y, t). Here, y, u, and v stand for the luminance and chrominance features, $\theta_1, \ldots, \theta_K$ are the normalized texture features. For simplicity, we will denote each component as a subscript, e.g., $\mathbf{v}_y$ instead of $\mathbf{v}(x, y, t)_y$.

We preferred the YUV color space over the RGB. Most of the existing color segmentation approaches have utilized the RGB color space although the RGB space has machine oriented chromatics rather than human oriented chromatics.

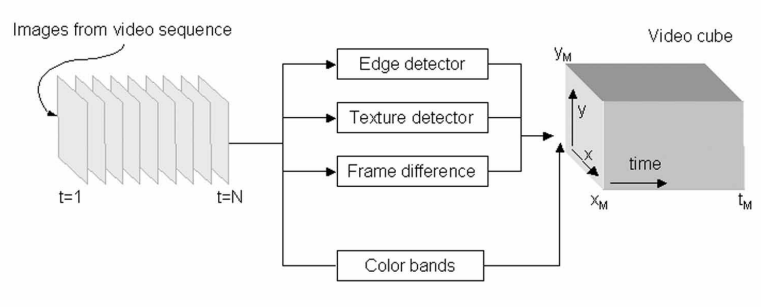


Fig. 2. Video-cube generation from the raw and processed images of the video sequence.

One other disadvantage of the RGB space is the dependency of all three parameters from the light intensity. The YUV space performs in accordance with human reception and more importantly, inter-color distances can be computed using the L^2-norm.

The texture components $\mathbf{v}_{\theta_1}, \ldots, \mathbf{v}_{\theta_K}$ are obtained by Gabor transform [7]. Gabor filters are quadrature filters and can be used to extract a certain wavelength and orientation from an image with a specified bandwidth. 2-D Gabor filters $h(x, y)$ have the functional form

$$h(x,y) = g(x,y)e^{-2\pi(ux+vy)}, \quad g(x,y) = \frac{1}{2\pi\sigma_g^2} e^{-\frac{x^2+y^2}{2\pi\sigma_g^2}} \tag{1}$$

where σ_g^2 specifies effective width, and u, v specify modulation that has spatial frequency $f = \sqrt{u^2 + v^2}$ and direction $\theta = tan^{-1}(v/u)$. Then the texture scores are found by

$$\mathbf{v}_\theta = |h(x,y) \otimes I(x,y)|. \tag{2}$$

We chose the values for the spatial frequency $f = 2, 4, 8$ and the direction $\theta = 0, \pi/4, \pi/2, 3\pi/4$, which leads to a total of 12 features. The computed texture scores are normalized as described in [8].

3 Object Segmentation

3.1 Pre-Filtering

The color channels of the input video $\mathbf{v}_y, \mathbf{v}_u, \mathbf{v}_v$ are first filtered to remove out noise. Another reason of pre-filtering is to prepare video-cube to the volume growing stage. Elimination of the image flickers prevents from excessive segmentation, and decreases computation load significantly. A fast 3×3 median filter [9] that exploits 2-D coherence is utilized together with a 5×5 Gaussian filter to remove noise and smoothen image irregularities.

3.2 Marker Assignment

A video object is assumed to be made of smaller parts. Such parts are the group of points that are spatially consistent, i.e., color and texture distributions are uniform within. The grouped points, called as volumes, are expanded from seed points, called markers, as in the watershed transform [10]. The marker points serves as "basins" in the growing process. Note that, not the spatial position of a marker but its features are intended here. For each marker, a volume is assigned and volume's attributes are initialized. Therefore, a marker point should be selected such that it can characterize its enclosing volume as relevant as possible. The points have low local diversity are good candidates to represent their local neighborhood. A marker point m_i can be selected in three ways:

