Video Object Segmentation and Tracking Using ψ-Learning Classification

Yi Liu, *Student Member, IEEE,* and Yuan F. Zheng, *Fellow, IEEE*

***Abstract*—As a requisite of the emerging content-based multimedia technologies, video object (VO) extraction is of great importance. This paper presents a novel semiautomatic segmentation and tracking method for single VO extraction. Unlike traditional approaches, the proposed method formulates the separation of the VO from the background as a classification problem. Each frame is divided into small blocks of uniform size, which are called *object blocks* if the centering pixels belong to the object, or *background blocks* otherwise. After a manual segmentation of the first frame, the blocks of this frame are used as the training samples for the object-background classifier. A newly developed learning tool called ψ-learning is employed to train the classifier which outperforms the conventional Support Vector Machines in linearly nonseparable cases. To deal with large and complex objects, a multilayer approach constructing a so-called *hyperplane tree* is proposed. Each node of the tree represents a hyperplane, responsible for classifying only a subset of the training samples. Multiple hyperplanes are thus needed to classify the entire set. Through the combination of the multilayer scheme and ψ-learning, one can avoid the complexity of nonlinear mapping as well as achieve high classification accuracy. During the tracking phase, the pixel in the center of every block in a successive frame is classified by a sequence of hyperplanes from the root to a leaf node of the hyperplane tree, and the class of the block is identified accordingly. All the object blocks thus form the object of interest, whose boundary unfortunately is stair-like due to the block effect. In order to obtain the pixel-wise boundary in a cost efficient way, a pyramid boundary refining algorithm is designed, which iteratively selects a few informative pixels for class label checking, and reduces uncertainty about the actual boundary of the object. The proposed method has been applied on video sequences with various spatial and temporal characteristics, and experimental results demonstrate it to be effective, efficient, and robust.**

***Index Terms*—ψ-learning, support vector machines (SVM), video object (VO) extraction, VO segmentation and tracking.**

I. Introduction

In the past several years, there has been rapidly growing interest in content-based functionalities of video data, such as video editing, content-based image retrieval, video indexing, video event analysis, and etc. To facilitate these functionalities, MPEG-4, the new video compression standard, introduces the concept of video objects (VOs) that correspond to semantic entities [1], [2]. In addition to natural interpretations, the object-based representation also offers flexible content manipulations. However, how to obtain VOs from the raw data is still a very challenging problem.

Many automatic video segmentation approaches can be found in the literature [3]–[10], and according to the primary criterion for segmentation they can be roughly categorized into two classes: spatial-based methods and temporal-based methods. The spatial-based segmentation method partitions each frame into homogeneous regions with respect to color or intensities. Then every region is tracked through time using the motion information. Typical partitioning algorithms include morphological watershed [3], K-means clustering [4], region growing [5], and the recursive shortest spanning tree [6]. A major advantage of the spatial-based segmentation approach is that it can yield relatively accurate object boundary. However the computational complexity is quite high and thus limits their usage in real-time applications since the segmentation has to be done on the whole image for every frame.

The temporal-based segmentation approach [7]–[9], on the other hand, utilizes the motion rather than spatial information to obtain the initial position and boundary of VOs. Because the objects of interest are usually moving, change detection is the major scheme for segmentation that can be done on the inter-frame or background-frame basis. Due to the image noise, objects' boundaries are often irregular and require to be refined using the spatial information of the image. As the boundary fine-tuning procedure involves only the segmented moving region instead of the whole frame, higher efficiency is achieved. Unfortunately smooth motion, which is the essential assumption of the temporal-based method, may not always hold. For instance, when the frame loss occurs during the transmission of video or the object exhibits abrupt variation of motion, the performance degrades. Another innovative approach is to fuse the intermediate results obtained by using different methods of segmentation [10].

Because of the semantic meaning a VO may carry, it may actually consists of arbitrary collections of image regions which may undergo noncoherent motions. For example, a person who is waving is not homogeneous as a VO in terms of color or motion. For this reason semiautomatic video segmentation, which defines the objects through users' supervision and tracks them in an unsupervised manner, has received a lot of attention [11]–[14]. In the semiautomatic framework, the objects of interest are initially extracted with user's assistance and then a model representing the object is created. A variety of models have been proposed including two-dimensional (2-D) mesh [15], [16], binary model [17], deformable templates [18],

Manuscript received October 7, 2003; revised February 9, 2004. This paper was recommended by Associate Editor L. Onural.

Y. Liu is with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: liuyi@ece.osu.edu).

Y. F. Zheng is with the Shanghai Jiao Tong University, Shanghai 200030, China, on leave from the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: zheng@ece.osu.edu).

Digital Object Identifier 10.1109/TCSVT.2005.848346

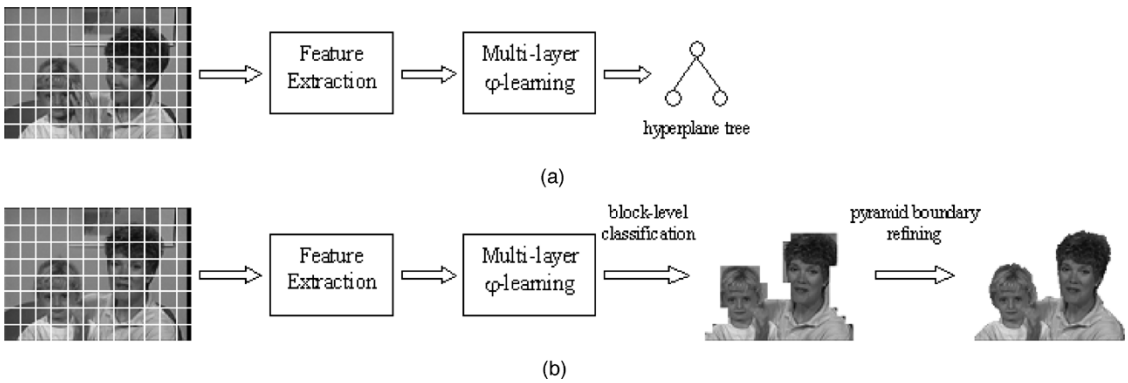


Fig. 1. Overview of the proposed approach. (a) Training phase. (b) Tracking phase.

corners and lines [19], etc. Then in the subsequent frames the object is tracked and located where the best match of the model is found.

We consider the object segmentation and tracking problem to be a classification problem. Single object tracking, for example, requires identifying each pixel as either foreground or background. Similarly, multiple object tracking can be formulated as a multiclass classification problem. From this perspective, we propose a novel semiautomatic approach for single VO extraction which is significantly different from aforementioned approaches yet overcomes many of their shortcomings.

