Video Object Tracking With Feedback of Performance Measures

Çiğdem Eroğlu Erdem, *Member, IEEE*, A. Murat Tekalp, *Fellow, IEEE*, and Bülent Sankur, *Senior Member, IEEE*

***Abstract*—We present a scalable object tracking framework, which is capable of tracking the contour of nonrigid objects in the presence of occlusion. The framework consists of open-loop boundary prediction and closed-loop boundary correction parts. The open-loop prediction block adaptively divides the object contour into subcontours, and estimates the mapping parameters for each subsegment. The closed-loop boundary correction block employs a suitably weighted combination of low-level features such as color edge, color segmentation, motion models, and motion segmentation for each subcontour. Performance evaluation measures are used in a feedback loop to evaluate the goodness of the segmentation/tracking in order to adjust the weights assigned to each of these low-level features for each subcontour at each frame. The framework is scalable because it can be adapted to track a coarse estimate of the boundary of selected objects in real-time, as well as pixel-accurate boundary tracking in off-line mode. The proposed method does not depend on any single motion or shape model, and does not need training. Experimental results demonstrate that the algorithm is able to track the object boundaries under significant occlusion and background clutter.**

***Index Terms*—Active contours, object tracking, performance evaluation.**

I. Introduction

The PROBLEM of two-dimensional (2-D) object tracking has attracted much attention due to its many applications in computer vision and video processing, including surveillance, content-based indexing and retrieval, object-based video coding, and video post-production. Some of these applications, such as surveillance, require automatic real-time processing while tolerating some performance inaccuracy. In other applications, such as video-post processing and object-based coding, accuracy is very important while the processing may not need to be done in real time and a reasonable amount of user interaction is allowed (in fact even desired). Thus, it is of interest to develop a generic scalable video-object tracking framework that can address all of these diverse requirements.

The simplest approach for fully automatic object segmentation/tracking is to use blue screening (chroma keying), which requires special video-capture apparatus. Other approaches for automatic segmentation assume that the background is stationary (or has global motion that can be compensated using a parametric model) and the object of interest is moving independently of background motion. Then, semantic object segmentation can be achieved by change detection and/or motion segmentation [1]. Other methods impose constraints on the shape of the tracked object by using 2-D [2], [3] or three–dimensional (3-D) shape [4], [5] models. Some algorithms acquire the 2-D-shape space information through training [6], [7], and use projections onto the shape space to estimate the most likely object boundary at a certain frame. The Condensation algorithm [8], which is a state–space sampling approach, needs the shape space to be known beforehand. Another application of learned motion models for tracking is presented in [9]. Pfinder [10] is a blob-tracking method that runs in real time, assuming that the background is relatively stationary. Another blob-tracking approach that uses color histogram of the tracked object is presented in [11]. Semiautomatic segmentation/tracking approaches incorporate minimal user interaction to resolve the ambiguity of semantic meaning. A recent such technique [12] tracks the visible boundary of the object under occlusion, where the initial object boundary is defined interactively and the boundary is modeled using active contours. In another recent work [13], the boundary of the tracked region is modeled using a hidden Markov model which incorporates different image cues and utilizes a joint probability data association filter to compute the transition probabilities. While some of these methods emphasize real-time tracking of approximate object boundaries, others address pixel-accurate object tracking in off-line mode. However, none is scalable to address both requirements in a single generic framework, nor do they provide a method for automatic adjustment of weights for each contour segment. Some of the object-segmentation/tracking algorithms in the literature are tabulated in Table I, together with their properties.

In this paper, we propose a scalable video-object tracking framework, which employs performance evaluation measures in a feedback loop. An overview of the framework is given in Section II. In the (near) real-time mode, our scheme employs an open-loop contour tracking paradigm, which is presented in Section III. In the pixel-accurate tracking mode, a feedback loop based on performance measures comes into play as discussed in Section IV. The object contour is divided into subcontours, and several low-level features such as color edge, color segmentation, motion models, and motion-segmentation information are employed in the feedback loop to track each subcontour. Section V details the main novelty of the proposed closed-loop

Manuscript received June 3, 2002; revised December 1, 2002. This work was supported by the Scientific and Technical Research Council of Turkey (TÜBİTAK-BAYG) and by Boğaziçi University Research Fund under project 99A203.

Ç. E. Erdem and B. Sankur are with the Department of Electrical and Electronics Engineering, Boğaziçi University, İstanbul 80815, Turkey (e-mail: cigdem@ieee.org; sankur@boun.edu.tr).

A. M. Tekalp is with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627 USA and also with the College of Engineering, Koç University, Rumelifeneri Yolu, Sarıyer, İstanbul, Turkey (e-mail: tekalp@ece.rochester.edu; mtekalp@ku.edu.tr).

Digital Object Identifier 10.1109/TCSVT.2003.811361

TABLE I
SOME OF THE OBJECT SEGMENTATION/TRACKING METHODS IN THE LITERATURE

Algorithm by	Important Features	Training & Assumptions	Shape Model	Method Used	Specific Task
Isard [8]	Uses B-splines	Learn shape space, constant velocity	Shape space known	State-space sampling	Track objects of shape space
Wren [10]	Single person, Fixed camera	Learn stationary background	Human body model	Blob tracking	Human tracking
Wachter [5]	Integrate edge & region information	Constant velocity, Interactive initialization	Elliptical cones connected by joints	2D Projection of the 3D model	Human tracking
Fu [12]	Occlusion detection	Draw initial contour	None	Scale-invariant snake	Track occlusion boundaries
Ju [3]	Patches form a chain	Define initial patches	Planar Limbs	Predict location of next patch	Track human limbs
Cham [2]	High dimensional state space	State-space search	Scaled prismatic model, defines initial model	Multiple hypothesis testing	Articulated objects
Comaniciu [11]	Color histogram	Initial region marked	No shape model (elliptical region),	Histogram matching using mean-shift	Track non-rigid objects (regions)
Baumberg [7]	B-splines, uniform motion	Learn shape space automatically	Eigenshapes	Kalman filter	Track human motion
W^4 [14]	Gray scale images	Learn background, Learn projection templates	Silhouettes of people	Shape analysis and tracking	Surveillance of people
Proposed Method	Performance feedback	Initial boundary marked	None	Feature tracking, adaptive snake	Rigid & non-rigid objects

scheme, i.e., the use of feedback performance scores to automatically adjust the weight parameters in the energy minimization. The performance metrics used are presented in the Appendix. Section VI provides experimental results to demonstrate the performance improvements.

II. Overview of the Framework

We propose a scalable framework for 2-D video object tracking as shown in the block diagram in Fig. 1. The block depicted by the dashed lines on the left part of Fig. 1 performs a coarse tracking of the location of the object from frame $t-1$ to t in real-time or near real-time. The global motion compensation box is activated in the presence of fast camera motion, such as zoom or pan. If a feature point cannot be properly tracked due to occlusion, temporal and spatial prediction schemes can be employed for that point until (and if) it becomes uncovered again. We note that this coarse boundary-tracking block may be replaced by other real-time region (blob) tracking schemes without affecting the rest of the framework.

However, the output of the coarse tracker does not always yield pixel-accurate contour localization. Thus, the remainder of the proposed framework, which is indicated by dashed lines on the right of Fig. 1, deals with enhancing the accuracy of the tracked contour (possibly for non real-time applications) by using a closed-loop energy-minimization scheme. The boundary is modeled as a union of contour segments that minimize a set of energy terms derived from edges [15], color-segmentation boundaries, and motion-segmentation boundaries [16]. After each iteration, the goodness of the tracking results are evaluated using a set of performance metrics (that do not need ground-truth information) to fine tune the weights assigned to each low-level cue, such as color edges, color segments, motion models, etc.

The advantages of the tracking framework can be listed as follows.

- It can track nonrigid objects as well as rigid ones without prior training, since no parametric shape space model is assumed.
- It is robust to abrupt changes in motion direction of the object, since we do not impose any prior motion model to the object.
- The algorithm can track moving objects in cluttered scenes successfully, since motion-segmentation boundaries are taken into account.
- It is reasonably robust to occlusions.

III. Open-Loop Boundary Prediction

In this section, we present a new open-loop boundary-prediction method based on contour segmentation and piecewise contour mapping. We also address handling of occlusions. The open-loop part of the proposed framework can be used both in interactive (non real-time) or real-time mode.