- Uniformly distributed: The V is divided into identical smaller cubes and their centers are selected as markers. However, an edge point can be chosen as well, which reduces the accuracy of volume growing.
- Minimum gradient magnitude: Markers are selected if the gradient magnitude is minimum. Let S be the set of all possible spatiotemporal points, i.e., it is all the points of V initially. The gradient magnitude is computed from the color channels, and the minimum gradient magnitude point is chosen as a marker. A preset neighborhood around that marker is removed from the set S. The next minimum in the remaining set is chosen, and selection process repeated until no point remains in the video-cube.
- Minimum gradient with volume growing: The minimum m_i is chosen as above. Instead of removing a preset neighborhood around the marker, a volume W_i is grown as explained in the next section, and all the points of the volume is removed from the set S

$$m_i = \arg\min_S \nabla V(x,y,t) \quad ; \quad S = V - \bigcup_{j=1}^{i} W_j. \tag{3}$$

Finding minimum is a computationally expensive process. Rather than searching the full-resolution video-cube V, a down-converted version is used. More computational reduction is achieved by dividing down-converted video-cube V into slices in time or other axes. Minimum is found for the first slice, and a volume is grown, then the next minimum is searched in the next slice, and so forth.

3.3 Volume Growing

Volumes are enlarged as "inflating balloons" from markers by applying distance criteria. For each volume W_i, a feature vector ω^i that is similar to the video-cube point's feature vector is defined. Two distance criteria d_g,d_l are designed. The first criterion d_g measures the distance between the feature vector of the current volume and the candidate point. In the spatial sense, this criterion is a volume-wise "global" measure. The second criterion d_l determines the distance between the feature vectors of the current volume and another point that is already included in the current volume and also adjoint to the candidate. Thus,

the second criterion can be viewed as a "local" measure. A global threshold ϵ_g helps limiting the range of feature distance, i.e., color variation in the volume. Local threshold ϵ_l prevents from trespassing edges even the global threshold permits. The global and local thresholds are adaptively determined from the video-cube V. Let x^- be an unmarked candidate point that is adjoint to the current volume. Let x^+ be another point adjoint to x^- but already included in the current volume W_i. Then, the first global distance d_g is defined as

$$d_g(\omega^i, \mathbf{v}^-) = \sum_k |\omega_k^i - \mathbf{v}_k^-| \qquad k : y, u, v, \theta_1, ..., \theta_{12} \tag{4}$$

where $\mathbf{v}^-$ and $\mathbf{v}^+$ are the feature vectors of x^- and x^+. Similarly, the second local distance d_n is

$$d_l(\omega^i, \mathbf{v}^-) = \sum_k |\mathbf{v}_k^+ - \mathbf{v}_k^-| \qquad k : y, u, v, \theta_1, ..., \theta_{12}. \tag{5}$$

If the distances d_g and d_l are smaller then ϵ_g and ϵ_l, the point x^- is included in the volume W_i. The neighboring point x^- is set as an active surface point for W_i, and the feature vector for the marker is updated accordingly. In the next iteration, the neighboring pixels of the active surface points are examined. Volume growing is repeated until no point remains in the video-cube.

The thresholds are made adaptable to input video by using the variance and dynamic range of the features. Variance of a feature gives information about the distribution of that feature in the video-cube. A small variance indicates smooth distribution, whereas, high variance refers to texture and edgeness. To The global threshold should be tuned up large if the variance is high and it should be scaled to the dynamic range. Let I represent a feature, i.e, $I \equiv Y$ for luminance. The global variance σ^2 and mean η are simply

$$\sigma^2 = \frac{1}{M} \sum_{x,y,t \in V} (I(x,y,t) - \eta)^2, \qquad \eta = \frac{1}{M} \sum_{x,y,t \in V} I(x,y,t). \tag{6}$$

where M is the total number of points. The dynamic range μ for is

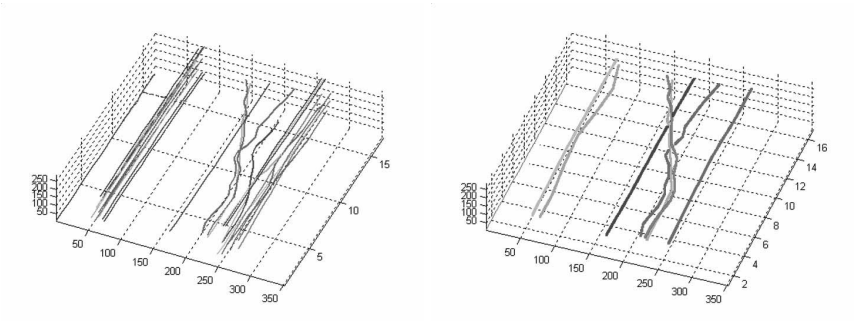
$$\mu = \max I(x,y,t) - \min I(x,y,t) \ , \quad x,y,t \in V. \tag{7}$$