The basic idea of our approach is to decompose each frame into small blocks, classify them as foreground or background, and form the object of interest by all the foreground blocks. Evidently in our approach the classification accuracy is a very crucial issue since higher classification accuracy leads to better tracking performance. So we employ ψ-learning [20], a newly proposed learning machine, as the classifier due to its outstanding generalization ability. Meanwhile in order to obtain high classification accuracy at an affordable computational cost, we also develop four innovative mechanisms including *local and neighboring feature representation*, *multilayer classification*, *block-level classification*, and *pyramid boundary refining*, which will be discussed in detail later in this paper.

Fig. 1 presents an overview of the proposed scheme. As one can see, it consists of two phases: 1) the training phase and 2) the tracking phase. The training phase begins with dividing the first frame, chosen as the training frame, into blocks that are defined as *object blocks* or *background blocks* depending on which class the pixels in the block center belong to. Every centering pixel as well as every block is represented by the local and neighboring features. Then through the multilayer ψ-learning, a set of linear decision functions that are stored in a tree structure are obtained. In the tracking phase, each subsequent frame is also divided into blocks, and for each block the set of decision functions are evaluated to decide whether the pixel at its center belongs to the object or not, which consequently determines the class label of the block. Finally, the tracking mask is formed by all the identified object blocks. At this point the resolution of object's boundary is as large as the size of the block. To obtain pixel-wise accuracy, we design a so-called *pyramid boundary refining algorithm* which is able to refine the object boundary in an efficient and scalable manner.

Comparing with previous works, our method has following advantages.

1) ***Low computational complexity***. In the proposed classification framework, the time-consuming processes of object modeling, extracting, and searching are avoided. Moreover, the tracking is achieved through the testing phase of the learning machine, which only requires evaluation of a small number of linear functions. As a result, our approach has lower computational complexity than many spatial-based approaches while providing comparably accurate object boundaries;
2) ***Robust to motion fluctuation***. Because object tracking is conceived as a classification problem, temporal correspondence of the object between frames is automatically maintained through the classification and therefore free of any motion assumption. As a result, our approach can perform well even when the object stays still for arbitrarily long period of time or when its different parts exhibit different motion characteristics.
3) ***Robust to occlusion***. Occlusion is a very challenging scenario for both automatic and semiautomatic approaches, such as the template-and-matching method. The proposed semiautomatic approach, in contrast, by decomposing object into blocks, is able to recognize un-occluded object portions as long as they still exhibit the object features.

There are only a few approaches that handle VO tracking as a classification problem, and the most similar work is done by Doulamis *et al.* [21]. Our work, however, is different from [21] in three major aspects. First, the classifiers employed are different. Second, in the classification step we introduce the block-level classification and a pyramid refining scheme to refine the boundary so as to save computational cost while in [21] the classification is carried out pixel by pixel. Third, [21] uses an automatic approach, yet ours is semiautomatic because the user's involvement is needed to define the object of interest in the training phase. It should also be emphasized that only object/background separation is addressed in this paper. Nevertheless, the extension of the current approach to multiple object tracking is possible and some discussions for that purpose are presented in Section VII.

The rest of the paper is organized as follows. Section II briefly introduces ψ-learning and support vector machines (SVMs).

Section III explains the multilayer classification scheme in detail as well as the local and neighboring feature representations. Then the block-level classification and the pyramid boundary refining algorithm are represented in Section IV and Section V, respectively. Experimental results are shown in Section VI which is followed by conclusions in Section VII.

II. ψ-Learning

ψ-learning is a newly developed classification approach that constructs the decision function f by directly minimizing the generalization error [20]. Although motivated by the same concept of margin-maximization as in SVM and can be considered as a variation of it, ψ-learning demonstrates its advantage over SVM in *linearly nonseparable cases* both theoretically and experimentally [20]. In this section, we will introduce this new learning tool and compare it with SVM.

Assume a binary linear classification scenario for a set of input vectors denoted as X and the class index of the vectors denoted as Y. Instances of X and Y are denoted as x and y, respectively. ψ-learning seeks a linear optimal decision function $f(x) = \omega \cdot x + b$ based on a training set, where ω is a vector that is of the same dimension as x, and b is a scalar.

Suppose that the training set has N elements $(x_1, y_1), \ldots, (x_N, y_N)$, where x_i represents the training sample and $y_i \in \{+1, -1\}$ is the desired output corresponding to x_i. Once trained, a machine will classify an input vector x according to the sign of $f(x)$, that is

$$\hat{y} = \begin{cases} 1, & f(x) \geq 0 \\ -1, & f(x) < 0. \end{cases}$$

how to derive $f(x)$ differentiates ψ-learning from SVM.

A. Derivation of ψ-Learning

The *generalization error* of classification is defined as $Err(f) = P(Yf(X) < 0)$ [20]. It is easy to see that

$$\begin{aligned} E\left(1 - Sign\left(Yf(X)\right)\right) &= (1-1)P\left(Yf(X) \geq 0\right) \\ &\quad + (1-(-1))\,P\left(Yf(X) < 0\right) \\ &= 2 \cdot P\left(Yf(X) < 0\right) \end{aligned} \tag{1}$$

which gives us

$$Err(f) = \frac{1}{2} E\left(1 - \mathrm{Sign}\left(Yf(X)\right)\right). \tag{2}$$

The *training error*, which is the empirical version of (2), is equal to

$$(2N)^{-1} \sum_{i=1}^{N} \left(1 - \mathrm{Sign}\left(y_i f(x_i)\right)\right) \tag{3}$$

where N is the size of the training set. Minimizing (3) complies with *empirical risk minimization* induction principle. However, observing that $Err(f)$ is the "average" over the ensemble while the training error is just the average error over some realizations, there may not be quite small discrepancy between those two quantities, especially when N is small [22].

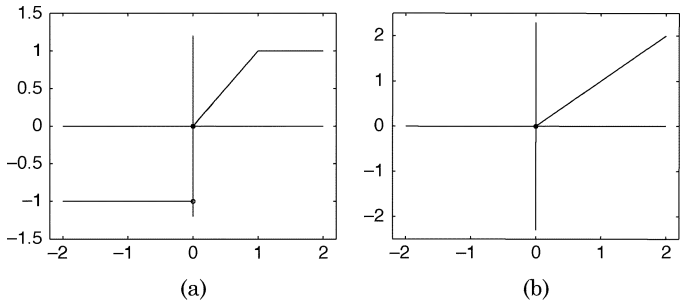


Fig. 2. ψ function for ψ-learning and $(\cdot)_+$ function for SVM.

In order to bound the difference between (2) and (3), a term $(1/2)\|\omega\|^2$, which is inversely proportional to the separating margin of classification in SVM, is added to (3). Then the following objective function is to be minimized:

$$\frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{N} \left(1 - \mathrm{Sign}\left(y_i f(x_i)\right)\right) \tag{4}$$

where C is the tuning parameter used to balance the separating margin and the training error. A large value of C indicates the importance of the empirical error.