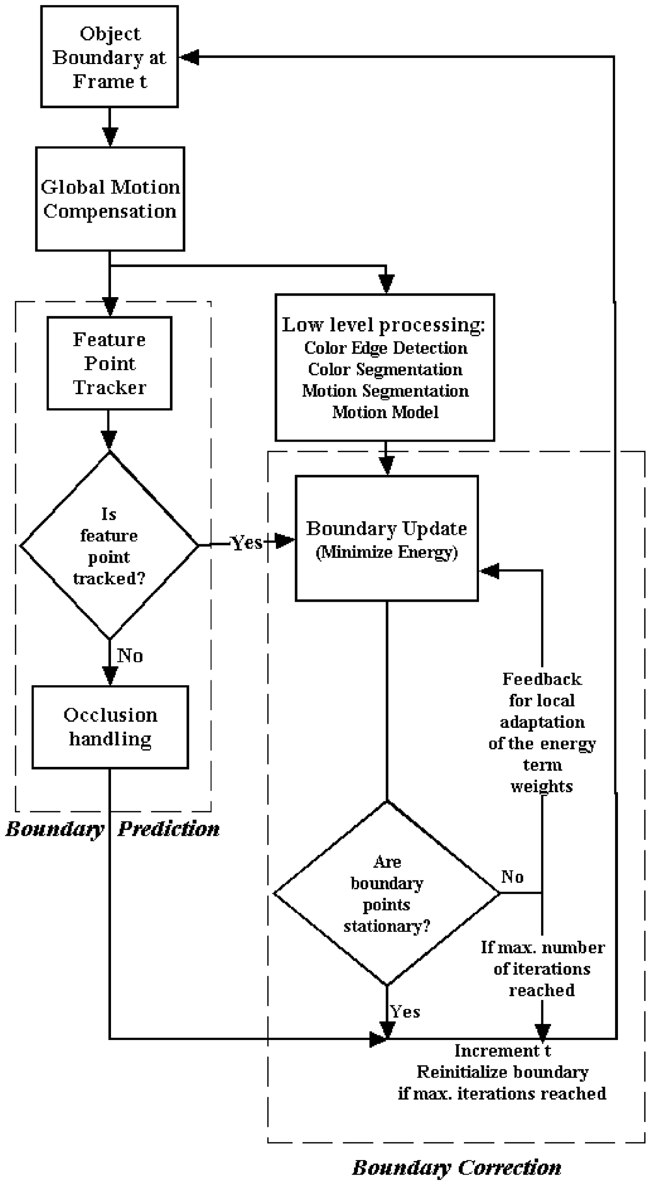
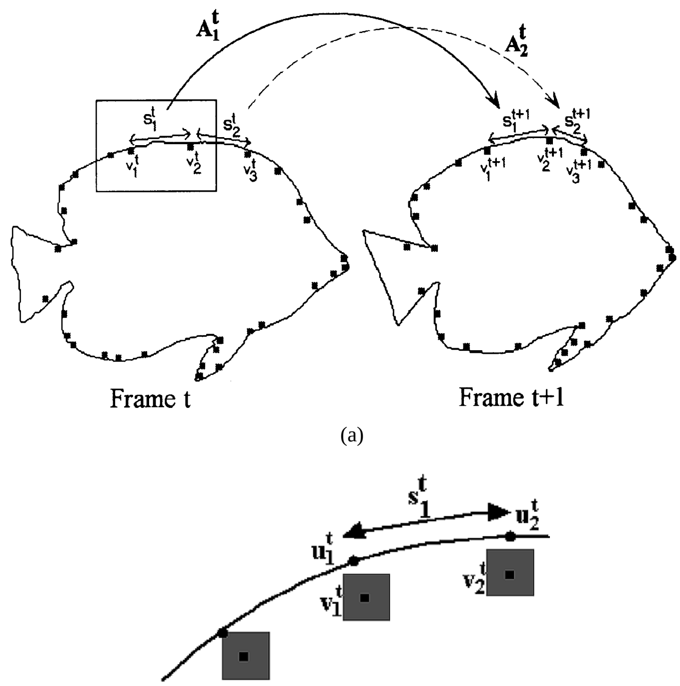


Fig. 1. Tracking framework consists of the *boundary prediction* and the *boundary correction* steps.



(a)

(b)

Fig. 2. (a) Locally adaptive boundary prediction using feature-point tracking. (b) Magnified view of the section of the boundary shown in the box. The large square image blocks denote the feature points, the centers of which are labeled as v_1^t and v_2^t. Feature points consist of 7×7 image blocks represented by their centroids. The small circular marks on the boundary denote the nearest points to each feature centroid. The first segment of the boundary between the first two projected points at time t is labels as s_1^t, where clockwise traversal of the boundary is assumed.

A. Boundary Initialization

In the interactive (non real-time) mode, the object of interest may be determined in the first frame by the user. For example, the intelligent scissors [17] algorithm may be used for this purpose. In the real-time (noninteractive) mode, a change-detection algorithm may be used to determine the object of interest in the first frame under certain assumptions.

B. Boundary Prediction

In the following, we present our new boundary prediction method, where the Shi—Tomasi feature tracking algorithm [18] has been used in one of the steps. The main idea of our algorithm is also illustrated in Fig. 2.

1) Let the boundary of the object (the segmentation map) in the first frame be given, and represented by the $2 \times N_t$ matrix $\mathbf{B^t}$ consisting of all boundary pixels at frame t, where

$$\mathbf{B}^t = [\,\underline{u}_1^t \quad \underline{u}_2^t \quad \cdots \quad \underline{u}_{N_t}^t\,] \tag{1}$$

$$\underline{u}_i^t = [\,x_{i,b}^t \quad y_{i,b}^t\,]^T, \qquad i = 1, \ldots, N_t. \tag{2}$$

The variables $(x_{i,b}^t, y_{i,b}^t)$ denote the location of the ith boundary pixel at frame t.

We first find a given number M_t of "good" feature points [18] inside the segmentation map that are *close* to the boundary. A feature point is close to the boundary if its minimum distance from the boundary is smaller than a certain threshold (3–5 pixels). Let $\mathbf{F}^t$ denote the $2 \times M_t$ matrix of detected feature points, which are arranged in their order of appearance while tracing the boundary

$$\mathbf{F}^t = [\,\underline{v}_1^t \quad \underline{v}_2^t \quad \cdots \quad \underline{v}_{M_t}^t\,] \tag{3}$$

$$\underline{v}_i^t = [\,x_{i,f}^t \quad y_{i,f}^t\,]^T, \qquad i = 1, \ldots, M_t. \tag{4}$$

2) Segment the boundary contour by "projecting" each feature point onto the boundary. First, for each feature point v_i^t, $i = 1, \ldots, M_t$, we find the nearest boundary pixel u_j^t, $j \in \{1, \ldots, N_t\}$ such that

$$\|\underline{v}_i^t - \underline{u}_j^t\| < \|\underline{v}_i^t - \underline{u}_k^t\|, \qquad i = 1, \ldots, M_t, \\ \forall j, k \in \{1, \ldots, N_t\}, j \neq k. \tag{5}$$

Then, the boundary is divided into M_t+1 segments using the projection points found in (5). Therefore, a segmented representation of the contour of the object at time t is given by

$$\mathbf{B}^t = [\mathbf{s}_1^t \quad \mathbf{s}_2^t \quad \cdots \quad \mathbf{s}_{M_t}^t] \tag{6}$$

where each submatrix $\mathbf{s}_i^t$ $(2\times p_i)$ consists of the locations of the p_i boundary points between the projections of the feature points $\underline{v}_i^t$ and $\underline{v}_{i+1}^t$ onto the boundary. Note that $p_1+p_2+\cdots+p_{M_t}=N_t$, so that $\mathbf{B}^t$ is $2\times N_t$. The selected feature-point pairs $(\underline{v}_i^t, \underline{v}_{i+1}^t)$ and the associated boundary line segments $(\mathbf{s}_i^t)$ are illustrated in Fig. 2.

3) Track the selected feature points to the next frame using the algorithm in [18]. For each feature point $\underline{v}_i^t$ at frame t, we estimate its location at frame $t+1$, $\underline{v}_i^{t+1}$ (for the time being assume that all feature points have been successfully tracked to the next frame).
4) Using the motion information of each successive pair of feature points $(\underline{v}_i^t \rightarrow \underline{v}_i^{t+1}, \underline{v}_{i+1}^t \rightarrow \underline{v}_{i+1}^{t+1})$, we estimate a transformation matrix for the boundary segment s_i^t such that

$$\begin{bmatrix} x_i^{t+1} & y_i^{t+1} & 1 \\ x_{i+1}^{t+1} & y_{i+1}^{t+1} & 1 \end{bmatrix} = \begin{bmatrix} x_i^t & y_i^t & 1 \\ x_{i+1}^t & y_{i+1}^t & 1 \end{bmatrix} \mathbf{A}_i^t \tag{7}$$

$$\mathbf{A}_i^t = \begin{bmatrix} z_i^t\cos\theta_i^t & -z_i^t\sin\theta_i^t & 0 \\ z_i^t\sin\theta_i^t & z_i^t\cos\theta_i^t & 0 \\ a_i^t & b_i^t & 1 \end{bmatrix} \tag{8}$$

where z_i^t denotes scale parameter, θ_i^t denotes the rotation angle, and a_i^t, b_i^t denote two translation parameters.
5) Transform the ith segment of the boundary at frame $t(s_i^t)$ using the transformation matrix $\mathbf{A}_i^t$ to obtain the predicted object boundary in frame $t+1$.

The user has the option of declaring the whole or parts of the object as rigid. In that case, feature points *inside the rigid parts of the object* may also be used in model parameter estimation to add robustness to the algorithm.

C. Occlusion Handling

Sometimes, a feature point may not be properly tracked to the next frame. Note that, the feature tracker employs block-matching using a block of pixels around the centroid of the feature. A feature point is declared as untracked when a certain MSE threshold is exceeded [18]. There are two main reasons for failure of tracking:

1) the object region around the feature point has been occluded (self-occlusion or occlusion by another object);
2) the feature point is near the boundary (that is, the block centered at the feature point contains pixels from the background) and the background pixels has changed.

If a feature block is not tracked properly, we try to predict its location using motion information of the successfully predicted nearby (temporal or spatial) features. We propose a combined spatial and temporal prediction scheme, which is described in detail in the following.

1) Temporal Prediction: Let the location of a feature block at frame t, which was not tracked to frame $t+1$, be $\underline{v}_i^t$, $i\in\{1,\ldots,M_t\}$. Its location is predicted using the average of its motion vectors in the last K frames

$$\hat{\underline{v}}_i^{t+1} = \underline{v}_i^t + \frac{1}{K}\sum_{k=0}^{K-1} \underline{m}_i^{t-k} \tag{9}$$

$$\underline{m}_i^t = \underline{v}_i^t - \underline{v}_i^{t-1} \tag{10}$$

where $\underline{m}_i^t$ denotes the motion vector of feature block i at frame t and K is the number of frames to average the motion vector. The parameter K may be adjusted depending on the activity present in the scene.

2) Spatial Prediction: Another approach for predicting the location of an untracked feature point at frame $t+1$ is to use other feature points at frame t that have been tracked to frame $t+1$ and have "similar" properties. Two feature points are "similar" if they are spatially close to each other and move coherently. If the object is specified to be rigid, all the feature points can be treated as belonging to the same group.

The temporal and spatial prediction results can be combined to estimate the feature point location.

