Model-Based Segmentation and Tracking of Head-and-Shoulder Video Objects for Real Time Multimedia Services

Huitao Luo, *Member, IEEE,* and Alexandros Eleftheriadis, *Member, IEEE*

***Abstract*—A statistical model-based video segmentation algorithm is presented for head-and-shoulder type video. This algorithm uses domain knowledge by abstracting the head-and-shoulder object with a blob-based statistical region model and a shape model. The object segmentation problem is then converted into a model detection and tracking problem. At the system level, a hierarchical structure is designed and spatial and temporal filters are used to improve segmentation quality. This algorithm runs in real time over a QCIF size video, and segments it into background, head and shoulder three video objects on average Pentium PC platforms. Due to its real time feature, this algorithm is appropriate for real time multimedia services such as videophone and web chat. Simulation results are offered to compare MPEG-4 performance with H.263 on segmented video objects with respects to compression efficiency, bit rate adaptation and functionality.**

***Index Terms*—Low bit rate coding, MPEG-4, statistical modeling, video object segmentation.**

I. Introduction

OBJECT segmentation from image and video is a recent popular topic in multimedia research because of common interests in better object-based functionalities. MPEG-4 offers a general framework for object-based system but leaves the segmentation design as an open issue. In the literature, large numbers of segmentation algorithms, both fully automatic and semi-automatic, have been published. Among them, activities on automatic segmentation have mainly focused on motion detection [1]–[6], motion segmentation [7]–[10], or joint motion and (chromatic) region segmentation [11], [12]. The basic assumption is that the *object* can be described as either the *moving* region or the region that moves with *consistent parameters*. In contrast, typical semi-automatic algorithms utilize certain user interactions to specify object on one or several anchoring frames, and tracking algorithms are then designed to track the object in the temporal direction. For example, Chalom and Bove [13] used multiple features to model pixels and to track objects. Marcotegui *et al.* [14], and Zhong and Chang [15] proposed to segment the initial object into regions and then track each of them across frames; Malassiotis and Strintzis [16], and Toklu and Tekalp [17] used active mesh to represent and track objects; while Gu [18], Fu [19], and Luo [20] proposed to track object boundaries based on neighboring color and motion information.

From an application point of view, the available segmentation algorithms can also be classified into those for online applications and those for offline applications, as proposed by Correia and Pereira [21]. Though many multimedia applications are online and real-time (to name a few of them: video surveillance, videophone, web chat, human computer interface, etc.), most reported algorithms are useful only for offline applications. Among the algorithms we reviewed so far, most of the semi-automatic algorithms and motion segmentation algorithms are not good for offline applications because of slow speed and/or requirements on user interaction. Motion detection based algorithms are more appropriate for online applications because of their "moving foreground" and "static background" assumption, which lead to less intensive computation requirements. However, at the same time, this assumption limits the generality of the algorithms, i.e., when the tracked object stops moving, these algorithms fail. In other words, these algorithms are valid only if the assumption or the domain knowledge is valid.

We propose a real-time algorithm to segment head-and-shoulder type video sequences. The motivation stems from popular presence of head-and-shoulder type video signal in real-time services such as videophone and web chatting, etc., which form the application domain of our work. By focusing to this specific application domain, the segmentation algorithm could be designed to exploit domain knowledge and realize real-time performance with less computation complexity. In our work, statistical models are used to model the domain knowledge. We assume that in a typical setup of a head-and-shoulder video, the foreground object is one person in a head-and-shoulder pattern and the background is relatively static. Our algorithm aims to segments the input video into three video objects (VO)s: a background, a head/face and a shoulder. The domain knowledge is modeled from two aspects. One is a blob modeling of each VO region's color and spatial distribution, and the other is shape modeling. With the concept of statistical model, the segmentation and tracking problem is turned into model parameter fitting and updating. In addition, fast spatial and temporal filters are designed to generate video objects with refined boundaries, and a hierarchical structure is used at the system level to coordinate the processing. We implemented the algorithm and had it run in real time over a Pentium PC with off-the-shelf cameras. Within its application domain, this algorithm performed much better than motion detection algorithms we tested.

Manuscript received December 13, 1999; revised July 18, 2002. The associate editor coordinating the review of this paper and approving it for publication was Prof. Yao Wang.

H. Luo is with Hewlett-Packard Laboratories, Palo Alto, CA 94304 USA (e-mail:huitao@exch.hpl.hp.com).

A. Eleftheriadis is with Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: eleft@ee.columbia.edu).

Digital Object Identifier 10.1109/TMM.2003.813285

In principle, this work is related to the widely used "chroma-key" technique and human body tracking work of "Pfinder" [22]. It is different from chroma-key in its modeling of foreground and background. Because of its statistical models, our system has more stable performance than chroma-key over inexpensive consumer CCD cameras. In addition, the model updating and tracking mechanism enables our system to tolerate moderate background changes that generally fail a typical chroma-key system. Our system is similar to "Pfinder" in the utilization of blob model as a means to track a region with respect to its chromatic and spatial distributions. However, our contributions are in anchoring blob tracking with results from shape tracking. This way, more stable tracking results can be realized over longer time. Moreover, our system is more concentrated on generating accurate object support, while "Pfinder" emphasizes more on tracking and understanding the semantic meaning of human motions.

The structure of this paper is as follows. First in Section II, we discuss blob-based region modeling and Kalman filters for blob tracking. Section III covers shape modeling. Section IV discusses a hierarchical system structure. Segmentation experiments are presented in Section V. In Section VI, we discuss the application of segmentation to real time videophone services. The paper is concluded in Section VII.

II. Blob Based Region Modeling and Tracking

The basic nature of the algorithm is an online one. First, we assume a background scene that contains no foreground, which enables the creation of a background model. When the foreground enters, another model is then created for the foreground.

A. Foreground Model

In our work, the foreground is considered a "head-and-shoulder," which is modeled with two connected "blob"s. Here the definition of blob is similar to that in [22], i.e., each blob has a spatial (x, y) and chromatic (Y, U, V) Gaussian distribution, and a support map that indicates whether a pixel is a member of the blob. In this model, each pixel is represented by a feature vector (x, y, Y, U, V). The feature vectors of the pixels belonging to blob k have a Gaussian distribution with mean vector $\mathbf{m}_k$ and covariance matrix $\mathbf{C}_k$. Because of their different semantics, the spatial and chromatic distributions are assumed independent, i.e., the matrix $\mathbf{C}_k$ is assumed block-diagonal.

For convenience of discussion, some definitions related to blob modeling are defined as follows. First, the support map $s_k(x, y)$ for blob $k, k = 1, 2, 3, \ldots$, is defined as

$$s_k(x,y) = \begin{cases} 1, & \text{if pixel } (x,y) \text{ is in blob } k, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Based on $s_k(x, y)$, blob k's *containing rectangle* rect_k is defined as the minimal rectangle that covers the blob pixels. For segmentation purposes we also define a cumulative support map $s(x, y)$ for each image as

$$s(x,y) = \begin{cases} k, & \text{if } s_k(x,y) = 1, \quad k = 1, 2, 3, \ldots, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

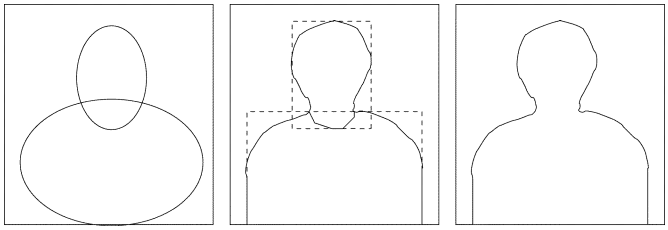


Fig. 1. (Left) Blob representation; (middle) support map with containing rectangles; (right) foreground map.