However, the cost function represented by (4) has a numerical problem. If we scale ω and b by a same positive factor M, the new function $f(x) = (\omega/M) \cdot x + (b/M)$ would yield the same classification result for the same x and in turn give the same training error. Meanwhile $(1/2)\|\omega\|^2$ decreases by $1/M^2$. In this way ω and b continue to decrease until both reach the machine precision, and as expected the final solution to (4) turns out to be a meaningless function $f(x) = 0$. To overcome this drawback, the Sign function is replaced by the ψ function, which as shown in Fig. 2(a), penalizes the points that enter the strip $0 \leq yf(x) < 1$.

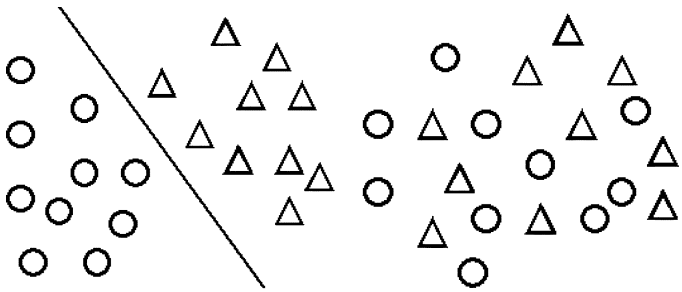
With the ψ function, the cost function becomes

$$\frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{N} \left(1 - \psi\left(y_i f(x_i)\right)\right). \tag{5}$$

Equation (5) needs to be minimized with respect to ω and b *without constraints*. Furthermore, it can not be solved directly by quadratic programming as SVM does, which makes optimization even more complex. An algorithm for implementation is addressed in [20].

B. Comparing ψ-Learning With SVM

As a powerful learning machine, SVM has been successfully applied in a variety of areas including object recognition [23]–[25], communications [26], [27], and recently image/video analysis [28]–[30]. Based on the *structural risk minimization* induction principle, SVM provides a guaranteed bounded risk value even when the number of the training set is small. Detailed description of SVM can be found in [31]–[33].



(a) (b)

Fig. 3. (a) Linearly separable case. (b) Linearly nonseparable case.

For a linearly nonseparable case, SVM is the solution to the following optimization problem:

$$\begin{aligned} &\text{Minimize}: \quad f(\omega,\xi)=\frac{1}{2}\|\omega\|^2+C\sum_{i=1}^{N}\xi_i \\ &\text{Subject to}: \; y_i(\omega_i\cdot x_i+b)\geq 1-\xi_i,\; i=1,2,\ldots,N \end{aligned} \tag{6}$$

where $\xi_i, i=1,\ldots,N$, are called slack variables and are related to the soft margin. It is easy to see that every ξ_i satisfies:

$$\xi_i=\begin{cases} 0, & 1-y_i f(x_i)\leq 0 \\ 1-y_i f(x_i), & 1-y_i f(x_i)>0. \end{cases} \tag{7}$$

With (7), the cost function for SVM can be rewritten as

$$\text{Minimize}: \quad \frac{1}{2}\|\omega\|^2+C\sum_{i=1}^{N}(1-y_i f(x_i))_+. \tag{8}$$

The plot of $(\cdot)_+$ function is displayed in Fig. 2(b). Comparing (5) with (8), one can see that ψ-learning and SVM have similar objective functions but with difference in the second term.

In the linearly separable case as shown in Fig. 3(a), the inequalities $y_i f(x_i)\geq 1$ are forced true in both ψ-learning and SVM. In this regard, the two approaches are essentially the same. In the linearly nonseparable case as shown in Fig. 3(b), the second term of the two approaches behaves differently toward the wrongly classified samples. In SVM, the samples will affect the location of the hyperplane depending on their distances to the decision boundary. In other words, they force the estimated boundary to move toward them. The farther the distance of a sample, the stronger the moving force toward the sample, reflecting the fact that the $(\cdot)_+$ function is linearly proportional to its variable in the positive side. In contrast to SVM, the ψ function keeps constant when its variable is larger than 1, which forces ψ-learning to treat the wrongly classified samples in the same way regardless of how far they are. Consequently, the hyperplane is robust against those samples. This major difference leads to ψ-learning outperforming SVM in linearly nonseparable cases.

Since the blocks of object and background are not all separable in our block-based approach, we choose ψ-learning instead of SVM for block classification which should result in a better performance than using SVM. The details of this application are discussed in the next section.

B_1	B_2	B_3
B_8	B_0	B_4
B_7	B_6	B_5

Fig. 4. Eight-connected neighboring blocks of block B_0.

III. Multilayer ψ-Learning for Object Tracking

A. Local and Neighboring Features

For object tracking, the ideal result is to extract the object at the pixel level resolution. If we represent individual pixels utilizing pixel-wise color or intensity information as most spacial-based approaches do, a lot of misclassifications would occur due to the negligence of the support of the spatial relationship among pixels. Take the *silent sequence* as an example, assuming that the human body is the object of interest. The background contains a large amount of small areas whose chrominance characteristics are very close to those of the face or hair regions. As a result, many background pixels will be tracked as those of the human body. To confront this problem we extract features from *block regions* centering at the pixel of interest, which describes a pixel not only by its chrominance or luminance values, but also some of the spatial structures among them. In this way more reliable classifications is able to be rendered.

Associated with each pixel there are two types of blocks defined for the feature extraction purpose: *unit blocks* and *neighboring blocks*. A unit block is the smallest block we are dealing with in our algorithm. More specifically, it has the size of 9×9 pixels and its centering pixel is what we want to represent and classify. Neighboring blocks, as the name suggests, are the 8-connected neighbors of the unit block as shown in Fig. 4. Two types of features are constructed accordingly: *local features*, denoted as $\vec{f}_{\text{local}}$, and *neighboring features* $\vec{f}_{\text{neighbor}}$, which are defined as follows.

1) Local Features: Local features extraction procedure collects the information from a unit block by applying the discrete cosine transform (DCT) and constructing a feature vector as follows:

$$\begin{aligned} \vec{f}_{\text{local}} &= (f_0,f_1,f_2,f_3)^T \\ &= \begin{pmatrix} c(0,0) \\ \sqrt{\sum_{j=1}^{N-1}c(0,j)^2} \\ \sqrt{\sum_{i=1}^{N-1}c(i,0)^2} \\ \sqrt{\sum_{i=1}^{N-1}\sum_{j=1}^{N-1}c(i,j)^2} \end{pmatrix} \end{aligned} \tag{9}$$

where $c(i,j)$ are the DCT coefficients, f_0 is the average intensity, and f_1 and f_2 represent the horizontal and vertical edges, respectively. All the other high-frequency information is contained in the last component f_3. Because of the unbalanced energy distribution among coefficients, many high frequency components $c(i,j)$ are close to zero. For this reason, we set $N=3$ in (9) and use only the first nine DCT coefficients when calculating $\vec{f}_{\text{local}}$.