3) Search Around the Predicted Location: The untracked features and their estimated locations are put in a list. In the next frame, we search around the estimated location to detect whether the feature point has become uncovered again; the matching criterion being the mean squared error. For the feature blocks that contain background pixels, the search is done in such a manner that only the pixels that fall into the segmentation map are considered in the MSE calculation. This is achieved by assigning the background pixels to a uniform value.

IV. Closed-Loop Boundary Correction

The predicted boundary found in the previous section may not exactly match the object boundary due to inaccuracies in feature point tracking and motion estimation. However, the predicted boundary is generally close enough to the actual object boundary to allow for snake-based boundary updating. This section presents the details of the proposed boundary correction scheme, also depicted in the right inset in Fig. 1.

The active contour model was first introduced in [19]. A recent paper [12] presents a new approach of associating the energy terms with boundary segments rather than node points in a dynamic programming energy minimization framework [20]. If only edge-based energy terms are used, the active contour is easily distracted by background clutter or inside texture and may snap to the wrong edges. The new formulation of the energy terms introduced in this section fuses the information from the color-segmentation boundaries, color edges and the motion-segmentation boundaries to prevent such distractions to a certain extent. We associate the energy terms with boundary segments as in [12].

Our boundary representation consists of polygonal snakes. Let $n_i^t = (x_i^t, y_i^t)$, $i=1,\ldots,N$ denote the selected node points along the predicted object boundary at frame t. Note that these points are different from the ones defined in (2) and (5). These node points are selected from the boundary points where the curvature value is high. Intermediate points are inserted if the distance between two consecutive points is large. In the rest of the paper, the superscript t will be dropped. The snake energy of the boundary is expressed as

$$E_{\text{snake}} = \sum_{i=1}^{N} (E_{\text{int},\,i} + E_{\text{ext},\,i}) \tag{11}$$

$$E_{\text{int},\,i} = \beta_{\text{curv},\,i} E_{\text{curv},\,i} \tag{12}$$

$$E_{\text{ext},\,i} = \beta_{\text{col},\,i} E_{\text{col},\,i} + \beta_{\text{edge},\,i} E_{\text{edge},\,i} + \beta_{\text{mot},\,i} E_{\text{mot},\,i} \tag{13}$$

where $E_{\text{col},\,i}$, $E_{\text{edge},\,i}$ and $E_{\text{mot},\,i}$ are external energy terms which are calculated from color-segmentation boundaries, edges, and motion-segmentation boundaries, respectively. $E_{\text{curv},\,i}$ is an internal energy term associated with the curvature of the boundary segment between nodes $i-2$ and i. The parameters $\beta_{\text{curv},\,i}$, $\beta_{\text{col},\,i}$, $\beta_{\text{edge},\,i}$, and $\beta_{\text{mot},\,i}$ are the weighting coefficients of the energy terms. The energy terms are elaborated in this section, and the weighting coefficients are discussed in Section V.

The snake energy is minimized using the dynamic programming method [20], where the search locations for each node are selected along the normal lines drawn to the $\{n_i\}$ node points as illustrated in Fig. 4. The dynamic programming step is iterated until most of the boundary points (e.g., 95%) are stationary. In the following, we discuss the external and internal energy terms in detail.

A. External Energy Terms

A valid assumption in object segmentation and tracking is that object boundaries coincide with color boundaries. However, color-segmentation boundaries and edges contain different information as can be observed in Fig. 3. If the object is also moving, object boundaries are also expected to coincide with motion boundaries. Based on these assumptions, we present new external energy terms below.

1) Color Boundary Energy: The color segmentation of each frame is estimated using the fuzzy c-means algorithm [21], where the number of classes are determined by the user. Alternatively, the number of classes can also be selected using cluster validity approaches such as the Zhang *et al.* [22] method. Then, the color boundaries are extracted from the segmented frame to obtain the binary contours [see Fig. 3(c)]. This contour map should be transformed such that it will be able to guide the snake nodes to the correct locations, even from distant places. This is achieved by using the Chamfer 3–4 distance transformation [23]. This transform approximates the Euclidean distance of a pixel location to the nearest contour point by transforming the binary contour image using the mask

$$\begin{bmatrix} 4 & 3 & 4 \\ 3 & 0 & 3 \\ 4 & 3 & 4 \end{bmatrix}. \tag{14}$$

In other words, a pixel location that is in the 4-neighborhood of a boundary pixel will have a value of 3, and a pixel which is a diagonal neighbor will have a value of 4. In Fig. 3(d), the Chamfer distance transformation (CDT) of the color-segmentation boundaries is shown. In Fig. 3(e), a plot of a cross section

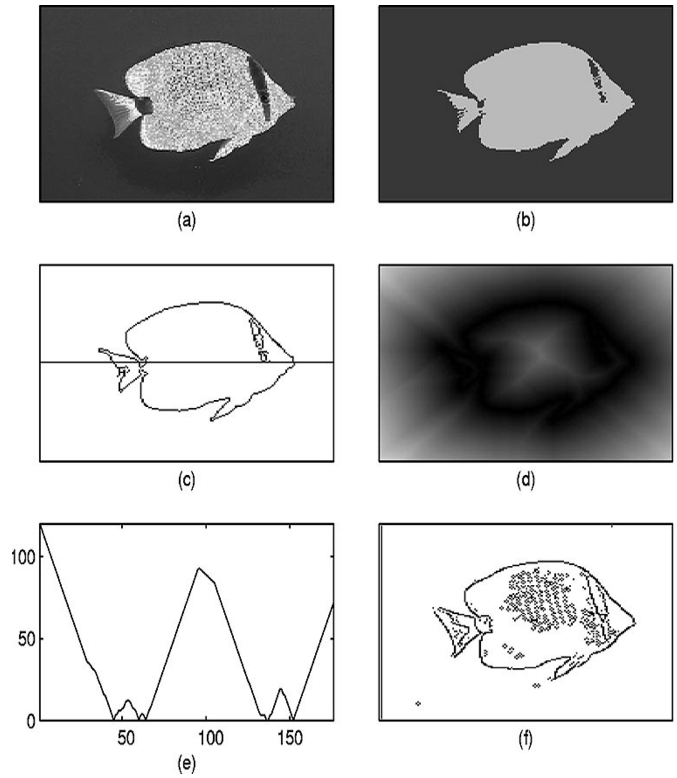


Fig. 3. (a) First frame of "Bream." (b) Two-class color segmentation. (c) Color-segmentation boundaries, where the middle cross section is marked with a line. (d) The Chamfer 3-4 transform of color-segmentation boundaries. Brighter image intensities denote larger values. (e) The plot of the cross section of the Chamfer distance transform corresponding to the line given in (c). (f) Canny edge field.

of the CDT is given. If the value of the Chamfer distance transform of the color segmentation at pixel location u is denoted by $\mathrm{CC}(u)$, then the color boundary energy term is defined as follows:

$$E_{\text{col},\,i} = \frac{\sum\limits_{u \in \overline{n_i(l) n_{i-1}(k)}} \mathrm{CC}(u)}{\|n_i(l) - n_{i-1}(k)\|}, \quad k = 1, \ldots, N_s;\ l = 1, \ldots, N_s. \tag{15}$$

In (15), $\overline{n_i(l) n_{i-1}(k)}$ denotes the line segment that is connecting the current search nodes $n_{i-1}(k)$ and $n_i(l)$ (see Fig. 4) during energy minimization using dynamic programming [20]. The indices k and l denote the search locations along the normal lines (which are the angle bisectors) drawn at nodes $i-1$ and i, respectively and N_s is the total number of search locations at a given node.

2) Edge Boundary Energy: Edges are found using the Canny edge detector [15] as seen in Fig. 3(f). Note that the edge map is quite noisy compared to the color-segmentation boundaries. Rather than finding the image gradient values as in [12], we use the Chamfer distance transformation on the Canny edge field to obtain the terrain that the snake will slide on. This approach gives us a smoothly varying field that enables the snake to move toward the edges even if the image gradient is zero at its current location. This enables the snake to be attracted to the edges from large distances. Let the Chamfer

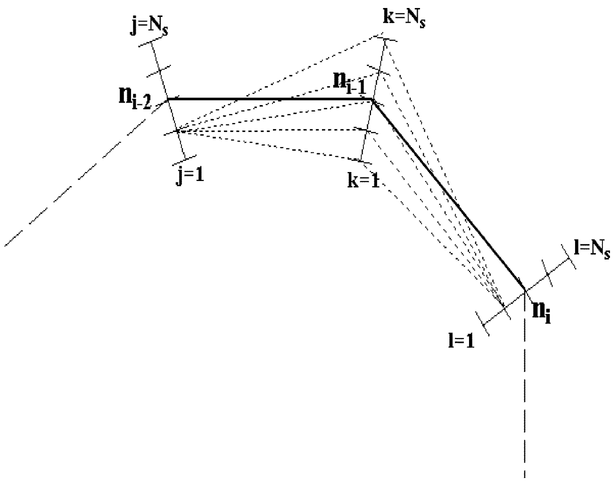


Fig. 4. Illustration of search locations at each node during dynamic programming.

distance transform of the edge map be denoted as $CE(u)$. The edge energy of a searched line segment is found as follows:

$$E_{\text{edge},\,i} = \frac{\sum\limits_{u \in \overline{n_i(l)n_{i-1}(k)}} CE(u)}{\|n_i(l) - n_{i-1}(k)\|}, \quad k = 1, \ldots, N_s;\ l = 1, \ldots, N_s. \tag{16}$$

3) Motion Boundary Energy: When the object is moving with a motion pattern different from the background, the motion boundaries provide an important cue about the object boundaries. However, most dense motion-estimation approaches such as block matching and the Lucas–Kanade method [24], [25] give poor motion boundaries, since they tend to smooth the motion field by using motion constancy assumptions inside blocks. Therefore, methods to refine the motion-segmentation boundaries using the more reliable color boundaries have been suggested [16] to perform reliable motion segmentation. In this section, we introduce a new snake energy term that utilizes this important cue: color-refined motion-segmentation boundaries.