Note the blob labeling as defined in (1) is exclusive, i.e., each pixel belongs to only one blob. In addition, we define the entire set of the support maps of foreground blobs as the foreground map $f(x, y)$:

$$f(x,y) = \begin{cases} 1, & s(x,y) \neq 0, \\ 0, & s(x,y) = 0. \end{cases} \tag{3}$$

The relation between these concepts can be better illustrated in Fig. 1, where the left image illustrates two blobs, the middle image shows a support map with containing rectangles, and the right image is a foreground map.

The rationale behind blob modeling is that it represents an image region that has chromatic and spatial similarity. By the definition of a blob and its associated support map, low-level, pixel oriented segmentation problem are associated with a high-level, semantically meaningful blob tracking. This way, high-level a priori knowledge can be used to guide pixel segmentation.

B. Background Model

The background is modeled as a texture map varying over time. In videophone applications, we assume the camera is static and there are no fast background changes. In this context, there are still several sources that may introduce temporal color variations in background pixels and thus influence an accurate modeling. These include the thermal noise of the sensor, the AGC (automatic gain control) function of the camera, the shading effect from foreground motions, etc.

Given all these factors, each background pixel is modeled as a Gaussian distribution in a normalized chromatic vector space (U^*, V^*), where $U^* = U/(Y + c)$ and $V^* = V/(Y + c)$ (c is a small constant). We denote the mean vector and covariance matrix as $\mathbf{m}_0$ and $\mathbf{C}_0$ respectively. In the segmentation loop, the model parameter $\mathbf{m}_0$ is created and updated in two steps. First, a frame level global shifting factor is estimated as

$$\mathbf{diff}_t = \frac{\sum_{f(x,y)=1} (\hat{\mathbf{y}}_t(x,y) - \hat{\mathbf{y}}_{(t-1)}(x,y))}{\sum_{(x,y)\in\mathcal{I}} f(x,y)}. \tag{4}$$

After that, the mean vector for each pixel is then updated by considering both local and global factors:

$$\mathbf{m}'_{0,t}(x,y) = \mathbf{diff}_t + \mathbf{m}_{0,t-1}(x,y), (x,y) \in \mathcal{I}, \tag{5}$$

$$\mathbf{m}_{0,t}(x,y) = \alpha * \hat{\mathbf{y}}_t(x,y) + (1-\alpha) * \mathbf{m}'_{0,t}(x,y). \tag{6}$$

Note in (4)– (6), $\hat{\mathbf{y}}_t(x, y)$ and $\mathbf{m}_{0,t}(x, y)$ are the observed feature vector and model parameters for the current background pixel at the spatial position (x, y) and temporal position t (in

succeeding references, $\hat{\mathbf{y}}_t, \mathbf{m}_{0,t}$ or $\hat{\mathbf{y}}, \mathbf{m}_0$ may be used when (x, y) and/or t information are not important). α is a weighting factor with $(0 \leq \alpha \leq 1)$. $\mathbf{diff}_t$ defined in (4) is mainly used to model chromatic changes introduced by camera AGC, which generally influences every pixels in the frame. Accordingly, (5) and (6) carry out global and local updating for each pixel model respectively. According to our experiments, covariance matrix $\mathbf{C}_0$ is relatively stable in tracking. Therefore $\mathbf{C}_0$ is only updated when large shift occurs in (5).

Unlike the foreground model, each background pixel is modeled individually. To express in a uniform way, the feature vectors of the background model can also be put into the vector space (x, y, U^*, V^*) by implicitly including the spatial coordinate of each pixel (x, y). This approach can accommodate a variety of complex backgrounds without limiting them to a structure. At the same time, this model is better than chroma-key because it accommodates temporal changes.

C. Region Classification

Given the foreground and background models, *maximum a posteriori probability* (MAP) principle is used to classify pixels into different regions. We have two foreground classes (*shoulder* and *head*, $k = 1, 2$) and one background class $(k = 0)$. To compensate the shading effect, we choose to use feature vector $\hat{\mathbf{y}} = (x, y, U^*, V^*)$ instead of (x, y, Y, U, V) for foreground blobs as well (the major modeling principle remains the same). So the log likelihood is expressed as

$$\ln(p(\hat{\mathbf{y}} \mid \Omega_k)) = -(\hat{\mathbf{y}} - \mathbf{m}_k)^T \mathbf{C}_k^{-1} (\hat{\mathbf{y}} - \mathbf{m}_k) - \ln(\det(\mathbf{C}_k)) \tag{7}$$

where $(k = 0, 1, 2)$, and Ω_k represents the event that the pixel belongs to class k. Based on Bayes principle, each pixel is labeled in the support map as

$$\begin{aligned} s(x, y) &= \arg\max_k (\ln(p(\Omega_k \mid \hat{\mathbf{y}}))) \\ &= \arg\max_k [\ln(p(\hat{\mathbf{y}} \mid \Omega_k) + \ln(p(\Omega_k))] \end{aligned} \tag{8}$$

where $\ln(p(\Omega_k)))$ is estimated from typical videophone pictures.

To convert the classified pixels into meaningful regions, two steps come next. First the foreground pixels are processed with morphological filters (close operation with 3×3 structuring element) to create a simple connected foreground map $f(x, y)$. Second the support map $s(x, y)$ is obtained by blob growing, i.e., each blob is grown out within the foreground map from their blob center to create a simple connected support map. This is illustrated in Fig. 2.

D. Blob Tracking Procedure

We discuss the basic tracking procedure in two loops: model initialization loop and tracking loop.

1) Initialization Loop: Initialization loop is to detect head-and-shoulder type foreground and to create the foreground and background models. Its logic steps are illustrated in Fig. 3. At the beginning, the background only scene is captured and a background model is created. When a foreground enters, the system detects model deviation and tries to analyze the size, speed and shape of the possible foreground and judge the likelihood of being a head-and-shoulder foreground. When a valid foreground is detected, a foreground model is created and the system enters the tracking loop.

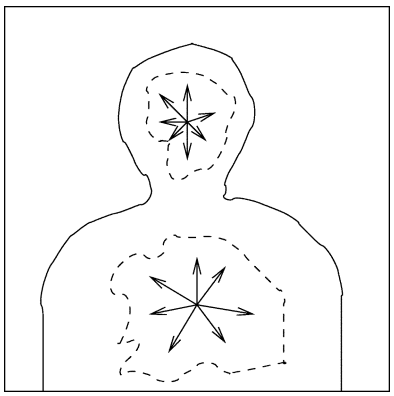


Fig. 2. Blob region growing illustration. Each blob's support map $s_k(x, y)$ is grown out from their blob centers on top of the foreground map $f(x, y)$.

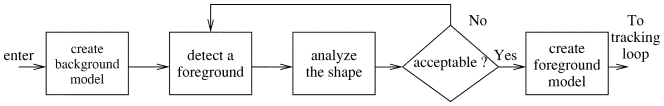


Fig. 3. Flowchart of the initialization loop.

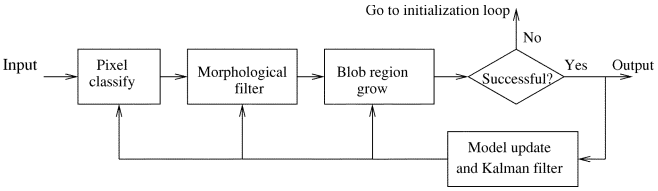


Fig. 4. Flowchart of the basic tracking loop procedure.

2) Tracking Loop: The flowchart of tracking loop is shown in Fig. 4. The major steps of region segmentation are pixel classification, foreground morphological filtering and blob region growing. Unlike low-level pixel classification, blob tracking is carried out at the model-level with semantic meanings. Though in this work, the concern of segmentation is to get an accurate support map for each blob rather than to track the semantic information of blob motion, good modeling and tracking of blobs are still important because the tracked blobs carry the statistical model parameters, which are important for support map segmentation.