2) Neighboring Features: In contrast to unit blocks, neighboring blocks contribute to the extraction of neighboring features. For a 9×9 unit block B_0, its neighbors are eight 9×9 blocks that are adjacent in the vertical, horizontal and diagonal directions. With $\mathrm{avg}(B_i)$ denoted as the average intensity of block B_i we compute the neighboring features as

$$\vec{f}_{\mathrm{neighbor}} = \begin{pmatrix} \mathrm{avg}(B_1 + B_2 + B_3) \\ \mathrm{avg}(B_3 + B_4 + B_5) \\ \mathrm{avg}(B_5 + B_6 + B_7) \\ \mathrm{avg}(B_7 + B_8 + B_1) \end{pmatrix}. \tag{10}$$

The calculations given above only consider the grayscale information. When the video sequence is chromatic, we compute (9) and (10) for each color component and then concatenate the vectors respectively to form the chromatically local and neighboring features.

The purpose of introducing both local and neighboring features is to make classification efficient and effective. The four-dimensional (or twelve-dimensional for chromatic sequences) local feature, rather than all DCT coefficients, reduces the data amount for representation while the neighboring features help separate the pixels that are similar when only local features are considered. Those two features will be used in the multilayer classification process.

B. The MultiLayer Scheme of Classification

Linear classification yields good performance when the object can be easily separated from the background. When the object and the background become complex and share some common features, the classification boundary tends to be nonlinear. Fig. 6(a) gives such an example, in which one linear decision function does not completely separate the object samples from the background. Consequently, a significant portion of the object is identified as background or vice versa. Some nonlinear decision functions have been investigated to solve this problem which unfortunately impose a high computational cost and remain to be an open topic of study technically [22].

We propose a hierarchical partition scheme which breaks the initial training set into many subsets, each of which contains samples that are more likely separated by a linear boundary. In other words, piecewise linear hyperplanes are used to approximate the nonlinear boundaries. Previously, [34] and [35] used this idea to yield better classification performance and to reduce the computation time.

We further propose a multilayer method that partitions the training set sequentially according to the results of the previous classification step. Instead of only one classifier, this method yields a hyperplane decision tree consisting of all the hyperplanes that are used to divide the training set. Each node of the tree represents one hyperplane, denoted as HP_s^l where the superscript l represents the level of the node while the subscript s denotes the path from the root to the current node, as shown in Fig. 5(a). After the first separation, each subset may still contain both object and background samples. Two hyperplanes are then generated to separate the two subsets, respectively. In Fig. 5(a), we use R and L to represent the two hyperplanes.

The hyperplanes along a path from the root to any one leaf node will eventually separate the object from the background.

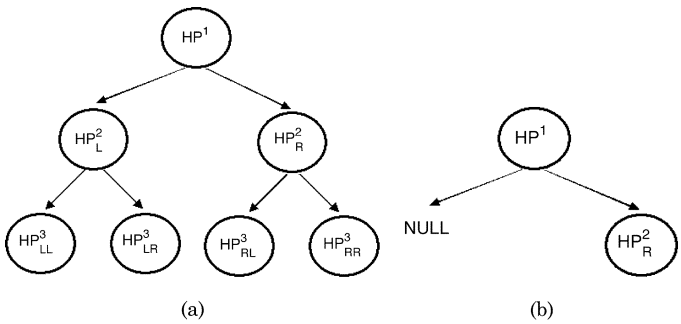


Fig. 5. Examples of a hyperplane tree. (a) General two-level hyperplane tree. (b) Hyperplane tree associated with the data shown in Fig. 6.

Fig. 6(c) displays two linear boundaries which are obtained when our approach is applied to the same samples in Fig. 6(a), while the constructed hyperplane tree is depicted in Fig. 5(b). For this particular case, the hyperplane tree is unbalanced because one subset after the first separation by HP^1 has only the object samples so that no more separation is needed. On the other side of HP^1 the subset has both object and background samples, which are further separated by HP_R^2.

It is important to note that ψ-learning is more suitable than SVM for this multilayer approach. As mentioned in Section II, due to the shape of the cost function, ψ-learning is more robust against the misclassified samples while SVM is more sensitive. Thus the hyperplane is aligned more closely to the local boundaries of the two classes of samples by ψ-learning than by SVM. The hyperplanes generated by SVM are strongly influenced by the global distribution of the training samples which is contrary to the objective of the multilayer approach. For example, the hyperplane HP_1 obtained by SVM for the cluster of Fig. 6(a) is shown in Fig. 6(d). Evidently, the SVM approach generates a compromise for all the samples and thus not suitable for further separation of misclassified blocks.

Technically, the multilayer scheme takes the following two steps. The first step is to generate the hyperplane tree. It begins with the initial training set $S = B \cup O$, where B and O represent the set of the background and the object, respectively. By training the learning machine using all the samples in S, the first hyperplane HP^1 representing the root of the tree is obtained. Depending on which side of HP^1 they are on, the samples in S is partitioned into two subsets denoted as $\hat{B}$ and $\hat{O}$. Usually, $\hat{B} \neq B$ and $\hat{O} \neq O$ because there always exist some background samples that are wrongly classified as the object by HP_1 and vice versa. If so, $\hat{B}$ and $\hat{O}$ are trained independently to obtain two additional hyperplanes, denoted as HP_R^2 and HP_L^2, respectively, and the tree size grows to two levels. At this point, the training set is divided into four subsets. If necessary the four subsets will be partitioned again, so forth and so on. In general, the more levels the tree has, the smaller the subsets which S is broken into. This process continues until the percentage of the misclassified samples in all the new subsets is no greater than a predetermined threshold ϵ, which is set to be 0.05 through out our experiments.

Once the hyperplane is obtained using the approach just described, it can be used to classify the pixels (i, j) in the subsequent video frames. It follows a sequential classification procedure starting from the root of the tree and ending at a leaf node.