The dense motion estimation is performed by the hierarchical version of the Lucas–Kanade motion estimation algorithm [25]. Following the approach of [16], the dense motion vectors are clustered into affine motion classes using the k-means algorithm. Then, color patches of the current frame obtained by color segmentation are assigned to one of these motion classes by comparing the dense motion field of the color patch with the pre-computed motion cluster centroids. For details the reader is referred to [16].

The estimated binary motion-segmentation boundaries are also transformed using the Chamfer 3–4 transform resulting in the $CM(u)$ field. The external energy term derived from motion-segmentation boundaries is calculated as

$$E_{\text{mot},\,i} = \frac{\sum\limits_{u \in \overline{n_i(l)n_{i-1}(k)}} CM(u)}{\|n_i(l) - n_{i-1}(k)\|}, \quad k = 1, \ldots, N_s;\ l = 1, \ldots, N_s \tag{17}$$

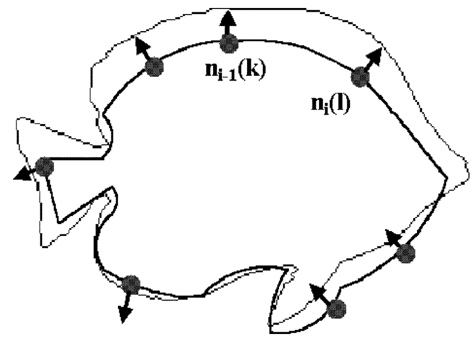


Fig. 5. Illustration of energy minimization using dynamic programming. The correct and estimated (to be updated) object boundaries are shown by the thin and thick lines, respectively. Points $n_i(l)$ and $n_{i-1}(k)$ denote the current and previous search nodes.

where $CM(u)$ denotes the Chamfer distance transform value of the color-refined motion-segmentation boundary at pixel location u.

B. Internal Energy Term

The curvature energy term is estimated using the following equation [12]:

$$E_{curv,\,i} = 2 + 2\cos\left(\angle \overline{n_{i-1}(k)n_{i-2}(j)},\ \overline{n_{i-1}(k)n_i(l)}\right) \tag{18}$$

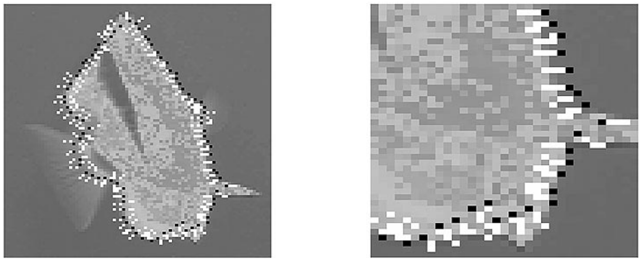
where the indices j, k and l denote the search locations along the normal lines drawn from the nodes $i-2$, $i-1$ and i, respectively, as shown in Fig. 4.

C. Selection of Search Directions

In [12], the search locations during dynamic programming are chosen as uniformly located points along the normal lines (angle bisectors) drawn at boundary nodes. However, a more efficient approach is to use the knowledge of the object and background colors obtained from the tracking results of the previous frames. As illustrated in Fig. 5, the search direction (toward the inside of the segmentation map or toward outside) can be pre-selected at each node. If the color of a search node is an element of the set of object colors, then we only search toward outside of the estimated boundary, otherwise we search toward the inside on the normal line.

In order to make this decision, we first estimate the probability density functions for the object and background colors using the tracking results of the previous frame. This is done by estimating the normalized color histograms. Let $P_o(c(i))$ and $P_b(c(i))$ denote the probability density functions (pdf's) of the object and background colors estimated using the color histograms, where $c(i)$ denotes the color vector at node n_i^t. Let $I(n_i^t)$ be an indicator function for the location of the current boundary node n_i^t (inside or outside the actual object boundary in the current frame), such that

$$I(n_i^t) = \begin{cases} \text{inside}, & \text{if } P_o(c(i)) > P_b(c(i)) + 0.1 \\ \text{outside}, & \text{if } P_b(c(i)) > P_o(c(i)) + 0.1 \\ \text{undecided}, & \text{else.} \end{cases} \tag{19}$$



(a) (b)

Fig. 6. (a) The predicted boundary to be corrected is shown by dark points. The selective search directions are shown by white lines. (b) Close-up of (a). Gray values of the image pixels have been rescaled to the range [64, 192] to increase the contrast between the black and the white markers and the image.

If we cannot decide whether the node point n_i^t is inside or outside of the actual object boundary, we then perform bidirectional search.

This selective search direction approach not only adds robustness to the algorithm but also reduces the search effort. The underlying assumption here is that the background colors are different from the object colors, at least around the object of interest. An example of selective search direction is shown in Fig. 6.

V. Adaptation of Weights Using Performance Scores

In the literature, there are no systematic ways to select the coefficients of the energy terms given in (12) and (13). However, these parameters are critical to the performance of the energy minimization. In this section, we present approaches to adjust the coefficients of the energy terms in a locally adaptive way using the performance evaluation metrics given in the Appendix.

The power of the proposed framework comes from the independent selection of the optimal weights for each subsegment of the object contour. This adaptive-update strategy even includes complete disabling of some energy terms. In traditional snake formulations, a single weight value is assigned for each energy term for the entire object contour. Hence, an energy term, say motion energy, may help to improve the accuracy of a part of the contour, while distorting a different part of the contour, since reliability of the motion information varies across the contour. The proposed formulation enables us to emphasize different energy terms for different parts (segments) of the object contour in order to obtain the best possible results as indicated by quantitative performance measures. This section discusses the selection of the weight factors for each segment of the object contour.

Note that since all the energy terms discussed in the previous section are calculated through the utilization of the Chamfer distance transform, they are inherently normalized, i.e., if we do not have a way of favoring one energy term over another, we can simply choose the weights as equal, which practically leads us to a weight-free snake.

A. Reliability Assessment of Color-Segmentation and Motion-Segmentation Boundaries

Before we try to snap the predicted object boundary to the color-segmentation and motion-segmentation boundaries, we should make sure that they are reliable.

Some parts of the color-segmentation boundaries may not coincide with actual object boundaries due to the inaccuracies of the color-segmentation algorithm or because of the similarity of the object and background colors. We assess the reliability of the color-segmentation boundaries using the color metric $d_{CB}(t;\, i)$ introduced in (29) in the Appendix. First, the color-segmentation boundaries are divided into small segments and the color differences for each segment are calculated using (29). Then, the mean (M_C) and the standard deviation (S_C) of the $d_{CB}(t;\, i)$ values are found. The segments that have $d_{CB}(t;\, i)$ values below $M_C - k \cdot S_C$ are labeled as unreliable. The labeling to obtain a color-segmentation reliability map is done by dilating the binary image consisting of the unreliable color-segmentation boundary segments using a square structuring element. An example is shown in Fig. 7.

In Fig. 7, we can observe that the unreliable segments are accumulated around several places, one of which is the dark (red) stripe above the eye of the fish. Since the color-segmentation map incorrectly bends inward at the location where the (red) dark stripe and the (blue) background meet, the inside–outside color distance at those locations is smaller compared to the other regions of the fish. Another region where unreliable segments are accumulated is the two sides of the rear fin, since the fin is too thin to draw normal lines toward the inside and outside of the boundary.

Similarly, some parts of the motion-segmentation boundaries may not be reliable due to the inaccuracies of one of the steps in the motion-segmentation algorithm. We assess the reliability of the motion-segmentation boundaries using the motion metric $d_M(t;\, i)$ introduced in (31) in the Appendix. First, the motion-segmentation boundaries are divided into small segments and the motion differences are calculated for each segment using (31). Then, the mean (M_M) and the standard deviation (S_M) of the $d_M(t;\, i)$ values are found. The segments that have $d_M(t;\, i)$ values below $M_M - k \cdot S_M$ are labeled as unreliable. The parameter k is chosen between 1–2 during the experiments.

B. Weight Adaptation

After the minimization of the overall energy (11), some segments of the estimated boundary of the object may not coincide with the actual color boundaries. This may indicate that the weight of the color energy term was not high enough (if the color segmentation was found to be reliable as discussed in the previous subsection). Similarly, if a segment of the boundary after energy minimization does not coincide with edges or motion boundaries, we may conclude that the weight of the respective term was low.

Following this line of thought, the coefficients of the energy terms may be adjusted as follows:

$$\beta_{\text{col},\, i} = g_1(d_{CB}(t;\, i)) \tag{20}$$

$$\beta_{\text{mot},\, i} = g_2(d_M(t;\, i)) \tag{21}$$

where $d_{CB}(t;\, i)$ and $d_M(t;\, i)$ are defined in (29) and (31), respectively. The functions $g_1(.)$ and $g_2(.)$ should be chosen such that, when $d_{CB}(t;\, i)$ or $d_M(t;\, i)$ are low at a certain segment, the corresponding weight should increase. For example, when

(a) (b) (c)

Fig. 7. (a) Frame 127 of the “Bream” sequence. (b) Color-segmentation boundaries. (c) Color-segmentation reliability map. Dark regions indicate unreliable color-segmentation boundaries.