Then the global threshold ϵ_g is then assigned by scaling the dynamic range as

$$\epsilon_g = \kappa \frac{\mu}{\sigma + 1} \tag{8}$$

where $\kappa > 1$ is the sensitivity parameter that sets how fine the final segmentation should be. It is observed that a good choice is $\kappa \approx 2$. The local threshold $\epsilon_l(x,y,t)$ is the average of the discontinuity between the neighboring point features scaled with a relative edgeness score:

$$\begin{aligned}
\epsilon_l(x,y,t) &= \tilde{\eta} \cdot \tilde{\mu}(x,y,t) \\
\tilde{\eta} &= \frac{1}{2M} \sum_{x,y,t \in V} |I(x,y,t) - I(x-1,y,t)| + |I(x,y,t) - I(x,y-1,t)| \\
\tilde{\mu}(x,y,t) &= \max I(i,j,t) - \min I(i,j,t) \qquad x-2, y-2 \le i,j \le x+2, y+2
\end{aligned}$$



(a) (b)

Fig. 3. Trajectories; (a) before clustering, (b) at the 7^{ht} object level of Children seq.

To fasten extraction of these statistics, only the first image of the video can be used instead of the whole video-cube V.

After volume growing, the video-cube is divided into multiple smaller parts. Some of these parts are negligible in size, however, they effect the computational load if the clustering stage. Also, some points, such as edges, are not grouped into any volume at all. To assign ungrouped points to a volume, the volume growing thresholds are relaxed iteratively. At each iteration, volumes are inflated towards the unmarked points until no more point remains. Small volumes are blended into the bordering most similar volumes that gives the best combination of the greatest mutual surface, the smallest color distance, the smallest mutual volume, and the highest compactness ratio as defined in the next section.

3.4 Self and Mutual Descriptors

Descriptors are used to understand various aspects of the volumes. The volumes W_i are represented by a set of self descriptors $f(i)$ and mutual descriptors $g(i,j)$ as summarized in Table 1. These descriptors identify motion, spatial, and color characteristics, as well as the mutual correlation. The volumes will be grouped with respect to their descriptors at the clustering stage in order to assemble the objects. For each volume W_i, a trajectory $T_i(t) = [X_i(t), Y_i(t)]^T$ is extracted by computing the frame-wise averages of volume's points coordinates

$$T_i(t) = \begin{bmatrix} X_i(t) \\ Y_i(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{N_t} \sum x_t \\ \frac{1}{N_t} \sum y_t \end{bmatrix} \quad ; \quad (x_t, y_t, t) \in W_i. \tag{9}$$

Trajectories are the center of masses of regions in an image frame, hence they approximate the translational motion of the region. This is a nice property that can be used to initialize parameters in motion model fitting stage. Sample trajectories can be seen in Fig. 3 a-b. Then, the distance $\Delta d_{ij}(t)$ between the trajectories $T_i(t)$ and $T_j(t)$ at time t is calculated to determine motion based descriptors

$$\Delta d_{ij}(t) = \sqrt{(X_i(t) - X_j(t))^2 + (Y_i(t) - Y_j(t))^2}. \tag{10}$$