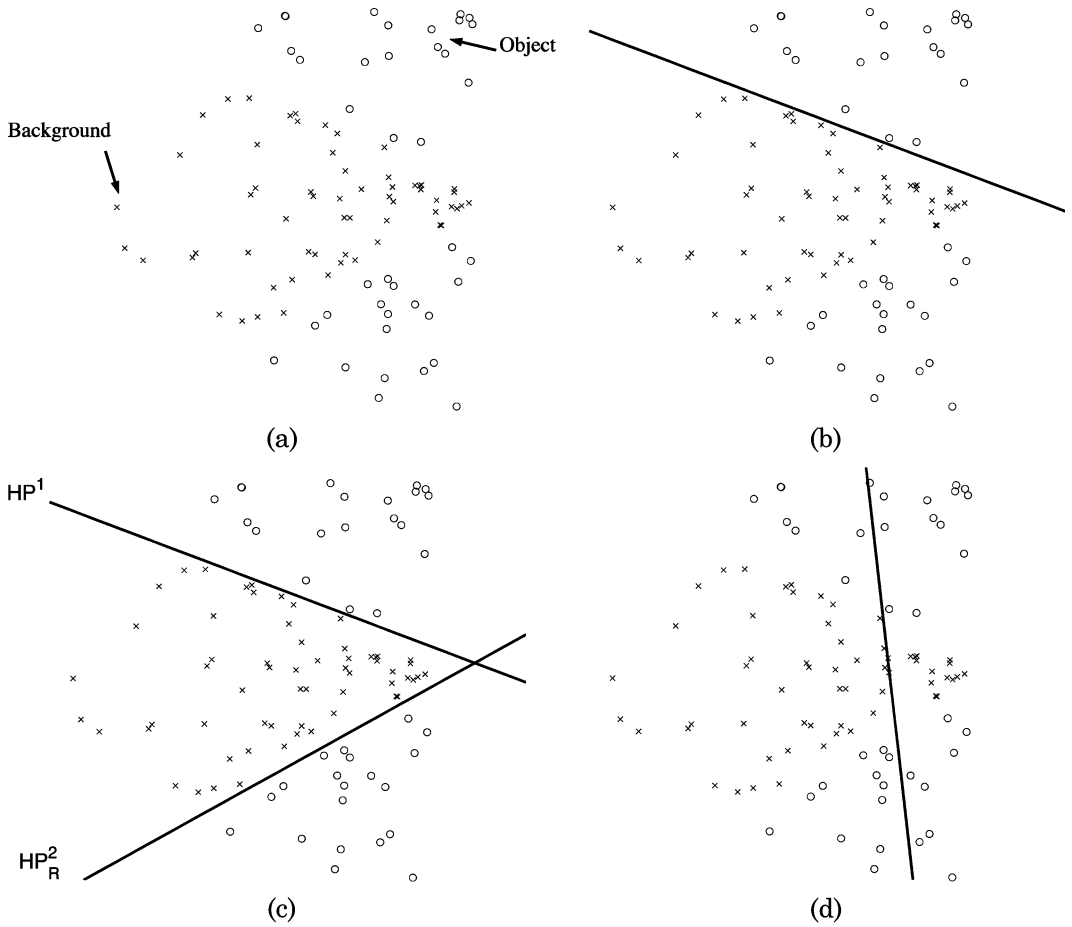


Fig. 6. Illustration of the multilayer method. (a) 100 training samples S_i with coordinates $(x_1^i, x_2^i) = (\rho_i \cos\theta_i, \rho_i \sin\theta_i)$ are randomly generated in the left-hand side circle of the unit disk. The sample S_i is labeled as "background" if $\theta_i \in [(2\pi/3)\ (4\pi/3)]$. Otherwise, it is labeled as "object". (b) Hyperplane ($w = [28.1\ 32.2]^T$, $b = -4.3$) is obtained by training the first layer. (c) One additional hyperplane HP_R^2 ($w = [29.2\ -22.8]^T$, $b = -1.5$) is obtained by training the second layer. (d) A different hyperplane ($w = [5.4\ 0.27]^T$, $b = 0.9$) is obtained when the first layer is trained using SVM.

Every time a node is encountered, the corresponding decision function is evaluated. For an intermediate node, the sign of the result determines which branch of the tree to go: positive sign directs to the left and negative the right, for example. Finally, at a leaf node the sign indicates the class: object or background, and the class label of the pixel which is denoted as $C_\psi(i,j)$ is accordingly obtained: 1 or -1.

Now with the local features, the neighboring features, and the multilayer ψ-learning tool, we are ready to do the tracking job. The most straightforward method is to calculate $C_\psi(i,j)$ for every pixel and then conform the object by all the pixels whose class labels are 1. For the video sequences we experiment with, the maximum number of layers yielded by the multilayer method is three when we choose $\epsilon = 0.05$. So in order to determine the class label of one pixel we just have to evaluate no more than 3 linear functions, which evidently requires low computational complexity. Two examples of the tracking mask after different layers are given in Figs. 7 and 8, respectively.

IV. Classification at the Block Level

It has been shown at the end of the previous section, we can achieve object tracking through pixel-by-pixel classification. Yet in most video frames there is abundant spatial redundancy that we can take advantage of to make the tracking step more efficient. Let p denote a pixel and $N(p,d)$ the set of pixels within a small distance d from p. Due to the spatial redundancy of images, the class labels of p and $N(p,d)$ tend to be consistent with each other. In other words, if p belongs to the object then it is very likely that $N(p,d)$ belong to the object too, except for the pixels lying around the object boundary. Based on this observation, we introduce the concepts of *object blocks* and *background blocks*, and suggest the classification be done at the block level.

Defining an object block as a block whose centering pixel belongs to the object and an background block otherwise, we propose a block-level classification method which is summarized as follows:

1) divide current frame into blocks of size $(2^{N_0}+1)\times(2^{N_0}+1)$ with one pixel overlapping in both vertical and horizontal directions;
2) calculate $\vec{f}_{\text{local}}$ and $\vec{f}_{\text{neighboring}}$ of the centering pixel;
3) evaluate the set of decision functions that have been trained through multilayer ψ-learning to determine the class labels of the centering pixels as well as the labels of blocks;
4) classify all the pixels within the block as object if the block is an object block; otherwise as background.

It would be more common if we have had used the block size $2^{N_0}\times 2^{N_0}$. However, an odd number of pixels is preferred in our approach because of the necessity of "centering pixel." For this

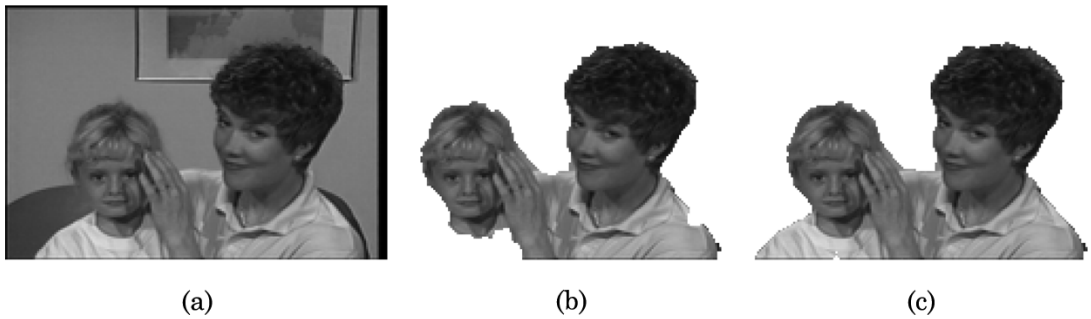


Fig. 7. Tracking results of *Mom & Daughter* sequence after different layers. (a) Original frame where mom and daughter are the object of interest. (b) Tracking result after the first layer. (c) Tracking result after the second layer.