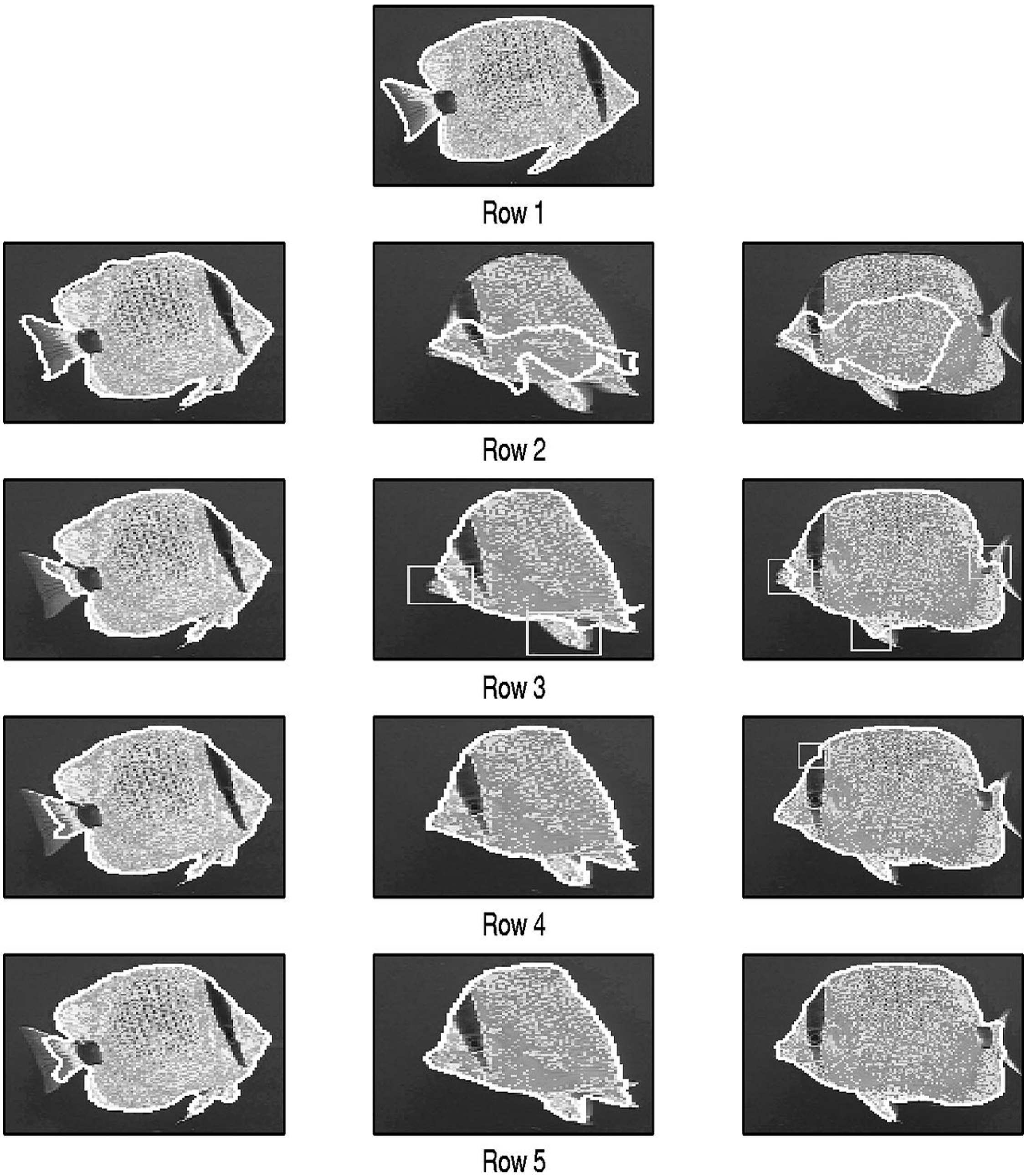


Fig. 8. *Row 1*: hand-drawn boundary at frame 100. *Row 2*: open-loop tracking results for frames 109, 121, and 128. *Row 3*: tracking results after boundary correction using only edge and curvature energy terms with equal weighting. Incorrectly segmented regions are marked with boxes. *Row 4*: tracking results after boundary correction using all the energy terms with equal weighting. *Row 5*: closed-loop tracking results using all the energy terms with adaptive weighting.

the value of $d_{CB}(t; i)$ is low at a certain boundary segment, this indicates that that boundary segment does not coincide with a color boundary, that is, we are far from the color boundary. Then if we increase the weight of the color boundary energy term, we should be forcing the algorithm toward the nearest color boundary faster.

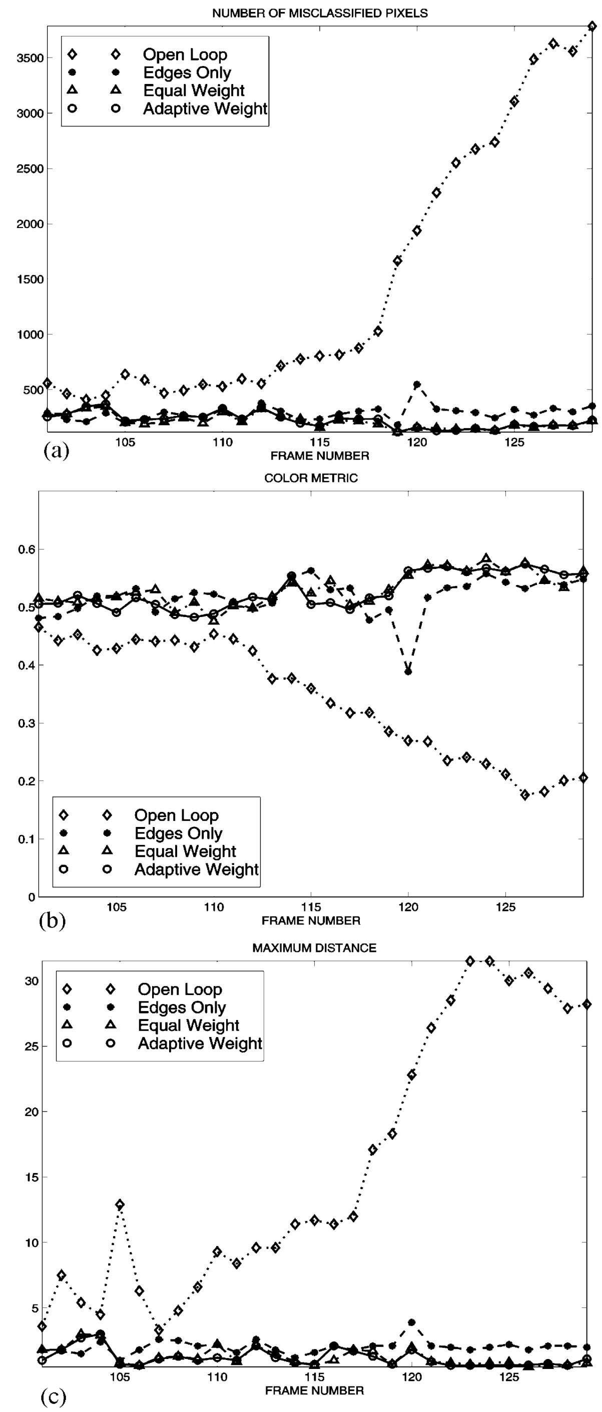


Fig. 9. (a) Number of misclassified pixels for each frame of "Bream" with respect to the ground-truth segmentation maps. (b) Color metric for each frame. (c) Maximum distance for each frame.

We want to distribute the weight budget among the energy terms for each boundary segment i, such that

$$\beta_{\text{mot},\,i} + \beta_{\text{col},\,i} + \beta_{\text{edge},\,i} = 1. \tag{22}$$

Since motion-segmentation boundaries are a subset of the edges and of the color-segmentation boundaries, they are more likely to coincide with actual object boundaries if they exist and are reliable. Therefore, we give the top priority to the motion energy, i.e., if motion-segmentation boundaries are nearby the current segment, we would like to go toward them first. Then, the selection strategy of β_{mot} is as follows:

TABLE II
SUMMARY OF QUANTITATIVE COMPARISONS FOR "BREAM" SEQUENCE

	Average number of misclassified pixels	Average color metric	Average max. pixel distance from actual object boundary
Open Loop	1473	0.340	17.3
Edges Only	287	0.516	2.05
Equal Weight	211	0.530	1.27
Adaptive Weight	207	0.527	1.26

$$\beta_{\text{mot},\,i} = \frac{K_1}{d_M(t;i)+1} \tag{23}$$

where the parameter K_1 is set to zero if the distance of the current segment to the nearest motion-segmentation boundary is above a certain value (e.g., 10 pixels) or if the current segment is in an unreliable motion-segmentation boundary region. Otherwise, K_1 is set to 1. Considering that $d_M(t;\,i)$ takes values between 0 and 1, the values that $\beta_{\text{mot},\,i}$ can take is:

$$\beta_{\text{mot},\,i} = \begin{cases} 0, & K_1 = 0 \\ 0.5 < \beta_{\text{mot},\,i} < 1, & K_1 = 1. \end{cases} \tag{24}$$

Once we determine the motion weight in the above fashion, the weight of the color boundary energy term $\beta_{\text{col},\,i}$ is determined as follows:

$$\beta_{\text{col},\,i} = \frac{K_2(1-\beta_{\text{mot},\,i})}{d_{CB}(t;\,i)+1}, \tag{25}$$

where the parameter K_2 is set to zero if the distance of the current segment to the nearest color-segmentation boundary is above a certain value (e.g., 10 pixels) or if the current segment is in an unreliable color-segmentation boundary region. Otherwise, K_2 is set to 1. Considering that $d_{CB}(t;\,i)$ takes values between 0 and 1, the values that $\beta_{\text{col},\,i}$ can take are

$$\beta_{\text{col},\,i} = \begin{cases} 0, & K_2 = 0 \\ \dfrac{1-\beta_{\text{mot},\,i}}{2} < \beta_{\text{col},\,i} < 1 - \beta_{\text{mot},\,i}, & K_2 = 1. \end{cases} \tag{26}$$

Finally, the weight of the edge energy term is determined as follows:

$$\beta_{\text{edge},\,i} = 1 - \beta_{\text{mot},\,i} - \beta_{\text{col},\,i}. \tag{27}$$

The numerical range of the energy terms $E_{\text{col},\,i}$, $E_{\text{edge},\,i}$, and $E_{\text{mot},\,i}$ are comparable since they are all calculated through the Chamfer distance transformation. If maximum search length is r for a certain node, then the maximum value that each of the energy terms can take is approximately $4r$. The maximum value that the curvature energy term E_{curv} can take is 4 as given in (18). Therefore, the coefficient of the curvature term should be multiplied by the search range.