Each blob is tracked independently using Kalman filter. The observation vector for blob k includes the blob's center (x_k, y_k) and blob's statistical width and height (w_k, h_k):

$$\hat{\mathbf{Y}}_k = (x_k, y_k, w_k, h_k) \tag{9}$$

where w_k and h_k are defined as $w_k = 2\sigma_{x_k}$ and $h_k = 2\sigma_{y_k}$, and obtained from the feature vector covariance matrix $\mathbf{C}_k$. The dynamic model is a discrete Newtonian physical model of rigid body motion, which has the form

$$\hat{\mathbf{X}}(t + \Delta t) = \mathbf{\Psi}(\Delta t)\hat{\mathbf{X}}(t) + \boldsymbol{\xi}(t) \tag{10}$$

where $\hat{\mathbf{X}}$ is the state vector, $\mathbf{\Psi}$ is the state transition matrix and $\boldsymbol{\xi}$ is the noise term. The state vector $\hat{\mathbf{X}}$ and noise vector $\boldsymbol{\xi}$ each

contain four variables for the position of observation vector $\hat{\mathbf{Y}}$, four for the velocity and four for the acceleration, i.e.,

$$\hat{\mathbf{X}}(t) = \begin{pmatrix} \hat{Y} \\ V \\ A \end{pmatrix}, \quad \text{and} \quad \boldsymbol{\xi}(t) = \begin{pmatrix} \boldsymbol{\xi}_{\hat{Y}} \\ \boldsymbol{\xi}_{V} \\ \boldsymbol{\xi}_{A} \end{pmatrix}. \tag{11}$$

In other words, both $\hat{\mathbf{X}}$ and $\boldsymbol{\xi}$ contain 12 variables. From Newtonian physics, we have

$$\boldsymbol{\Psi}(\Delta t) = \begin{pmatrix} I & I\Delta t & 0 \\ 0 & I & \frac{1}{2}I(\Delta t)^2 \\ 0 & 0 & I \end{pmatrix} \tag{12}$$

where I is a 12×12 unit matrix.

In practice, the Kalman filter is used to predict the model parameters of each blob in the next frame, which is the start point of region classification discussed in Section II-C. In return, the result of region classification in current frame is used to update the blob model parameters $\mathbf{m}_k, \mathbf{C}_k, (k = 1, 2)$, and background model parameters $\mathbf{m}_0, \mathbf{C}_0$. It also serves the observation input that drives the Kalman filter for the next prediction. Sometimes when the foreground moves too fast for the filter to follow or just moves out of the scene, the system cannot find appropriate support maps at the predicted positions of the current frame. In these cases, as indicated in Fig. 4, the system automatically changes its status back to the initialization loop.

III. Shape Modeling and Recognition

In the previous sections, a blob based region modeling and tracking approach were discussed. Blobs incorporate domain knowledge quite naturally into the segmentation problem. However, this approach is mainly based on the chromatic cues, which is dependent on the camera's quality and the color contrast between the foreground and background. For real time tracking purposes, it is also likely that errors get accumulated during the course. In order to stabilize the blob tracking process, the system should be adjusted from time to time. In our work, shape cues are used to meet the requirement.

A basic intuition to support this idea is that people can sometimes identify the head region and the shoulder region only based on the object silhouette. However, because the foreground is always in motion and its silhouette keeps changing, it is not in every case that we can analyze the silhouette successfully for segmentation purpose. For example, in Fig. 5 we have two foreground shapes to be analyzed. In the left image, the shape feature is strong and the head region and shoulder region can be easily separated based on the foreground shape information, while in the right image, the shape feature is not so obvious. Therefore, shape analysis only works as an adjustment or auxiliary approach to previous region based approach.

To solve this problem, a *shape recognition* module is added into the shape analysis procedure. We define those shapes with strong shape features as *canonical head-and-shoulder shape*. Those shapes outside this category are named *noncanonical shape*. If a foreground shape belongs to the canonical category, we are sure that we can locate the head region and shoulder region with a high reliability only based on shape information, then the head blob and shoulder blob are located with shape cues. Otherwise the blobs are tracked with chromatic cues as discussed in Section II. This way, we can get the most out of the domain knowledge to stabilize the tracking algorithm.

Fig. 5. Comparison of the limits of shape analysis for blob tracking.

A. Shape Modeling

Though a quantitative definition of canonical or noncanonical category is hard to give and may depend on specific algorithm of shape analysis, inclusiveness is not the requirement of this work. Rather, because the purpose of shape analysis is to stabilize the region-based blob tracking, we can define the canonical shape category as a limited size. That is, only those with strong shape features are included such that if a shape is accepted as a canonical one, then the segmentation output based on the shape analysis is highly reliable. In this sense, we model the canonical shape category as a high dimensional Gaussian distribution and create the model by statistical learning.

1) Fast Vectorization: First, a vectorization algorithm is used to convert a shape into a feature vector. A fast algorithm is designed to fit in with real time applications. It is illustrated in Fig. 6(a): the foreground region is divided uniformly into N stripes in the vertical direction and the horizontal center and the width of each stripe are measured to form a $2N$ dimension feature vector $\mathbf{v}$.

Suppose the input of the algorithm is a segmented foreground picture f with each background pixel $f(x, y) = 0$ and foreground pixel $f(x, y) = 1$. The origin of the image coordinates is at the lower left corner of the image. Then the detailed algorithm can be expressed with the following pseudocode.

```
(1)  topRow = sup_{f(x,y)=1}(y);
(2)  unit = (topRow)/N;
(3)  k = 0;
(4)  for(row = unit/2; row < topRow; row+ = unit){
(5)     start(k) = inf_{f(x,row)=1}(x);
(6)     end(k) = sup_{f(x,row)=1}(x);
(7)     k++;
(8)  }
(9)  center0 = 0;
(10) for(i = 0; i < N; i++){
(11)    width(i) = end(i) − start(i);
(12)    center(i) = (end(i) + start(i))/2;
(13)    center0+ = center(i);
(14) }
(15) center0/ = N;
(16) k = 0;
(17) for(i = 0; i < N; i++){
(18)   v(k++) = width(i)/unit;
(19)   v(k++) = (center(i) − center0)/unit;
(20) }.
```

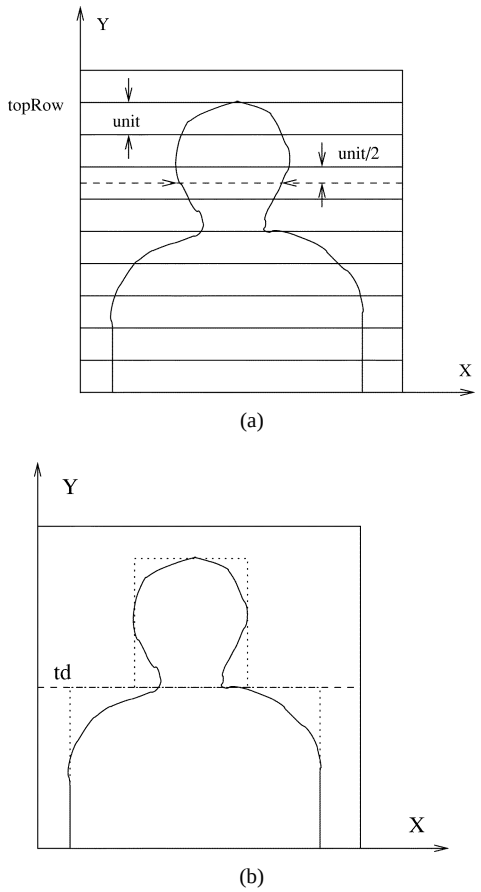


Fig. 6. (a) Stripe based shape vectorization illustration. (b) Illustration of shape-assisted blob region location.

In above pseudocode, (1)–(2) find the width (unit) of sampling stripes, (3)–(8) actually do the sampling, (9)–(14) convert the sampled position data ($\mathrm{start}(i)$ and $\mathrm{end}(i)$) into width and center data, and, finally in (15)–(20), the sampled data is further normalized with respect to size (dividing by unit) and position ($\mathrm{center}(i)$ being measured with respect to center0, which is obtained by averaging the sampled data), producing the feature vector $v(i)$, which represents only the shape information.