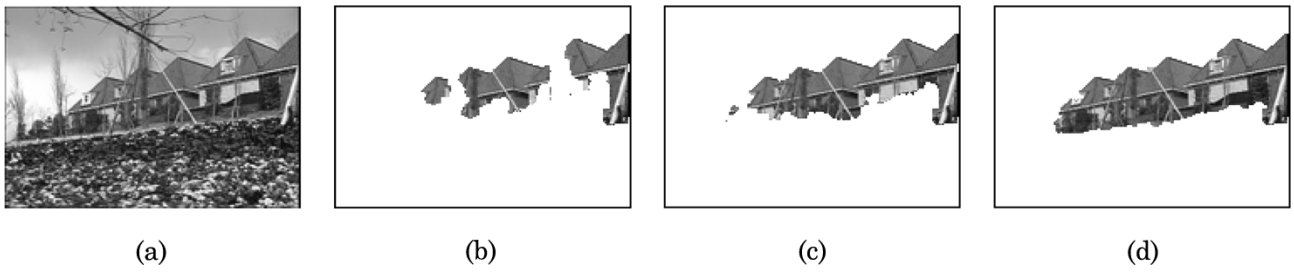


Fig. 8. Tracking results of *Flower Garden* sequences after different layers. (a) Original frame where the houses are the object of interest. (b) Tracking result after the first layer. (c) Tracking result after the second layer. (d) Tracking result after the third layer.

Fig. 9. Tracking results of *Mom* sequence using block-level classification when block size is 9×9. (a) Frame 1. (b) Frame 118. (c) Frame 138.

reason, the block size is chosen to be $(2^{N_0}+1)$ pixels in width and height. As for the introduction of one pixel overlapping, we will give the reason in the next section.

Significant saving in the computational cost is one of the benefits of this block-level representation and classification method. By using the block level classification as explained above, an image of size $M \times N$ is decomposed into exact $L_M \times L_N$ blocks. In other words, we have $M = 2^{N_0} L_M + 1$ and $N = 2^{N_0} L_N + 1$ for some integers L_M and L_N. For that decomposition, the DCT and multilayer classification is computed by $L_M \times L_N$ times instead of $M \times N$ times because we only have to compute $C_\psi(i,j)$ for the centering pixel of each block. Therefore the computation is theoretically reduced by

$$\gamma = 1 - \frac{L_M L_N}{MN} = 1 - \frac{\frac{M-1}{2^{N_0}} \frac{N-1}{2^{N_0}}}{MN} \geq \frac{2^{2N_0}-1}{2^{2N_0}} \qquad (11)$$

in comparison with pixel-by-pixel classification, which converges to 1 quickly with the increase of N_0.

The image size is an important factor to consider when we choose the value of N_0, and usually large images can have relatively large N_0. In the meantime, the size of the object should also be taken into consideration. If the object we intend to track is quite small, a big block size will not be appropriate. Through the experiments, we find $N_0 = 3$ (the block size is 9×9) is a good choice for the video sequences we are working with, and in that case the computation reduction would be around $63/64 \cong 98.4\%$ according to (11).

As shown in Fig. 9, the block-level classification scheme is quite effective even when the object undergoes considerable deformation. The drawback, however, is the stair-like object boundary due to the block effect. Although this coarseness is tolerable in some applications such as target positioning, many others do require pixel-wise accuracy. To address this problem, we propose a pyramid boundary refining algorithm which refines the object boundary in an efficient and scalable way and will be explained in the next section.

V. Pyramid Boundary Refining Algorithm

Fundamentally the pyramid boundary refining algorithm is an iterative process that keeps refining the object boundary until the pixel-wise resolution is reached. During the refining process, a *class map* CM^L is maintained as a binary image that stores the

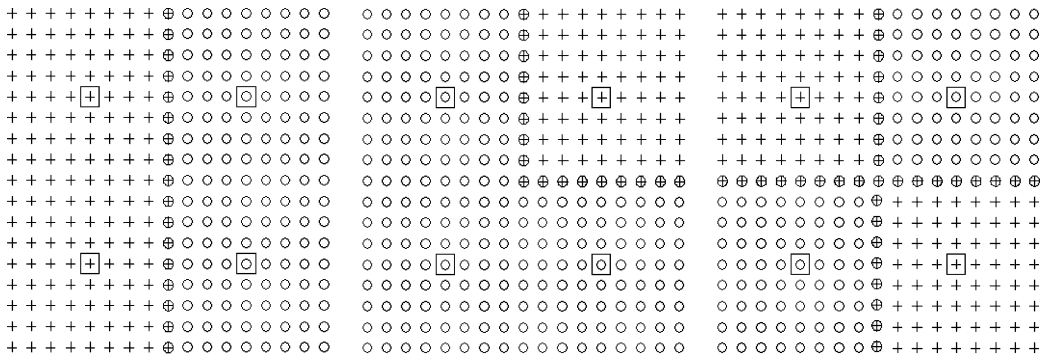


Fig. 10. Considering "+" and "o" as symmetric and also ignoring the orientation, the tiles only exhibit 3 different transition patterns. The points surrounded by □ are the centering pixels of the blocks. (a) Pattern #1. (b) Pattern #2. (c) Pattern #3.

segmentation result obtained after L iterations. The pixel value of CM^L is defined as:

$$\mathrm{CM}^L(i,j) = \begin{cases} -1, & \text{if pixel } p(i,j) \text{ is identified as} \\ & \quad \text{background after } L \text{ iterations;} \\ 1, & \text{otherwise.} \end{cases} \quad (12)$$

Some regions in the class map CM^L are identified as the *boundary zone* (BZ^L) in which the boundary is possibly located and therefore the pixels' class assignments present ambiguity. Initially BZ^0 is a quite large area. In order to reduce the uncertainty about the boundary's actual location, a special group of pixels in the boundary zone, which are named as *boundary seeds* (BSs) and denoted as S^0_{BS}, are selected for class label checking. In other words, their $C_\psi(i,j)$ are computed. According to the newly obtained class labels, the class map is updated such that the block size around the boundary is decreased. Also BZ^0 is reduced to BZ^1 which is only half as large. The similar process continues to increase the boundary resolution until no refinement is needed.

In the following two subsections, we will explain the initialization and iteration steps of the pyramid boundary refining algorithm in detail.