After the coefficients have been updated, the energy minimization is repeated forming the closed-loop. The algorithm

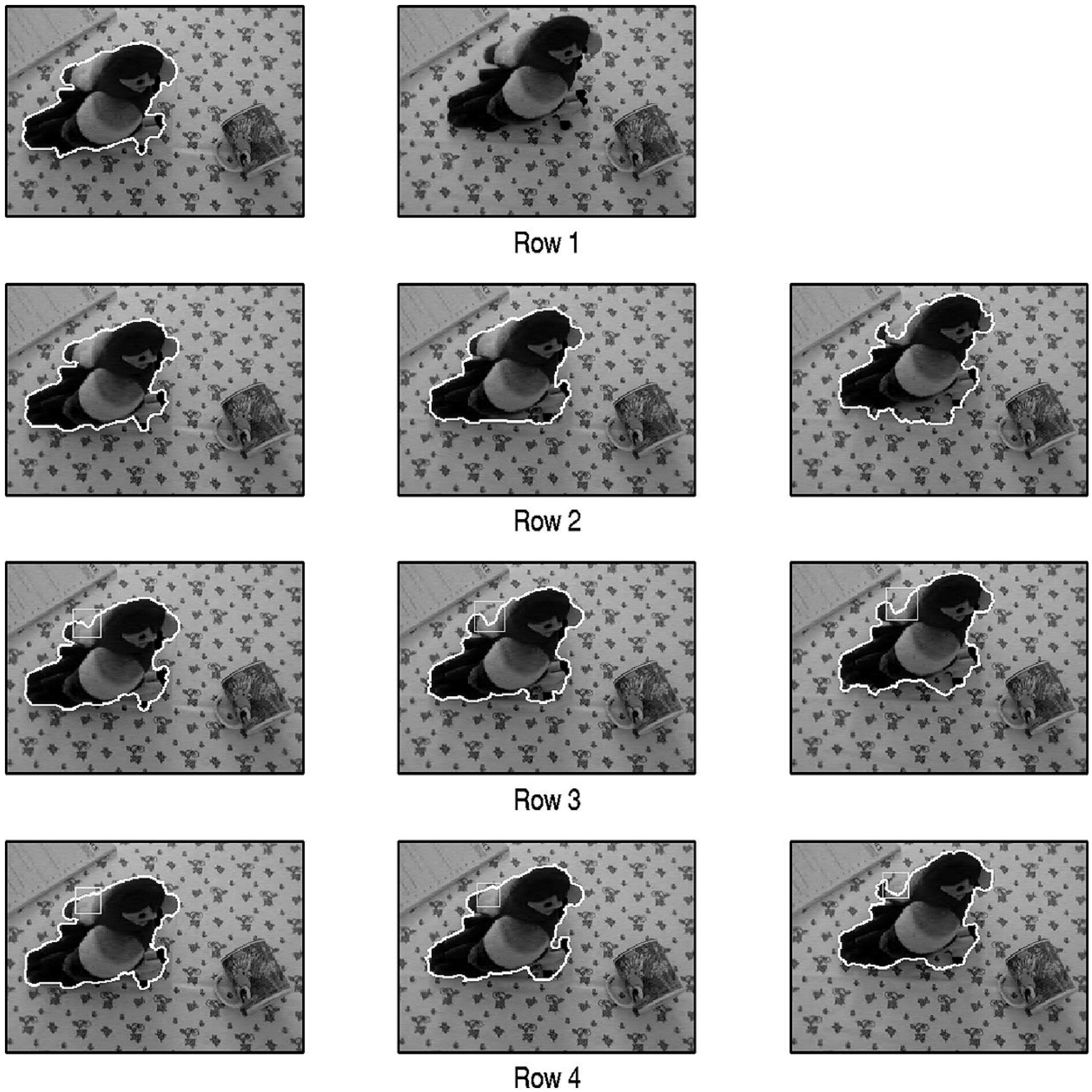


Fig. 10. *Row 1*: The first frame of the "Parrot" sequence with the hand-drawn boundary (left), the 18th frame of the "Parrot" sequence. The amount of motion from frame 1 to frame 18 is approximately (26, -20) pixels in the x and y directions. *Row 2*: Tracking results after boundary correction for frames 2, 8, and 18 using only edge and curvature energy terms with equal weighting. *Row 3*: Tracking results after boundary correction using all the energy terms with equal weighting. *Row 4*: Closed-loop tracking results using all the energy terms with adaptive weighting. Compare the equal weight and adaptive weight results, especially the sections marked with the boxes.

exits from the loop when most of the node points become stationary between iterations or a maximum number of iterations is reached.

VI. Experimental Results

The algorithm is tested on three video sequences containing rigid and nonrigid objects. The sequences used are "Bream," "Parrot," and "Coastguard" sequences.

A. Results for the "Bream" Sequence

The Bream sequence contains a fish swimming and changing direction as shown in Fig. 8. There is significant self-occlusion in this sequence as the fish turns from right to left. In the following experiments, the boundary is determined interactively by the user in frame 100 and tracked automatically afterwards.

In the second row of Fig. 8, the open-loop tracking results are given. That is, the boundary is updated based only on the boundary prediction scheme described in Section III. Although the tracking results are good until frame 110, the predicted object boundary deviates significantly from the actual object boundary after frame 110.

In the third row of Fig. 8, the boundary correction results are given using the edge and the curvature terms only with equal weighting, similar to traditional snake algorithms. The results are good until frame 115, but some deviations from the original boundary are observed as marked with square boxes, especially when the fish starts to turn from right to left.

In the fourth row of Fig. 8, the tracking results after boundary correction are shown with equal weighting of all the three energy term weights. Although the results are much better than the third row, the final boundary in frame 128 tends to snap to the incorrect color-segmentation boundary as was discussed in Fig. 7.

In the fifth row of Fig. 8, the closed-loop tracking results using the proposed adaptive weighting strategy is shown. It can

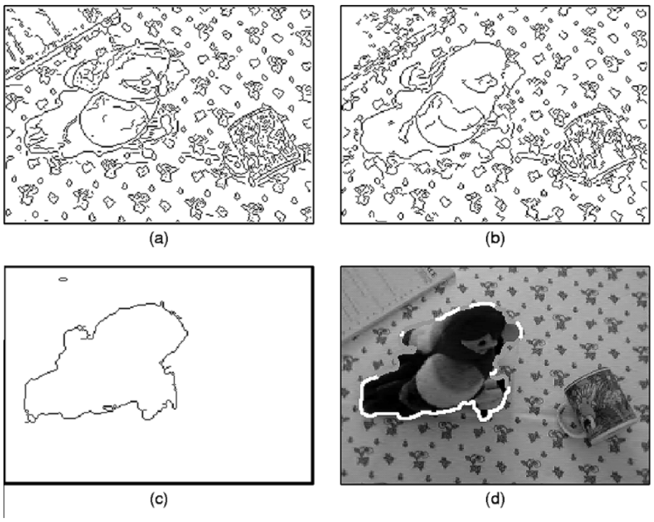


Fig. 11. (a) Edges of the second frame of the "Parrot" sequence. (b) Color-segmentation boundaries. (c) Color-refined motion-segmentation boundaries. (d) Boundary sections drawn with white lines indicate the regions where the weight of the motion energy term is the largest compared to the other terms. The boundary sections drawn with gray lines indicate the regions where the weight of the motion energy is zero.

be observed that the outcome is definitely more satisfactory than the results shown in row 4. The problem at frame 128 is successfully eliminated.

In order to compare the four rows quantitatively, the number of misclassified pixels, the average color metric, and the maximum distance from the actual object boundary for each frame are plotted in Fig. 9. The average values of the plots for frames 101–129 are summarized in Table II. The adaptive weighting strategy gives the least average number of misclassified pixels and the least average maximum distance from the object boundary. As seen in the Table II, the maximum distance between the estimated object boundary and the actual object boundary for the whole sequence is smallest for the last row.

B. Results for the "Parrot" Sequence

Another sequence that is used to test the algorithm is the Parrot sequence, the first and the 18th frames of which are shown in the first row of Fig. 10. The amount of motion from frame 1 to frame 18 is approximately (26, -20) pixels in the x and y directions. The object and the background in this sequence is quite cluttered.

The second row of Fig. 10 shows the results using only edge and curvature terms. The third row shows the results using all the energy terms with equal weighting. The fourth row shows the results using all the energy terms with adaptive weighting. If we compare the equal weight and adaptive weight results, especially the sections marked with boxes, we can see that the adaptive weighting strategy is more successful in tracking the left wing of the Parrot. The equal weighting strategy is affected by the imperfection of the color-refined motion-segmentation boundaries which are shown in Fig. 11(c). However, the adaptive weighting strategy overcomes this problem by making the weight of the motion energy term zero in those regions as shown in Fig. 11(d). In this figure, the boundary sections drawn with gray lines indicate the regions where the color-refines motion-segmentation boundaries are found unreliable and edges are given more weight. The boundary sections drawn with white lines indicate the regions where the color-refined motion-segmentation boundaries are reliable and more weight is given to the motion energy term.