color mean	$f_1(i)$	$\frac{1}{N_i}\sum \mathbf{v}_y(x_k)\ ,\ x_k \in W_i$
volume	$f_2(i)$	$\bigcup x_k\ ,\ x_k \in W_i$
surface	$f_3(i)$	$\sum x_k \cap x_l\ ,\ x_k \in W_i\ x_l \in W_j\ i \neq j$
compactness	$f_4(i)$	$f_2(i)/(f_3(i))^2$
vertical translation	$f_5(i)$	$y_{1,i} - y_{N,i}$
horizontal translation	$f_6(i)$	$x_{1,i} - x_{N,i}$
route length	$f_7(i)$	$\sum \lvert T_i(t) - T_i(t-1)\rvert$
average x position	$f_8(i)$	$\frac{1}{N_i}\sum X(x_k)\ ,\ x_k \in W_i$
average y position	$f_9(i)$	$\frac{1}{N_i}\sum Y(y_k)\ ,\ x_k \in W_i$
existence	$f_{10}(i)$	$\sum i_t\ ;\ i_t = 1 \leftarrow T_i(t) \neq 0$
mean of distance	$g_1(i,j)$	$\frac{1}{N_i \cap N_j}\sum \Delta d_{ij}(t)$
variance of distance	$g_2(i,j)$	$\frac{1}{N_i \cap N_j}\sum (\Delta d_{ij}(t) - g_1(i,j))^2$
maximum distance	$g_3(i,j)$	$max \Delta d_{ij}(t)$
directional difference	$g_4(i,j)$	$\sum \lvert T_i(t) - T_i(t-1) - T_j(t) + T_j(t-1)\rvert$
compactness ratio	$g_5(i,j)$	$\frac{f_4(W_i \cup W_j)}{f_4(W_i)+f_4(W_j)}$
mutual boundary ratio	$g_6(i,j)$	$f_3(i) + f_3(j) - f_3(W_i \cup W_j)/f_3(i)$
color difference	$g_7(i,j)$	$\lvert f_1(i) - f_1(j)\rvert$
coexistence	$g_8(i,j)$	$\sum i_t \wedge j_t\ ;\ i_t = 1 \leftarrow T_i(t) \neq 0$

Table 1. Self and Mutual descriptors

Motion characteristics such as vertical and horizontal motion, route length, mean and variance of distance, direction difference, and average change in the distance are derived from the trajectories. Therefore, without a computationally expensive method, the motion information is blended into segmentation efficiently. Each descriptor is linearly normalized to $[0, 1]$ by using its maximum and minimum at last.

3.5 Clustering

A fine-to-coarse type hierarchical merging method is chosen since the video-cube already divided into small parts before the clustering. The most similar volume pairs are merged to decrease the number of the volumes at each iteration. We define "similarity" as the degree of relevance of volumes in motion and shape. Two volumes are similar if their motion trajectories are consistent and they built a compact shape when combined together. Color aspects are omitted; partly because it was already included in volume growing, and also portions of a semantically meaningful object do not have to possess the same color aspects, i.e., a human face made up from different color regions, mouth, skin, hair, etc.

The most suitable descriptors are found to be variance of trajectory distance $g_2(i,j)$ for motion, and the compactness $g_4(i,j)$ and mutual boundary ratio $g_6(i,j)$ for shape relevance. Each volume W_i is compared to its neighboring volumes W_j, and a similarity score $S(i,j)$ for the pair is determined if the pair satisfies a set of constraints. The constraint set is embedded to prevent from

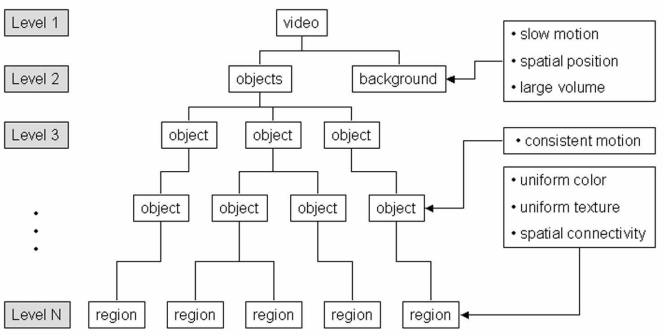


Fig. 4. Multi-resolution object tree.

degenerate cases. An example of such a case is that a volume encircling a smaller volume and causing the highest compactness ratio. Although such volumes motions may be inconsistent, still they will get a high similarity score. Another example of degenerate case is parts of the two objects that are not adjoint but perfectly aligned in motion. The shape relevance will be very low; although the motion is consistent, the similarity score could be small. An effective similarity score is formulated as

$$S_{ij} = -\log(g_2(i,j)) + g_4(i,j) + g_6(i,j). \tag{11}$$

After all similarity scores are computed for the possible volume pairs, the volume pair (i', j') that gives the maximum similarity score are merged together, the descriptors are updated and normalized accordingly.