A. Initialization Step

The algorithm starts with the block-level classification discussed in Section IV. By dividing the frame into blocks of size $(2^{N_0}+1)\times(2^{N_0}+1)$, we obtain the initial segmentation result CM^0 as follows:

$$\mathrm{CM}^0(i,j) = C_\psi(i_n, j_n),\ |i-i_n| \le 2^{N_0-1},\ |j-j_n| \le 2^{N_0-1} \quad (13)$$

for all $p(i_n, j_n) \in S_c$, where S_c is the collection of the centering pixels of all blocks.

The next step in initialization is to determine the boundary zone BZ^0. Based on the assumption that the boundary is within the regions that exhibit transitions between object and background blocks, we first identify the *transition areas* in CM^0 that are defined as the union of the *transition blocks* which have at least one eight-connected neighboring block belonging to a different class. Then the transition areas are decomposed into so-called *transition tiles* (TTs), which are rectangular regions containing 2×2 transition blocks. Suppose at this initialization step the block size is 9×9 ($N_0 = 3$), and so the transition tiles are of size 17×17. Some transition tiles in CM^0 are shown in Fig. 10, in which the pixel is portrayed as "o" if it takes value 1, and "+" otherwise. An interesting phenomenon about the transition tiles is that if we consider "+" and "o" as symmetric and further ignore the tiles' orientation there are actually only three distinct transition patterns. As shown in Fig. 10, these three patterns convey different boundary information and thus need to be handled differently. The occurrence of pattern #1 implies a steep or nearly vertically located boundary in the tile, and therefore a rectangular boundary zone is designed as depicted in Fig. 11(a). If the slope of the boundary is relatively moderate, we get pattern #2 and accordingly the boundary zone is conceived as a "L" shape [Fig. 11(b)]. As for pattern #3, the boundary is assumed to be in the middle and form a cross shape [Fig. 11(c)]. By combining the boundary zone in all transition tiles of CM^0, BZ^0 is obtained.

B. Iteration Step

Fig. 12 gives a diagram of the core operations of the iteration step of the proposed refining algorithm. To explain the iteration step more clearly, we use $N_0 = 3$ as an example to show how the algorithm works during the first iteration before giving the general updating equations for CM^L and BZ^L.

After the determination of CM^0 and BZ^0, we have roughly known where the object boundary is. Its actual location, however, is still uncertain. In order to reduce the uncertainty, the pixels that lie *in the middle* of the 2-D boundary zone, which are depicted as △ in Fig. 11, are selected as boundary seeds S^1_{BS} for class label checking. More specifically, S^1_{BS} is constructed as the following:

$$S^1_{\mathrm{BS}} = \{p(i,j) | i = 4m+1, j = 4n+1, \text{ and pixel } p(i,j) \text{falls inside } \mathrm{BZ}^0\} \quad (14)$$

where m and n are positive integers.

Then for each element $s_n(i_n, j_n) \in S^1_{\mathrm{BS}}$, we apply the multilayer classification and determine its class label $C_\psi(i_n, j_n)$, with which CM^1 and BZ^1 can be generated.

1) $\mathrm{CM}^0 \rightarrow \mathrm{CM}^1$

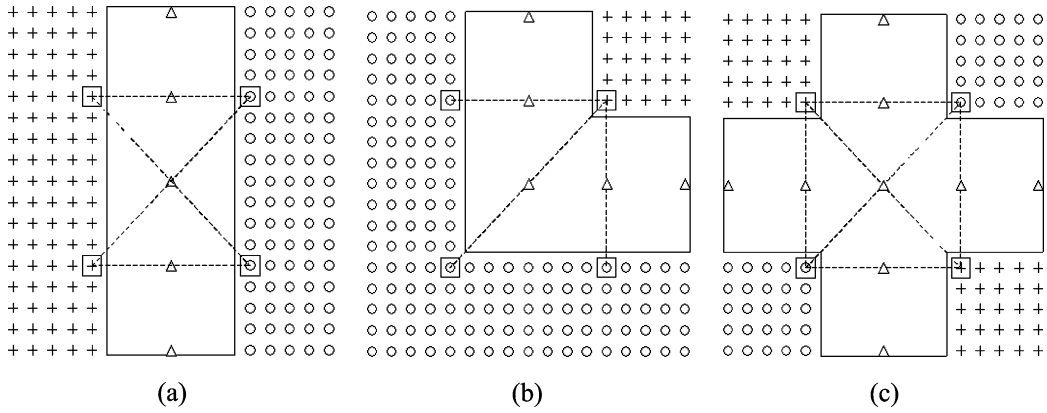


Fig. 11. Boundary zones (the blank areas) and boundary seeds ($\triangle$) determined for different patterns. (a) Pattern #1. (b) Pattern #2. (c) Pattern #3.

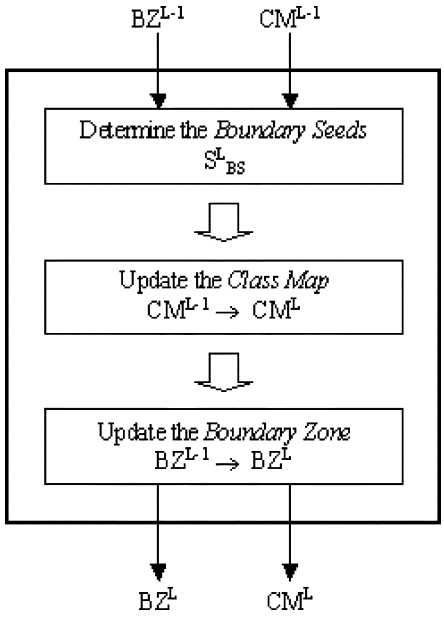


Fig. 12. Diagram of the iteration step of the pyramid boundary refining algorithm.

The class map CM^1 is updated from CM^0 as the following:

$$\mathrm{CM}^1(i,j)=\begin{cases} C_\psi(i_n,j_n), & \text{if } \exists s_n(i_n,j_n)\in S^1_{\mathrm{BS}} \text{ such that} \\ & |i-i_n|\le 2, |j-j_n|\le 2 \\ \mathrm{CM}^1(i,j), & \text{otherwise.} \end{cases} \tag{15}$$

The updating operation of the class map has two important properties. First, it does not affect the pixels falling outside of the boundary zone. As a result the segmentation results only experience small changes around the object boundary area. Secondly, the class labels of the pixels within the boundary zone are updated again at the block level as indicated in (15), hence the extracted object would still has the stair-like boundary at this point. However the block size, which is 5×5 now, is smaller than that of the initialization step. As a result, the block effect shown near the boundary has been reduced as one can see from the tracking mask shown in Fig. 15(c).