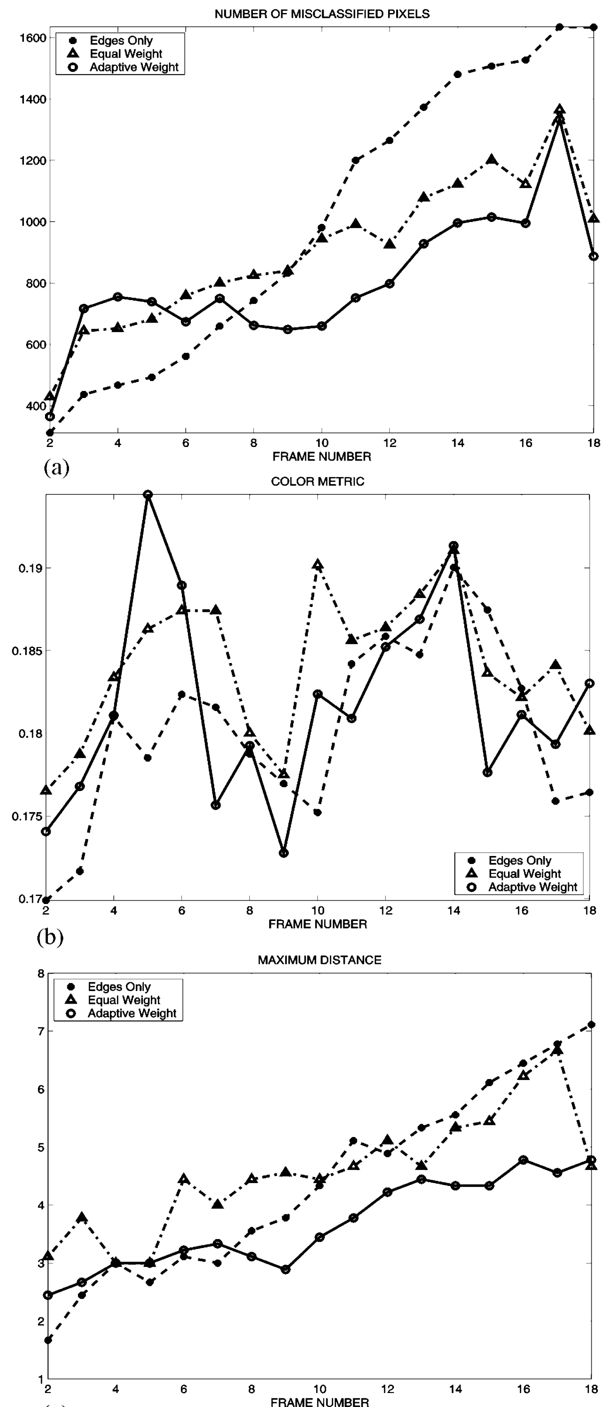


Fig. 12. (a) The number of misclassified pixels for each frame of "Parrot" with respect to the ground-truth segmentation maps. (b) Color metric for each frame. (c) Maximum distance for each frame.

In order to compare the results quantitatively, the number of misclassified pixels, the average color metric, and the maximum distance from the actual object boundary for each frame are

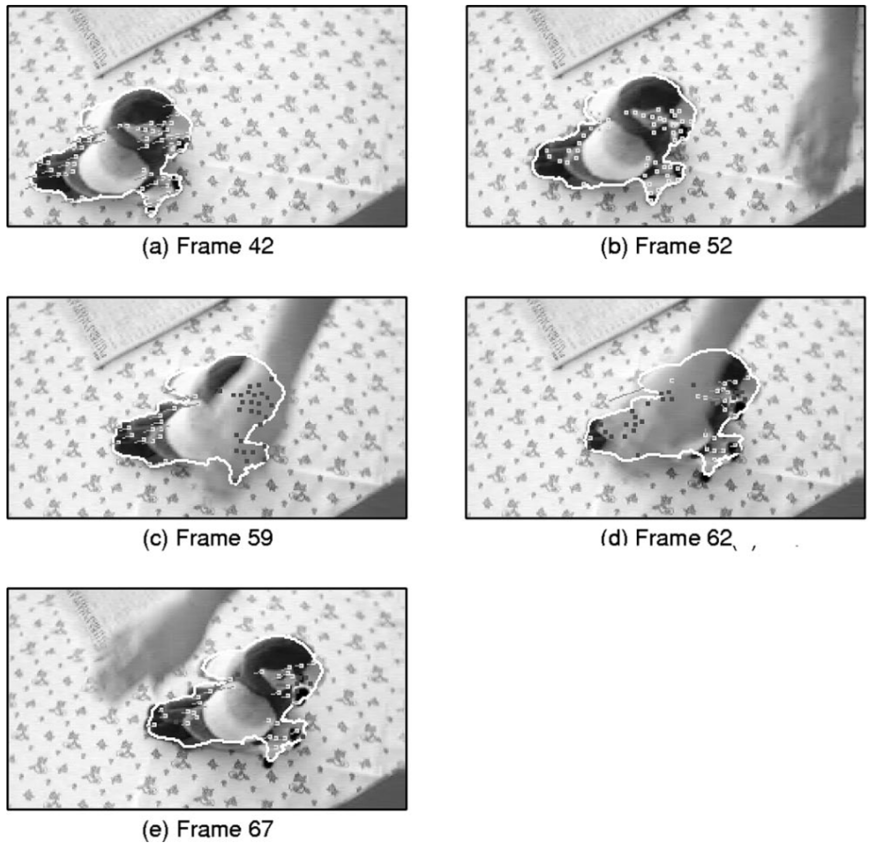


Fig. 13. Tracking results for the Parrot sequence with occlusion. The squares and the lines denote the tracked feature points and their motion vectors. Darker squares denote the predicted feature points.

TABLE III
SUMMARY OF QUANTITATIVE COMPARISONS FOR "PARROT" SEQUENCE

	Average number of misclassified pixels	Average color metric	Average max. pixel distance from actual object boundary
Edges Only	1006	0.180	4.40
Equal Weight	905	0.184	4.56
Adaptive Weight	804	0.181	3.66

plotted in Fig. 12. A quantitative comparison table is also given in Table III. It can be observed that the number of misclassified pixels and the average maximum distance from the actual object boundary is the smallest for the proposed adaptive energy weighting method.

C. Results for the "Parrot" Sequence With Occlusion

In order to illustrate the performance of the algorithm under multiobject occlusion, we have given the tracking results of another experiment, where the Parrot object is occluded significantly by a hand. The open-loop tracking results together with the tracked (light gray) and predicted (dark gray) feature points are given in Fig. 13. The motion vectors of the tracked features are shown with gray lines. The object is restricted to be rigid and affine motion parameters are estimated using all the tracked feature points inside the object. Outlier motion vectors are eliminated before affine parameter estimation. It can be seen that the occlusion handling techniques described in Section III-C are successful in predicting the locations of the occluded features.

D. Results for the "Coastguard" Sequence

Another sequence used in the experiments is the Coastguard sequence. The boat object in this sequence is rigid and the background is quite cluttered. There is also severe blurring when the camera moves suddenly around frame 72, as can be seen in Fig. 14.

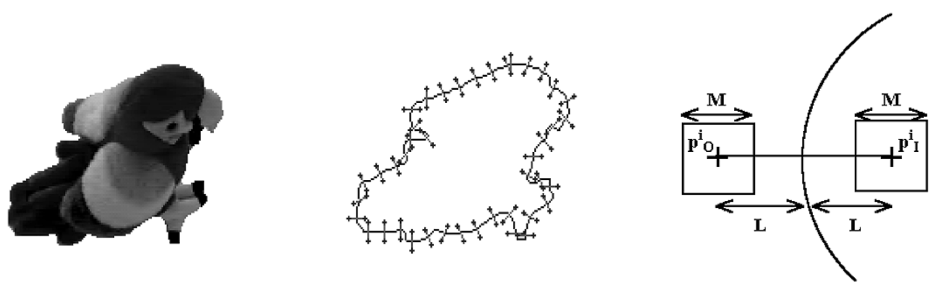
The open-loop tracking results with the imposed rigidity constraint is given in Fig. 14. Feature points which have been successfully tracked at each frame are shown with gray squares and the predicted feature points are shown with dark gray squares. All the feature points inside the segmentation map of the boat are used to estimate the motion of the boundary of the object. The algorithm tracks the boat successfully despite the heavy blurring due to camera movement in frame 72.

For sequences with rigid objects, the boundary prediction block without the boundary correction step can be sufficient, as demonstrated in this example.

VII. CONCLUSIONS

We have presented a novel scalable framework for tracking rigid and nonrigid objects. The approach consists of an initial rough boundary prediction, and if necessary followed by a fine-tuning step. New energy terms derived from color and motion-segmentation boundaries are presented for improving the accuracy of the estimated boundary. The weights of these energy terms are determined adaptively in a feedback loop using the intrinsic performance metrics. The experimental

Fig. 14. Open-loop tracking results of the boat in “Coast Guard.” Close-ups of frames 1, 45, 72, and 86 are shown. Tracked and predicted feature points are shown by light and dark squares, respectively. Motion vectors of the feature points are shown by gray lines.



(a) (b) (c)

Fig. 15. (a) Video object in frame 1 of “Parrot” sequence. (b) Boundary of the video object with the normals. (c) Close-up of a boundary normal. The points “just inside” and “just outside” of the boundary are denoted as p_I^i and p_O^i, respectively.

results demonstrate that the presented framework is quite successful in tracking the boundary of various rigid and nonrigid objects accurately, without using object-specific models or training.

APPENDIX
PERFORMANCE EVALUATION METRICS

In this section we present two intrinsic performance metrics, which uses color and motion features [26].

A. Color Metric Based on Intra-Frame Boundary Color Differences

The first metric is based on the assumption that object boundaries coincide with color boundaries, therefore it compares the color of the pixels “just inside” and “just outside” of the estimated object boundary. In order to define “just inside” and “just outside,” we draw short normal lines of length L to the estimated object boundary at equal intervals toward the inside and outside of the object as illustrated in Fig. 15(b). The points at the ends of these normal lines are marked with plus signs. A closer look at one of these normal lines is given in Fig. 15(c).

We define the color difference metric calculated along the boundary of the object in frame t as

$$0 \leq d_{CB}(t) = 1 - \frac{1}{K_t} \sum_{i=1}^{K_t} d_{CB}(t;\, i) \leq 1 \tag{28}$$

$$d_{CB}(t;\, i) = 1 - \exp\left(-\frac{\|C_O^i(t) - C_I^i(t)\|^2}{\sigma^2}\right) \tag{29}$$

where K_t is the total number of normal lines in frame t, $C_O^i(t)$ is the average color calculated in the $M \times M$ neighborhood of the pixel $p_O^i(x,\, y;\, t)$ using Y–Cb–Cr color space, and σ is chosen around 300, meaning a deviation of ten in all color bands will give us a distance value around 0.6. The average color inside $C_I^i(t)$ is defined similarly.