This algorithm is actually a controlled polygon approximation of the original shape. The Euclidean distance of vector $\mathbf{v}$ is a good indication of the shape difference because of the polygon approximation nature. Though the shape vector is not rotational invariant, for most real time videophone applications it is not necessary to recognize rotational invariant shapes[1]. The accuracy of the approximation depends on the choice of N and the complexity of foreground shape. According to our observation, $N = 15$ is big enough for most videophone applications in QCIF size.

2) Gaussian Modeling: With above vectorization algorithm, the canonical shape category Ω is modeled as a $2N$ dimension unimodal Gaussian distribution characterized by a mean vector $\bar{\mathbf{v}}$ and a covariance matrix Σ. The likelihood of a shape vector $\mathbf{v} = \bar{\mathbf{v}} + \tilde{\mathbf{v}}$ is given by

$$P(\mathbf{v}\,|\,\Omega) = \frac{\exp\left(-\frac{1}{2}\tilde{\mathbf{v}}^T\boldsymbol{\Sigma}^{-1}\tilde{\mathbf{v}}\right)}{(2\pi)^N \det(\boldsymbol{\Sigma})^{\mathbf{1/2}}}. \tag{13}$$

A sufficient statistic for characterizing the likelihood is the Mahalanobis distance:

$$D = \tilde{\mathbf{v}}^T\boldsymbol{\Sigma}^{-1}\tilde{\mathbf{v}}. \tag{14}$$

In practice, the mean $\bar{\mathbf{v}}$ and covariance $\boldsymbol{\Sigma}$ are obtained through a set of training shapes.

B. Eigen-Analysis and Classification

In (13) and (14), the covariance matrix $\boldsymbol{\Sigma}$ is $2N$ by $2N$. In order to reduce the computation complexity, the matrix is decomposed via an eigenvector transform:

$$\boldsymbol{\Sigma} = \boldsymbol{\Phi}^T\boldsymbol{\Lambda}\boldsymbol{\Phi} \tag{15}$$

where $\boldsymbol{\Phi}$ is the eigenvector matrix and $\boldsymbol{\Lambda}$ is the corresponding diagonal matrix of eigenvalues. With eigenvector transform, the likelihood equation becomes:

$$P(\mathbf{v}\,|\,\Omega) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{2N}\frac{b_i^2}{\lambda_i}\right)}{(2\pi)^N\det(\boldsymbol{\Sigma})^{1/2}} \tag{16}$$

where $\mathbf{b} = \boldsymbol{\Phi}^T\tilde{\mathbf{v}}$ is the new vector under the orthogonal transform. Similarly, the Mahalanobis distance is converted to

$$D = \sum_{i=1}^{2N}\frac{b_i^2}{\lambda_i}. \tag{17}$$

In principle, eigenvectors correspond to the principal axes of the subvector space and the eigenvalues are the corresponding principal variance. Although in above orthogonal transform, all of the $2N$ eigenvectors are necessary to represent the distribution exactly, only a small number K ($K \ll 2N$) of them are generally needed to encode the samples within the subspace with tolerable errors. These K vectors are often called principle components and the approach called principle component analysis (PCA). With PCA, for each vector $\mathbf{v}$, only the first K projections of b_i are necessary for the computation of likelihood in (16) or Mahalanobis distance in (17). The computation is significantly reduced.

In practice, the thresholding of the Mahalanobis distance (17) is used in our work and it is approximated with two thresholding inequalities:

$$D_1 = \sum_{i=1}^{K} b_i^2/\lambda_i < T_1 \tag{18}$$

$$D_2 = \|\mathbf{v}\|^2 - \sum_{i=1}^{K} b_i^2 < T_2. \tag{19}$$

A shape is determined as in the canonical shape category only if its feature vector meets above two inequalities. Here D_1 is the Mahalanobis distance and D_2 is the energy of the feature vector's projection on the complementary space of the subspace spanned by these first K eigenvectors. In other words, we require that the projection inside the subspace is close to the

[1]By not rotational invariant we mean that canonical shapes are only modeled at the upright position as compared with positions such as upside down, because we assume people generally sit upright before camera. However, common shape changes at this position, i.e., slightly rotated head, are to be accommodated by our model.

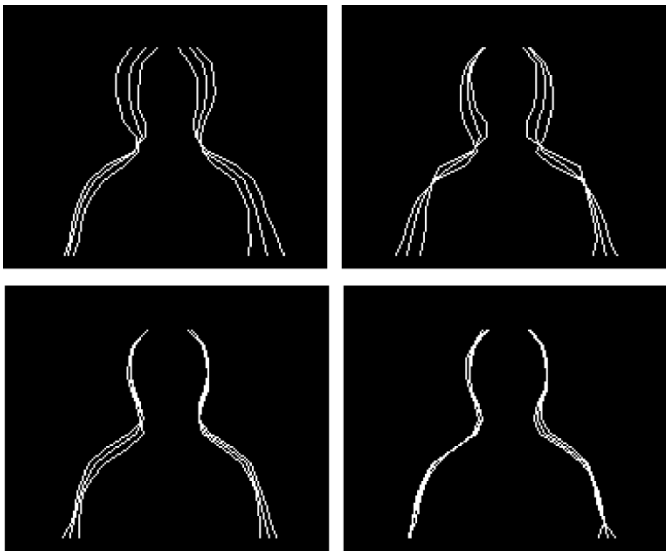


Fig. 7. Canonical shape category's eigen-shape illustration.

center, and the projection energy outside the subspace is small. This thresholding by projection may incur some error in the mathematical sense, but is justified by computation benefits.

In the training process, about 200 pictures of canonical shape category[2] were used. The result of eigen-analysis shows that the first six eigenvectors cover 92% of total energy. This is the thresholding K we used for our experiment. The shapes corresponding to the first four eigenvectors that obtained from our training set are shown in Fig. 7. They are ordered from left to right, top to bottom. In each figure, three shapes are overlaid corresponding to three vectors: $\bar{\mathbf{v}}, \bar{\mathbf{v}} + a\phi_i, \bar{\mathbf{v}} - a\phi_i$, where a is a weighting factor and ϕ_i is the i-th eigenvector. Readers can observe the distribution of shape changing along the first several principal eigenvectors and have a general sense of the canonical shape category's shape distribution discussed in this work.

C. Shape Assisted Blob Tracking

If a foreground shape is a canonical shape, by definition its shape features is used to segment the foreground into head region and shoulder regions, or in other words, to segment the foreground map $f(x, y)$ into a support map $s(x, y)$ of head blob and shoulder blob. A *containing rectangle* based algorithm is designed to segment a foreground map $f(x, y)$ into two blobs by setting a vertical threshold td:

$$s_1(x,y) = \begin{cases} f(x,y), & \text{if } y > \mathrm{td}, \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

and

$$s_2(x,y) = \begin{cases} f(x,y), & \text{if } y < \mathrm{td}, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

If we use operator $\mathcal{S}(s_k(\mathrm{td}))$ and $\mathcal{S}(\mathrm{rect}_k(\mathrm{td}))$ respectively to represent the size of support map k and size of its corresponding containing rectangle at the thresholding td, then the final segmentation thresholding $\hat{\mathrm{td}}$ is chosen as

$$\hat{\mathrm{td}} = \arg\min_{\mathrm{td}} \left\{ \sum_{k=1}^{2} [\mathcal{S}(\mathrm{rect}_k(\mathrm{td})) - \mathcal{S}(s_k(\mathrm{td}))] \right\}. \tag{22}$$

This algorithm is illustrated in Fig. 6(b). Note the coordinate system and the horizontal thresholding line that segments the foreground into head and shoulder regions. The purpose of this algorithm is to approximate the foreground map using two containing rectangles and choose the segmentation that minimizes the approximation error.