3.6 Object Number Estimation

Clustering is performed until the candidate volumes to be merged become inconsistent in motion. Motion inconsistency is measured by the variance of the trajectory distance descriptor $g_2(i,j)$. If the trajectory distance $g_2(i'_m, j'_m)$ of the most consistent pair in the current object level m is considerably higher than the $g_2(i'_{m-1}, j'_{m-1})$ at the previous level, the last merge is assumed to be a violation of the motion similarity. Thus, segmentation is halted:

$$\frac{\partial g_2(i'_m, j'_m)}{\partial m} \gg 1 \;\; ; \quad i'_m, j'_m = \arg\max(S_{ij}) \quad at \;\; level \;\; m \tag{12}$$

A multi-resolution object tree as illustrated in Fig. 4 is generated from the clustering results. By multi-resolution tree, segmentation is not repeated in case the number of objects is changed later. Also, this structure enables imposing relational constraints and analyzing object properties by graph theory methods.

4 Test Results and Conclusion

The proposed algorithm has been tested with standard MPEG sequences. Fig. 5 presents sample results. The first row (Fig. 5 a-b) shows original frames from two test sequences. The initial volumes after volume growing stage are given in the second row (Fig. 5 c-d). Here, volumes are color coded for illustration purposes. The initial number of volumes are 42 and 27 respectively. We noticed that the adaptive threshold estimation method enables us to confine the initial number of volumes to $20 \sim 50$ which is a very reasonable initial range for most sequences. Figures 5 e-f are the intermediate clustering results. Segmentation stopped at $m = 4$ and $m = 2$ object levels respectively without any user interface. The backgrounds and object boundaries were detected accurately.

The test results confirmed that the object extraction is robust even when the motion is large. Because no separate motion computation is involved in segmentation, our algorithm is computationally faster than any optical flow based or motion field modeling method that usually need some initial segmentation. Having an obvious advantage over the stochastic segmentation techniques, an object-wise multi-resolution representation is generated after clustering; therefore the extraction objects is not repeated in case the number of objects is changed. No user segmentation of object regions is involved, which is mostly required by object tracking based methods.

References

1. M. Kunt, A. Ikonomopoulos, and M. Kocher: Second generation image coding. Proceedings of IEEE, no.73, 549–574, (1985)
2. P. Bouthemy and E. Francois: Motion segmentation and qualitative dynamic scene analysis from an image sequence. Int. J. Comput. Vision, no 10, 157–187, (1993)
3. F. Meyer and P. Bouthemy: Region-based tracking using affine motion models in long image sequences. CVGIP-Image Understanding, 119–140, no 60 (1994)
4. B. Duc, P. Schtoeter, and J. Bigun: Spatio-temporal robust motion estimation and segmentation. Proc. Comput. Anall. Images and Patterns, 238–245 (1995)
5. J. Wang and E. Adelson: Representing moving images with layers. IEEE Transaction on Image Processing, no.3 (1994)
6. J. K. Aggarwal, L. S. Davis, and W. N. Martin: Corresponding processes in dynamic scene analysis. Proceedings of IEEE, no.69, 562–572 (1981)
7. A.K. Jain and F. Farrokhnia: Unsupervised texture segmentation using Gabor filters. Pattern Recognition, vol.24, 1167–1186 (1991)
8. O. Pichler, A. Teuner, and B. Hosticka: An unsupervised texture segmentation algorithm with feature space reduction and motion feedback. IEEE Transaction on Image Processing, 53–61 (1998)
9. M. Kopp and W. Purgathofer: Efficient 3x3 median filter computations. Technical University, Vienna (1994)
10. J. Serra and P. Soille: Watershed, hierarchical segmentation and waterfall algorithm. Proc. of Mathematical Morphology Appl. Image Process., 69–76 (1994)
11. A. Jain, M. Murty and J. Flynn, Data Clustering: A Review. ACM Computing Surveys, Vol. 31, 264–323 (1999)