B. Motion Metric

The assumptions that we make to define a motion metric are as follows.

1) The motion vectors of the object that are "just inside" of the object boundary and the background motion vectors that are "just outside" of the object boundary are different. In other words, object boundaries coincide with the motion boundaries.
2) Background is either stationary or has global motion which shall be compensated for.

In order to quantify how well the estimated object boundaries coincide with actual motion boundaries, we draw short normal lines to the boundary at regular intervals as shown in Fig. 15(b), and we look at the difference of the motion vectors around the points $p_O^i(x, y; t)$ and $p_I^i(x, y; t)$. The motion metric estimated following this approach for frame t can be expressed as:

$$0 \leq d_M(t) = 1 - \frac{\sum_{i=1}^{K_t} d_M(t; i)}{\sum_{i=1}^{K_t} w_i} \leq 1 \tag{30}$$

$$d_M(t; i) = d\left(v_O^i(t), v_I^i(t)\right) \cdot w_i \tag{31}$$

$$0 \leq w_i = R\left(v_O^i(t)\right) \cdot R\left(v_I^i(t)\right) \leq 1 \tag{32}$$

where $v_O^i(t)$ and $v_I^i(t)$ denote the average motion vectors calculated in a $M \times M$ square around the points $p_O^i(x, y; t)$ and $p_I^i(x, y; t)$, respectively, and $d(v_O^i(t), v_I^i(t))$ denotes the distance between the two average motion vectors which is calculated as

$$\begin{aligned} 0 &\leq d\left(v_O^i(t), v_I^i(t)\right) \\ &= 1 - \exp\left(-\frac{\|v_O^i(t) - v_I^i(t)\|^2}{\sigma^2}\right) \leq 1 \end{aligned} \tag{33}$$

where the parameter σ is chosen as 1, which means if the magnitude of the difference vectors is 1, the value of the motion distance will be 0.6.

In (30), $R(.)$ denotes the reliability of the motion vector $f^i(t)$ at point p^i [12]

$$R\left(v^i(t)\right) = \exp\left(-\frac{\|v^i(t) - b^i(t+1)\|^2}{2\sigma_m^2}\right) \cdot \exp\left(-\frac{\|c(p^i; t) - c(p^i + v^i; t+1)\|^2}{2\sigma_c^2}\right)$$

where $b^i(t+1)$ denotes the backward motion vector at location $p^i + v^i$ in frame $t+1$; $c(p^i; t)$ denotes the color intensity and the parameters σ_m, σ_c are chosen freely.

REFERENCES

[1] M. Irani and S. Peleg, "Motion analysis for image enhancement: resolution, occlusion, and transparency," *J. Vis. Commun. Image Repres.*, vol. 4, no. 4, pp. 324–335, 1993.

[2] T. Cham and J. M. Regh, "A multiple hypothesis approach to figure tracking," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 1999, pp. 239–245.

[3] S. X. Ju, M. J. Black, and Y. Yacoob, "Cardboard people: A parametrized model of articulated image motion," in *Proc. Int. Conf. Face and Gesture Recognition*, 1996, pp. 561–567.

[4] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Santa Barbara, CA, June 1998, pp. 239–245.

[5] S. Wachter and H. H. Nagel, "Tracking persons in monocular image sequences," *Comput. Vis. Image Understand.*, vol. 74, no. 3, pp. 174–192, June 1999.

[6] A. Blake and M. Isard, *Active Contours*: Springer-Verlag, 1998.

[7] A. M. Baumberg, "Learning deformable models for tracking human motion," Ph.D. dissertation, School of Comput. Studies, Univ. of Leeds, Leeds, U.K., Oct. 1995.

[8] M. Isard and A. Blake, "Condensation—Conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5–28, 1998.

[9] Y. Yacoob and L. S. Davis, "Learned models for estimation of rigid and articulated human motion from stationary or moving camera," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 5–30, 2000.

[10] C. R. Wren, A. Azerbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 780–785, July 1997.

[11] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of nonrigid objects using mean shift," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, vol. 2, Hilton Head Island, SC, June 2000, pp. 142–149.

[12] Y. Fu, A. T. Erdem, and A. M. Tekalp, "Tracking visible boundary of objects using occlusion adaptive motion snake," *IEEE Trans. Image Processing*, vol. 9, pp. 2051–2060, Dec. 2000.

[13] Y. Chen, Y. Rui, and T. S. Huang, "JPDAF based HMM for real-time contour tracking," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2001, pp. I-543–550.

[14] I. Haritaoğlu, D. Harwood, and L. S. Davis, "W-4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 809–830, Aug. 2000.

[15] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 679–698, June 1986.

[16] Y. Altunbaşak, P. E. Eren, and A. M. Tekalp, "Region-based parametric motion segmentation using color information," *Graph. Models and Image Processing*, vol. 60, no. 1, pp. 13–23, 1998.

[17] E. N. Mortensen and W. A. Barrett, "Interactive segmentation with intelligent scissors," *Graph. Models and Image Processing*, vol. 60, pp. 349–384, 1998.

[18] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[19] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.

[20] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 855–867, Sept. 1990.

[21] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. New York: Wiley, 1999.

[22] J. Zhang and J. W. Modestino, "Model-fitting approach to cluster validation with application to stochastic model-based image segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 1009–1017, Oct. 1990.

[23] G. Borgefors, "Distance transformations in digital images," *Comput. Vis., Graph., Image Processing*, vol. 34, pp. 344–371, 1986.

[24] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43–77, 1994.

[25] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. DARPA Image Understanding Workshop*, 1981, pp. 121–130.

[26] C. E. Erdem, A. M. Tekalp, and B. Sankur, "Metrics for performance evaluation of video object segmentation and tracking without ground-truth," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, vol. 2, Thessaloniki, Greece, Oct. 7–10, 2001, pp. 69–72.

Çiğdem Eroğlu Erdem (S'93–M'03) received the B.S. and M.S. degrees from the Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey, in 1995 and 1997, respectively and the Ph.D. degree from the Department of Electrical and Electronics Engineering, Boğaziçi University, Istanbul, Turkey, in 2002.

She was a Research and Teaching Assistant in the Department of Electrical and Electronics Engineering, Bilkent University, during 1995–1997, and then with the Department of Electrical and Electronics Engineering, Boğaziçi University, during 1998–2002. From September 2000 to June 2001, she was a Visiting Researcher in the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY. From July 2002 to February 2003, she was a Senior Researcher at Momentum Digital Media Technologies Inc., Istanbul, Turkey. She is currently a Postdoctoral Fellow at the Information and Communication Theory (ICT) Group of the Faculty of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands. She is also affiliated with the Philips Research Laboratories, Eindhoven, The Netherlands. Her research interests are in the area of digital image, video, and speech processing, including motion estimation, video segmentation, object tracking, 3-D scene reconstruction, and speech processing for natural lip animation.

A. Murat Tekalp (S'80–M'84–SM'91–F'03) received the M.S. and Ph.D. degrees in electrical, computer, and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1982 and 1984, respectively.

From December 1984 to August 1987, he was with Eastman Kodak Company, Rochester, NY. He joined the Electrical and Computer Engineering Department, University of Rochester, Rochester, NY, in September 1987, where he is currently a Distinguished Professor. Since June 2001, he has also been with Koc University, Istanbul, Turkey. He authored the book *Digital Video Processing* (Englewood Cliffs, NJ: Prentice-Hall, 1995) and holds five U.S. patents. His group contributed technology to the ISO/IEC MPEG-4 and MPEG-7 standards. His research interests are in the area of digital image and video processing, including video compression and streaming, video filtering for high-resolution video segmentation, object tracking, content-based video analysis, and summarization, multicamera surveillance video processing, and protection of digital content.

Dr. Tekalp was an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING (1990–1992), the IEEE TRANSACTIONS ON IMAGE PROCESSING (1994–1996), and the *Journal Multidimensional Systems and Signal Processing* (1994–1999). He was an Area Editor for the *Graphical Models and Image Processing* (1995–1998). He was also on the Editorial Board of the *Visual Communication and Image Representation* (1995–1999). Currently, he is the Editor-in-Chief of the EURASIP journal *Image Communication*. He was appointed as the Technical Program Chair for the 1991 IEEE Signal Processing Society Workshop on Image and Multidimensional Signal Processing, the Special Sessions Chair for the 1995 IEEE International Conference on Image Processing, the Technical Program Co-Chair for IEEE ICASSP 2000 in Istanbul, Turkey, and the General Chair of IEEE International Conference on Image Processing (ICIP) at Rochester, NY in 2002. He is the Founder and first Chairman of the Rochester Chapter of the IEEE Signal Processing Society. He was elected as the Chair of the Rochester Section of IEEE in 1994–1995. He received the National Science Foundation Research Initiation Award in 1988, was named Distinguished Lecturer by IEEE Signal Processing Society in 1998, and was awarded a Fulbright Senior Scholarship in 1999. He has chaired the IEEE Signal Processing Society Technical Committee on Image and Multidimensional Signal Processing (January 1996–December 1997).

Bülent Sankur received the B.S. degree in electrical engineering from Robert College, Istanbul, Turkey, and the M.Sc. and Ph.D. degrees from Rensselaer Polytechnic Institute, Troy, NY.

He has been active in the Department of Electric and Electronic Engineering, Boğaziçi University, Istanbul, Turkey, in establishing curricula and guiding research in the areas of digital signal processing, image and video compression, and multimedia systems. He has held visiting positions at the University of Ottawa, ON, Canada, Istanbul Technical University, Istanbul, Turkey, the Technical University of Delft, Delft, The Netherlands, and Ecole Nationale Superieure des Telecommunications, Paris, France.