[2]Training samples are chosen by human observation, but their statistical model is created by computer. This is the process of machine learning. In our experiment, training samples were obtained from several persons. For wider application domain, more training samples should be obtained or users may train the machine with each person's own training samples.

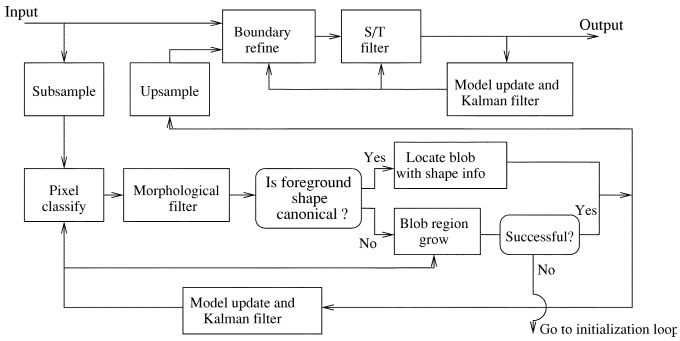


Fig. 8. Flowchart of the hierarchical tracking system.

IV. Hierarchical System Design

Though the region and shape statistical models reduce much analysis complexity, in practice, we find that to update the blob model parameters frame by frame still involves expensive computation. In addition, to make segmentation quality stable against noise, further filtering is necessary. A hierarchical structure is designed at the system level to solve these problems.

A. Hierarchical Architecture

In the new hierarchical design, the tracking loop of Fig. 4 is updated with Fig. 8, i.e., an input image is first subsampled by M in both horizontal and vertical directions. Model analysis (both region based blob model and shape model) and tracking are carried out in the obtained lower resolution image. The processing result is then up-sampled and further refined in the original resolution to produce the final output.

The benefits of this structure come from two aspects. First, because the statistical models are tracked and updated in the lower resolution image, the computation complexity is reduced by M^2. Second, when the segmentation result in the lower resolution image is mapped back to the full resolution image, only the boundary blocks[3] are further processed by the spatial and temporal filters that are designed to suppress noise and improve the boundary quality. All the interior blocks are skipped. One drawback, however, is that two versions of the background model are to be maintained, one in the lower and the other in the full resolution (for foreground model we can maintain just one set of model parameters and convert them

[3]One pixel in lower resolution image is mapped into one M by M block in full resolution image.

between different image resolutions). But compared with the benefits, this is not a big problem.

B. Processing on Subsampled Image

In Fig. 8, most function modules for lower resolution image processing were discussed in previous sections. The shape modules work as anchors for the region based tracking modules. They run in loop to find canonical shapes and locate the position of each blob. If the shape-based approach fails, the region-based blob analysis maintains its tracking with blob region growing module and information from the Kalman filter. At this stage, if system still can not find the expected blobs at the predicted positions, it changes its status back to the initialization loop. Unlike previous system defined by Fig. 4, the new system is more resistant to noise because of the shape analysis modules in the tracking loop.

C. Refining Processing on Full Resolution Image

In the full resolution layer, a boundary-refining module and a joint spatial and temporal (S/T) filtering module are designed to improve the boundary quality. In the three VOs we are going to get: background, head, and shoulder, head plus shoulder together is the foreground, only the boundary between the foreground and background are considered. If a head VO or a shoulder VO is required individually, the boundary between them is approximated with their containing rectangles.

In the hierarchical structure, each low-resolution-image pixel maps to a $M \times M$ block in the full resolution image. Both boundary-refining module and S/T filtering module process only the boundary block pixels.

Boundary refining module processes image in one frame to improve the spatial smoothness of final boundary. It includes three steps. First, pixels in the boundary blocks are classified based on the foreground model and background model in the full resolution layer. Second, morphological filters are used to connect the segmentation results in each boundary block with interior block regions, so as to produce a simple connected foreground map. After that, a relaxation procedure is carried out to improve the smoothness of the foreground boundary, which can be formulated as a statistical decision problem as follows.

Let $\Omega_k, (k = 0, 1)$ represents the events $f(x, y) = k, (k = 0, 1)$. For each boundary pixel, its MAP classification equation is

$$\ln(p(\Omega_k \mid \hat{\mathbf{y}})) = \ln(p(\hat{\mathbf{y}} \mid \Omega_k)) + \ln(p(\Omega_k)). \tag{23}$$

The first term in the right side can be obtained from (7). The second term $\ln(p(\Omega_k))$ works as a *smoothness measure*, which represents a priori knowledge. Because smoothness is a spatial feature, we define the *smoothness measure* of a boundary locally for each of its boundary pixel as [2] does. A priori density $p(\Omega_k)$ is modeled by a Markov random field considering a 3×3 neighborhood:

$$p(\Omega_k) = \frac{1}{Z} \exp\{-E(\Omega_k)\}, \quad (k = 0, 1) \tag{24}$$

where Z is a normalizing factor and the energy term E is defined as

$$E(\Omega_k) = (n_B(k)B + n_C(k)C). \tag{25}$$

In (25), $n_B(k)$ and $n_C(k)$ are the homogeneity measure of the neighborhood if current boundary pixel is labeled as k. They are obtained as follows. Each current boundary pixel constitutes eight *pixel-pair*s with its eight neighboring pixels. If both pixels in a pixel-pair have the same label, this pixel-pair is a *homogeneous pair*, otherwise it is an *heterogeneous pair*. n_B is the number of those heterogeneous pairs that are in vertical or horizontal positions and n_C is the number of those in diagonal positions. B and C are two weighting factors that represent the distance factor of those pixel-pairs in different positions in relation to the boundary pixel under consideration. We have $B = \sqrt{2}C$.

After spatial boundary refining, a seven-point three-dimensional (3-D) spatial–temporal median filter is used on the foreground map:

$$\begin{aligned}\mathrm{med}_t(x, y) = \mathrm{med}_7[&I_{t-1}(x, y), I_{t+1}(x, y), I_t(x - 1, y),\\ &I_t(x, y), I_t(x + 1, y), I_t(x, y - 1), I_t(x, y + 1)].\end{aligned} \tag{26}$$

The purpose of this median filter is to suppress the temporal high frequency noise on the boundary, which will be quite annoying when the segmented VOs are played back with an MPEG-4 player. Notice that this filter introduces delaying time of one frame. For real time applications, higher order median filter is not desirable.

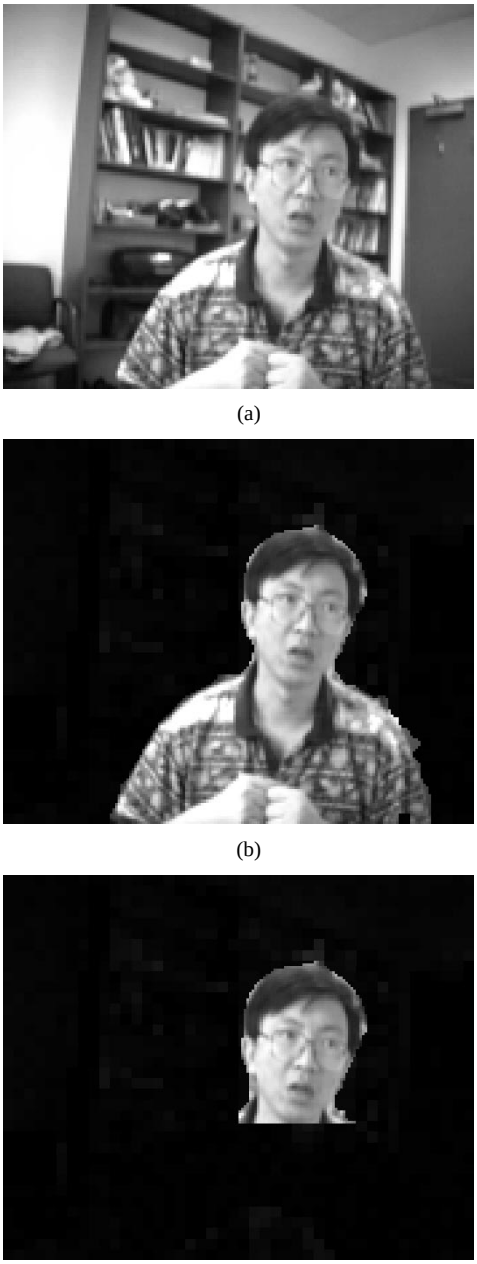
V. Segmentation Experiments

The algorithm is implemented on PC platforms with a variety of video capture hardware, including Intel's Proshare videoconference Kit, Intel's Create & Share Camera Pack and Sony's CCD SSC-S20 camera with Intel's Brooktree capture card. The segmentation performance is 15 fps (frames per second) on a Pentium-200 for QCIF (176×144) size input videos.