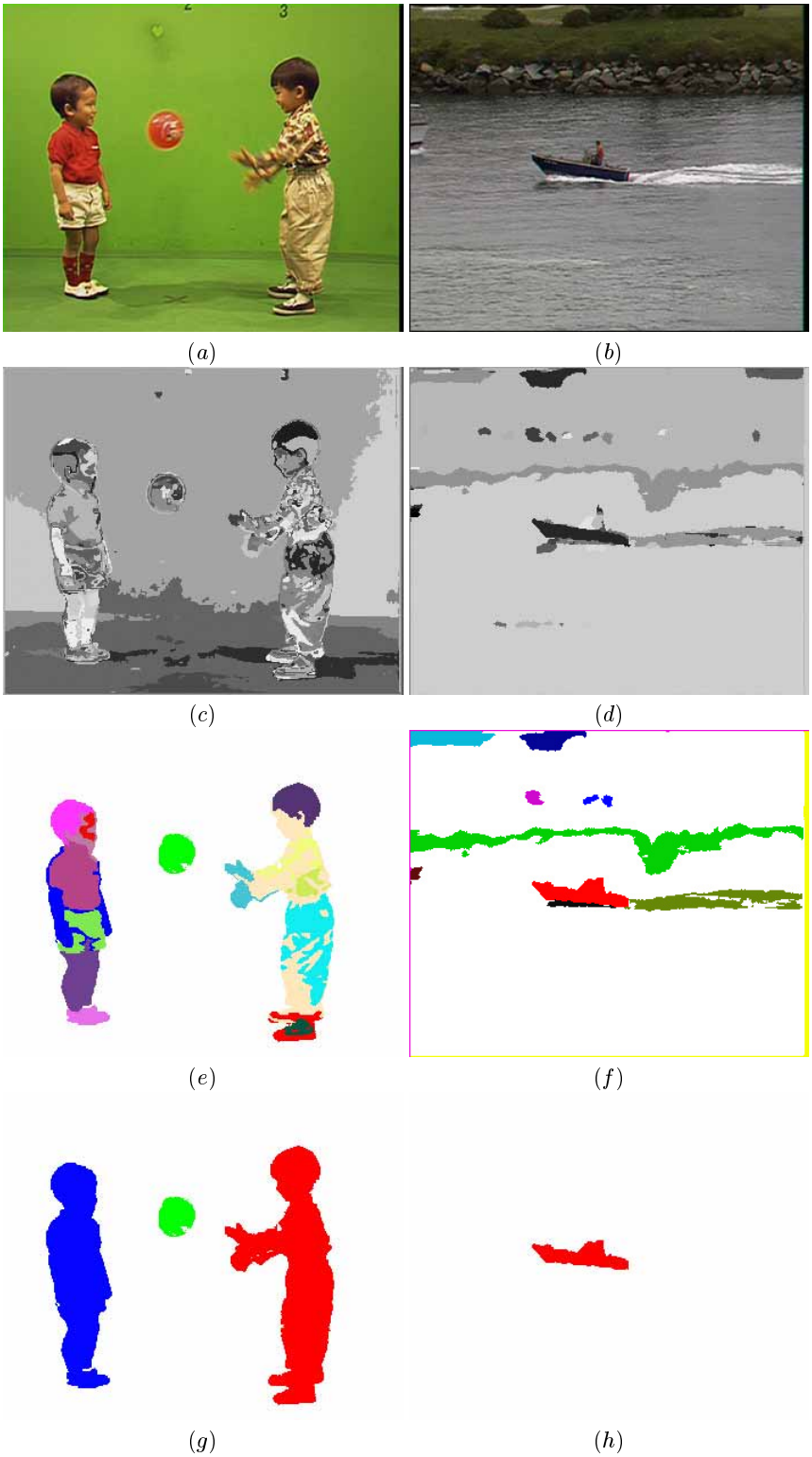


Fig. 5. (a-b) Frames form the input sequences, (c-d) initial volumes after volume growing, (e-f) intermediate clustering results at object levels $m = 20 - 12$ respectively, and (g-h) the final segmentation results.