Due to the online feature of the algorithm, we could not use standard sequences in the test (because standard sequences such as Akiyo do not have the online information we are using to initialize the system). Instead, several testing sequences are captured for testing purpose. One of them, in QCIF size, YUV format, 800 frames in length and 10 fps is available on the website[4] for this paper. This video sequence contains one person in his head-and-shoulder pattern as foreground with moderate motion.

Fig. 9 shows the segmentation result on one frame (the 500th frame) of the testing video sequence. Fig. 9(a) is the original input frame, (b) is the segmented foreground VO and (c) is the segmented head VO. Note that the boundary between the head VO and shoulder VO is approximated with their containing rectangle boundaries. In addition, though our segmentation algorithm is able to segment the input frame into three regions: background, head and shoulder; some time it is also possible to combine the foreground part (head and shoulder) as one VO, which is also semantically meaningful.

[4]http://www.ctr.colulmbia.edu/~luoht/research/rvSeg



(a) (b) (c)

Fig. 9. Segmentation result example on one frame (frame 530) of the testing sequence. (a) Original input frame. (b) Segmented foreground VO. (c) Segmented head VO.

Fig. 10. Comparison of boundary relaxation and 3-D filter effects.

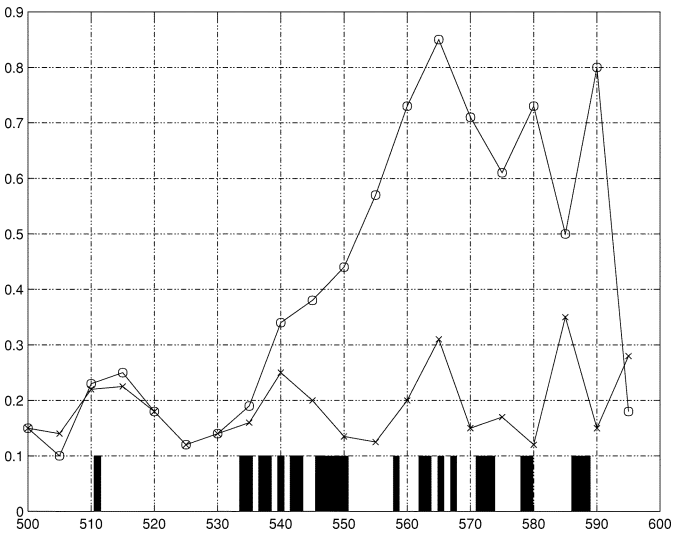


Fig. 11. Illustration of the function of the shape-based adjustment module.

To quantify the role of the shape-based adjustment module in the overall tracking system, a group of experimental data is used to compare the tracking performance with and without shape-based adjustment. The video data used is the mentioned 800-frame sequence. The accuracy of head region tracking is measured by comparing the tracked support map with a ground-truth support map, which is generated with a semi-automatic video segmentation tool [20] that we developed in our laboratory. The error rate is defined as

$$\text{error} = S_{\text{misclassified}} / S_{\text{ground-truth}}$$

where $S_{\text{misclassified}}, S_{\text{ground-truth}}$ represent the size of the misclassified and the ground-truth head region, respectively. Fig. 11 illustrates the result of this experiment (only the results for 100 frames, from frame 500 to 599 are included). In Fig. 11, the horizontal axis is the frame number and the vertical axis is the head region tracking error rate. The "o" curve represents the error rate of the tracking algorithm without shape-based adjustments and the "+" curve represents that of the algorithm with shape-based adjustments. For better observation, we also overlay a bar graph on the bottom of the figure that represents the detection of canonical shapes. On this bar graph, each bar along the horizontal axis means a detected canonical shape at the frame position. It can be seen that canonical shapes are detected on about 25% of the 100 frames. Due to this detection result, the tracking result using shape information is better than the result of the algorithm without it. This relation is especially obvious on frames from 550 to 600, where blob tracking errors get accumulated because of the large motion of head-and-shoulder foreground and the similar color of head and shoulder regions.

However, with canonical shape based adjustments, the tracking is much more reliable[5].

In Fig. 10, we compare the effect of boundary relaxation and 3-D spatial/temporal median filter. The first row from left to right are three consecutive original frames. The second row are their segmented results without relaxation and filtering. The third row are the final results with both relaxing and filtering. We can see that in the second row, some noise on the boundary is produced because parts of the background color are very similar to that of the foreground model. However, with boundary relaxation and 3-D filter, the boundaries in the third row are much smoother, both spatially and temporally.

VI. Application Discussion

With a real time object segmentation algorithm, real time multimedia services such as videophone and web-based video chatting can be improved in a number of ways by introducing MPEG-4 framework [23], [24]. In this section, we attempt to make comparison better MPEG-4 and H.263 using experimental data. For simulation purpose, we used MoMuSys MPEG-4 implementation version 7 [25] and Telenor's H.263 implementation version 2.0 [26], both are freely available on the web.

First, we study the compression gain from VO segmentation results. Because of the online feature of our segmentation algorithm, no standard video sequence could be used. Instead, we use the testing sequence described in Section V. Among the 800 frames, the 100 frames from 600 to 699 are chosen for this simulation.

The setup of MPEG-4 encoding and decoding is illustrated in Fig. 12. The input video I is first segmented and the foreground map f is obtained. With the segmented foreground map f, two VOs: one background B and one foreground F are further created. Among them, each background frame is obtained as

$$B_t(x,y) = \begin{cases} B_{t-1}(x,y), & \text{if } f_t(x,y) \neq 0, \\ I_t(x,y), & \text{otherwise.} \end{cases} \quad (27)$$

That is, a background pixel in the current frame is repeated with its value in the previous frame if it is occluded, or else it takes its current value. Both VOs are encoded and decoded separately, and then composed to create the final output O. For both VOs, MPEG-4 encoder uses the baseline mode,[6] VM4 rate control. The inter-VO rate control is realized by controlling the average quantizer of background VO to make it approximately equal to that of H.263 encoder. H.263 encoder also uses its baseline mode. Frame by frame average peak SNR of all the 100 frames are used as quality measure (when some frames are skipped by the encoder, the previous reconstructed frame is repeated to compute SNR at the decoder side). In the experiment, both MPEG-4 and H.263 used 10 fps output frame rate, neither of them skipped frames. In order to better evaluate the decoding

[5]In order to give the readers better sense of the accuracy and reliability performance of our algorithm, we put frame-by-frame segmentation results of all the 800 frames on our web as well.

[6]H.263 quantization table, no alpha threshold, no advanced prediction mode, and no shape effects mode.

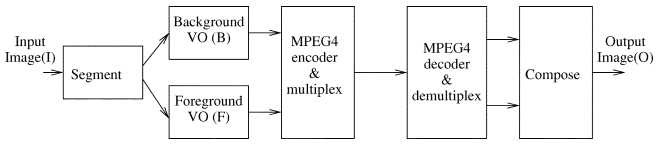


Fig. 12. Illustration of MPEG-4 encoder and decoder setup.

TABLE I
Comparison of Compression Efficiency of MPEG-4 and H.263

MPEG-4					H.263				
Rate	SNR	SNR-b	SNR-h	SNR-s	Rate	SNR	SNR-b	SNR-h	SNR-s
20.03	30.62	34.05	28.74	28.59	20.01	30.48	34.12	28.60	28.38
30.01	32.66	35.25	30.99	30.92	29.98	32.39	35.63	30.58	30.42
39.99	34.10	36.59	32.45	32.43	40.04	33.81	36.77	32.08	31.95
50.02	35.08	37.36	33.52	33.51	49.99	34.93	37.72	33.27	33.14
60.03	36.10	38.46	34.54	34.52	59.99	35.68	38.10	34.08	34.03

quality, region-based peak SNR is introduced as quality measure. For blob k's support region, its SNR is defined as

$$\text{SNR} = 10\log_{10}\left\{\frac{255^2\mathcal{S}(s_k)}{\sum_{s_k(x,y)=1}[I(x,y)-\hat{I}(x,y)]^2}\right\} \quad (28)$$

where $I(x,y)$ and $\hat{I}(x,y)$ are original and decoded pixel's scalar value, s_k is the blob support map defined in (1), and $\mathcal{S}(s_k)$ is the size of blob k's support map.

Table I compares the coding performance of MPEG-4 and H.263. Rate is in kilo bit per second. SNR, SNR-b, SNR-h and SNR-s refer to the Y component peak SNR of the entire frame, background region, head region and shoulder region, respectively. We can see that in the baseline setup and at the same bit rate, MPEG-4 achieves 0.16–0.42 dB gain in the frame level SNR compared with H.263. This is mainly because in an object-based coding approach, an encoder processes the foreground and background separately. It does not have to spend extra bits on coding the uncovered background while only spends moderate bits on shape coding. However, in a frame-based approach, uncovered background can not be handled by motion compensation and is expensive to encode.

Table I also shows different SNR distributions over different regions. In H.263, the background region always gets the best SNR result while the foreground region quality is not as good, because the foreground is always in motion and more difficult to compress. In MPEG-4, we reduce the background quality a little but the foreground quality is improved significantly. In Table I SNR-h and SNR-s for MPEG-4 are generally 0.5 dB better than those for H.263. Because the background is the biggest region in the testing frame, this measure penalizes MPEG-4 in the frame level SNR comparison. However, as the foreground is the focus of attention, the subjective result is better. That is, MPEG-4's coding efficiency is more than 0.16–0.4 dB better as indicated in Table I.

Second, we study the scalability gain from VO segmentation results. In traditional encoder like H.263, quantization adaptation is the only means to control the bit rate. With MPEG-4, *content-based scalability* can be added to quantization-based scalability. That is, we can choose to transmit part of the information that is semantically more important. In the case of videophone and web chatting, we can transmit only the head region (head

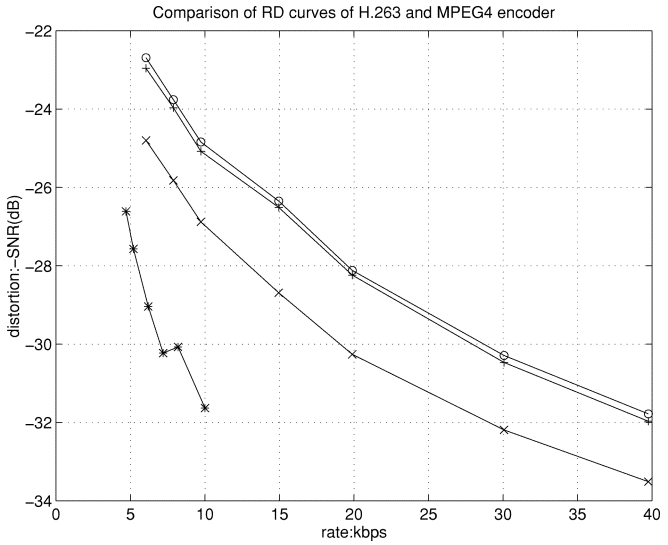
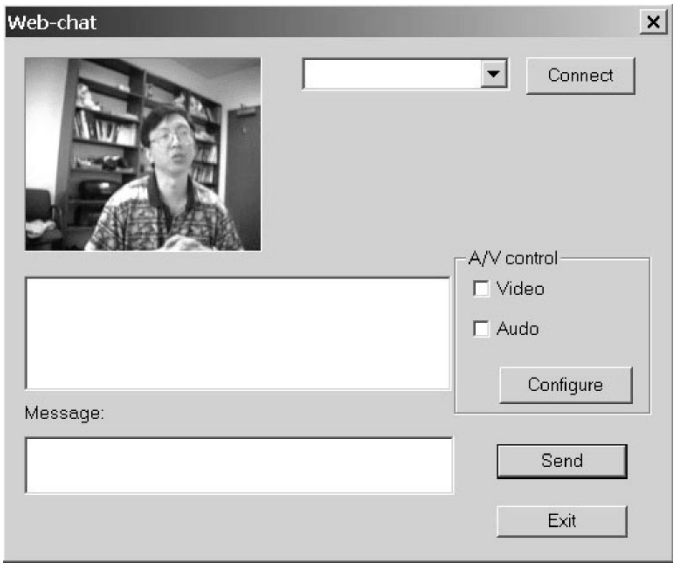


Fig. 13. Comparison of MPEG-4 and H.263 encoder behavior at low bandwidth. "∗": SNR for MPEG-4 encoder that encodes only the segmented head VO; "×": frame level SNR for H.263 encoder that encodes the same video without segmentation, "+" head region SNR for H.263 encoder, '"∘": shoulder region SNR for H.263 encoder.

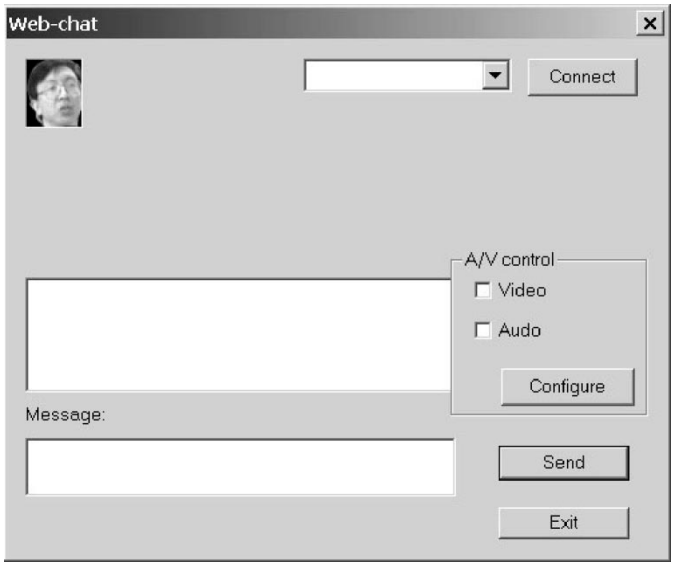
VO) in order to reach lower bit rate. According to our observation, as long as the segmentation quality is good, this type of scalability is acceptable to the end users.

Quantitative experiments are carried out on our 800-frame testing video. Frames 500 to 599 are used to compare the low bit rate behavior of MPEG-4 and H.263. In the comparison, we use MPEG-4 to encode the head VO only (obtained by segmentation with previous discussed online algorithm) and compare the SNR performance with H.263 that worked on full frame pictures. Both encoders use baseline mode with target frame rate set at 10 fps. The result is illustrated in Fig. 13. The figure is setup in a rate-distortion pattern, with -SNR used as distortion measure. For each encoder, different combinations of coding parameters are used to get sampling points. The sampling points are used to approximate the R–D behavior of each encoder (Sampling points are chosen that represent the best performance for both encoders in the experiment. Curves are not strict convex because different coding parameters and/or rate control options are chosen for different points in order to reach low bit rates). In the figure, it can be seen that using content-based scalability, MPEG-4 encoder reaches much lower bit rate. If we set 26 dB for the head region as the *acceptable quality boundary*, then for H.263 the lower bandwidth boundary is about 15 kbps, while for MPEG-4, the lower boundary is about 5 kbps. This is even lower than a good quality audio channel.

This experiment indicates that this technique will be useful in applications such as Internet video chatting, where no QoS is guaranteed, and friendly bandwidth adaptation is critical for a large scale fair sharing of available bandwidth resources. Fig. 14 illustrates the user interface of a web-based chatting software. In Fig. 14(a), a full resolution QCIF size video is sent when the network bandwidth is sufficient, while in Fig. 14(b), the video is automatically downscaled to only the head object to fit to a narrow bandwidth. Though the transmitted video is smaller, the motion of face region is well conveyed with audio, which maintains a good user feeling.



(a)



(b)

Fig. 14. User interface of a web-chat software that makes use of content-based scalability.

In addition, this segmentation and tracking approach is a possible solution to very-low-bit-rate coding. Though model-based-coding techniques like [27] can reach a bit rate of several hundred bits per second, it is too difficult to fit the model to human faces. Instead, our work suggests that the segmented head regions can be transmitted efficiently with MPEG-4. Because the bit rate is close to that of one channel audio, it is also likely to apply this technique to *video email*, in which head region is segmented and sent at several kbits per second.

VII. Conclusion

In this paper, a model based video analysis algorithm is proposed for videophone applications. Unlike other approaches, this algorithm emphasizes a real time performance. Application domain is limited and domain knowledge is abstracted and modeled with both blob based statistical region model and shape

model. With the assistance of model knowledge and a hierarchical processing structure, a QCIF size head-and-shoulder video can be segmented into background, head and shoulder three regions. Experiments show that this algorithm runs in real time on average PC platforms. We believe that this algorithm is a useful tool for applying the new object based MPEG-4 standard to popular real time video service. Our simulation results presented in the last part of this paper also support this opinion.

REFERENCES

[1] C. Lettera and L. Mastera, "Foreground/background segmentation in videotelephony," *Signal Process.: Image Commun.*, vol. 1, pp. 181–189, 1991.

[2] T. Aach, A. Kaup, and R. Mester, "Statistical model-based change detection in moving video," *Signal Process.*, vol. 31, pp. 165–180, 1993.

[3] M. Bichsel, "Segmenting simple connected moving objects in static scene," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 1138–1142, 1994.

[4] R. Mech and P. Gerken, Automatic segmentation of moving objects, , ISO/IEC JTC1/SC29/WG11, MPEG 96/1549, Nov. 1996.

[5] C. Kim and J. N. Hwang, "A fast and robust moving object segmentation in video sequences," presented at the ICIP, Kobe, Japan, Oct. 1999.

[6] J. Stauder, R. Mech, and J. Ostermann, "Detection of moving cast shadows for object segmentation," *IEEE Trans. Multimedia*, vol. 1, pp. 65–76, Mar. 1999.

[7] J. Wang and E. Andelson, "Representation moving images with layers," *IEEE Trans. Image Processing*, vol. 3, pp. 625–638, Sept. 1994.

[8] M. M. Chang, A. M. Tekalp, and M. I. Sezan, "Simultaneous motion estimation and segmentation," *IEEE Trans. Image Processing*, vol. 6, pp. 1326–1333, Sept. 1997.

[9] E. Steinbach, P. Eisert, and B. Girod, "Motion-based analysis and segmentation of image sequences using 3-d scene model," *Signal Process.*, vol. 66, pp. 233–247, 1998.

[10] E. Tuncel and L. Onural, "Utilization of the recursive shortest spanning tree algorithm for video-object segmentation by 2-d affine motion modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 776–780, 2000.

[11] J. G. Choi, S. W. Lee, and S. D. Kim, "Spatio-temporal video segmentation using a joint similarity measure," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 279–285, 1997.

[12] Y. Altunbasak and A. M. Takalp, "Region-based parametric motion segmentation using color information," *Graph. Models Image Process.*, vol. 60, no. 1, pp. 13–23, 1998.

[13] E. Chalom and V. M. Bove Jr., "Segmentation of an image sequence using multi-dimensional image attributes," presented at the ICIP, Lausanne, Sept. 1996.

[14] B. Marcotegui *et al.*, "A video object generation tool allowing friendly user interaction," presented at the ICIP, Kobe, Japan, Oct. 1999.

[15] D. Zhong and S.-F. Chang, "An integrated approach for content-based video object segmentation and retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 1259–1268, Dec. 1999.

[16] S. Malassiotis and M. Strintzis, "Tracking textured deformable objects using a finite element mesh," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 6, pp. 756–775, 1998.

[17] C. Toklu, A. M. Tekalp, and A. T. Erdem, "Semi-automatic alpha plane generation by occlusion-adaptive mesh tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 4, pp. 624–629, 2000.

[18] C. Gu and M. C. Lee, "Semiautomatic segmentation and tracking of semantic video objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 572–84, 1998.

[19] 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.

[20] H. Luo and A. Eleftheriadis, "Designing an interactive tool for video object segmentation and annotation," presented at the ACM Int. Multimedia Conf., Orlando, FL, Oct. 1999.

[21] P. Correia and F. Pereira, "The role of analysis in content-based video coding and indexing," *Signal Process.*, vol. 66, pp. 125–142, 1998.

[22] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder, real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 780–785, July 1997.

[23] A. Puri and A. Eleftheriadis, "MPEG-4: An object-based multimedia coding standard supporting mobile applications," *ACM Mobile Network Applicat.J.*, Aug. 1997.

[24] T. Sikora, "The MPEG-4 video standard verification model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 19–31, 1997.

[25] MPEG-4 Software [Online]. Available: ftp://ftp.elec.qmw.ac.uk/public /telecom/sc29wg11/ MoMuSys~/VM970 704.tar.gz

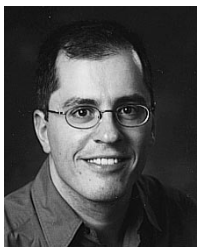
[26] H.263 Software TMN2.0 [Online]. Available: ftp://bonde.nta.no/pub/tmn/software/tmn-2.0.tar.gz

[27] K. Aizawa, "Human facial motion analysis and synthesis with application to model-based coding," in *Motion Analysis and Image Sequence Processing*. Norwell, MA: Kluwer, 1993.

Huitao Luo (M'01) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1994 and 1996, and the Ph.D. degree from Columbia University, New York, in 2000, all in electrical engineering.

He was with the IBM T. J. Watson Research Center, Yorktown Heights, NY, and AT&T Research Labs, Red Bank, NJ, during the summers of 1997 and 1999, respectively. Currently he is Member of Technical Staff at Hewlett-Packard Labs, Palo Alto, CA. His major research interests are image and video processing, pattern recognition, and multimedia systems.

Dr. Luo is a member of SPIE and ACM.



Alexandros Eleftheriadis (M'95) was born in Athens, Greece, in 1967. He received the diploma in electrical engineering and computer science from the National Technical University of Athens, Greece, in 1990 and the M.S., M.Phil., and Ph.D. degrees in electrical engineering from Columbia University, New York, in 1992, 1994, and 1995 respectively.

Since 1995, he has been with the faculty of the Department of Electrical Engineering, Columbia University, where he is currently an Associate Professor. He was also co-founder and Chief Technical Officer of Flavor Software, Inc., New York, a company that builds software for creating and distributing rich media content using MPEG-4, and continuous to serves as an advisor to several startup companies. In the summers of 1993 and 1994, he was with the Signal Processing Research Department of AT&T Bell Labs, Holmdel, NJ, where he performed research on very low bit rate coding systems. His research interests are in media representation, with emphasis on multimedia software, video signal processing and compression, and video communication systems.