2) $\mathrm{BZ}^0 \rightarrow \mathrm{BZ}^1$

Fig. 13 provides an example which considers a transition tiles of CM^0 that shows the pattern #1 to illustrate how the boundary zone can be further reduced according to the newly updated class map CM^1. Suppose the class labels of its boundary seeds are identified as shown in Fig. 13(a) and the class map is updated accordingly [Fig. 13(b)]. Although the boundary searching strategy remains the same, which is to focus on the areas showing the transition between the object and background, the size of the transition tiles becomes smaller. Three new transition tiles, each of which contains only 9×9 pixels now, are highlighted in Fig. 14. In spite of the smaller range, these transition tiles fortunately manifest very similar patterns as discussed in Fig. 11, and therefore their boundary zones and boundary seeds can be determined in a similar way. The union of the boundary zones in all transition tiles of CM^1 constitute the BZ^1, which are shown in Fig. 13(c). As one can see, the area of BZ^1 is nearly half as large as BZ^0 and with it the uncertainty about the actual boundary location is reduced.

Now, with the new class map CM^1 and new boundary zone BZ^1 available, we are ready for the next iteration which will go through the same steps as explained above. In general, the updating equations for each iteration can be summarized as the following.

- Updating S^L_{BS}

$$S^L_{\mathrm{BS}} = \{p(i,j) | i = 2^{N_0-L}m+1, j = 2^{N_0-L}n+1, \text{ and pixel } p(i,j) \text{ falls inside } \mathrm{BZ}^{L-1}\} \tag{16}$$

where m and n are positive integers, and $N_0 = 3$ for 9×9 initial block size.

- Updating CM^L

$$\mathrm{CM}^L(i,j) = \begin{cases} C_\psi(i_n,j_n), & \text{if } \exists s_n(i_n,j_n)\in S^{L-1}_{\mathrm{BS}} \text{ such that} \\ & |i-i_n|\le \lfloor \frac{S_L}{2} \rfloor, |j-j_n|\le \lfloor \frac{S_L}{2} \rfloor; \\ \mathrm{CM}^L(i,j), & \text{otherwise} \end{cases} \tag{17}$$

where $S_L = 2^{N_0-L}$.

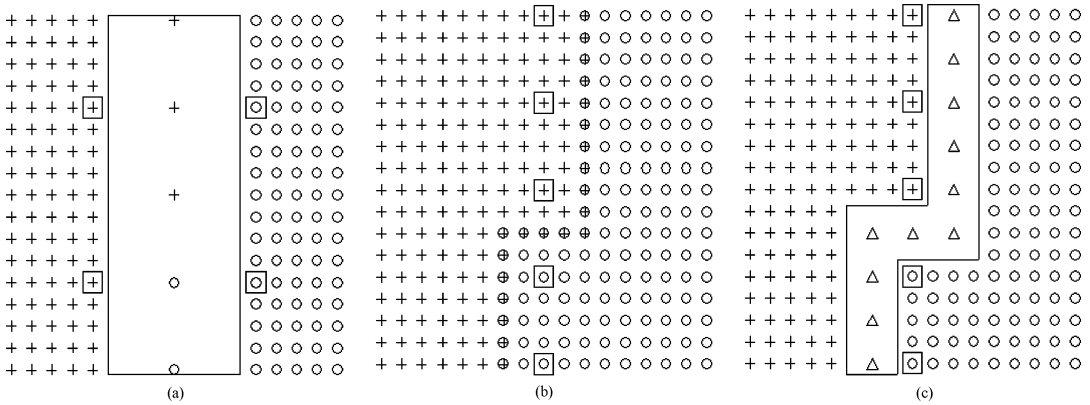


Fig. 13. Example of BZ updating ($N_0 = 3$). (a) Transition area with the BZ^0 and classified boundary seeds. (b) Updated class map CM^1. (c) Updated boundary zone BZ^1 with boundary seeds ($\triangle$) shown in the middle.

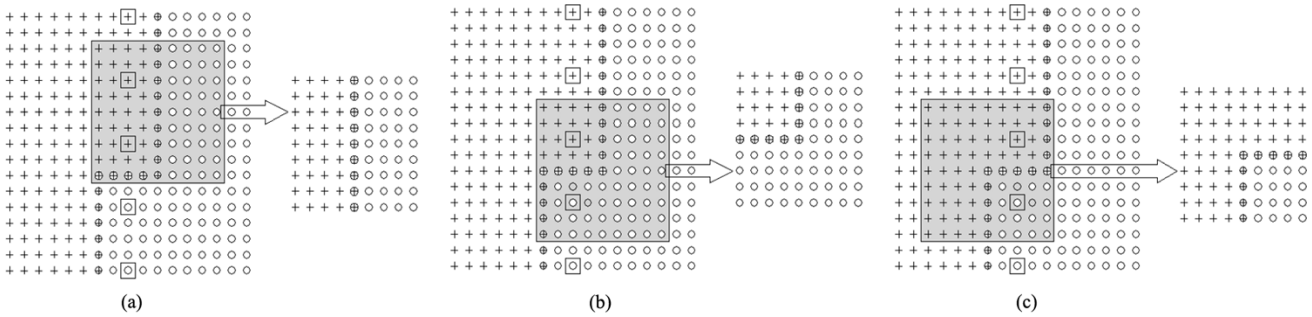


Fig. 14. Three transition tiles in CM^1 ($N_0 = 3$). (a) Transition tile of pattern #1. (b) Transition tile of pattern #2. (c) Transition tile of pattern #3.

- Updating BZ^L

$$\mathrm{BZ}^L = \bigcup_i \mathrm{BZ}\left(\mathrm{TT}_i^L\right) \tag{18}$$

where TT_i^L is the $(2^{N_0-L+1}+1) \times (2^{N_0-L+1}+1)$ transition tiles determined in CM^L, and $\mathrm{BZ}(\mathrm{TT}_i^L)$ denotes the boundary zone in the tile TT_i^L.

As one can see from the updating equations, the larger the L, the smaller the boundary zone BZ^L. When $L = N_0$ the boundary zone is only one pixel width and the $C_\psi(i,j)$ of the boundary seeds does not effect other pixels any more, which means the pixel-wise resolution is reached and therefore the iteration process stops. Fig. 15 shows the segmentation results of the same frame but of different boundary resolutions. The pyramid boundary refining algorithm works so effectively that almost the same tracking results are observed in Fig. 15(e) and Fig. 15(f), which are obtained by the proposed refining algorithm and pixel-by-pixel classification respectively.

It is self-evident that the block effect around the object boundary is eliminated at the expense of the increased computational complexity. As a result the processing speed is surely not as fast as the block-level classification. However because the multilayer classification is carried out only on the pixels selected as the boundary seeds, the run time is reduced to about 1/10 of that of the pixel-by-pixel classification method. Another important property of the proposed refining algorithm is its flexibility. Depending on different applications, the iteration process can stop whenever the desired boundary resolution is reached. Hence by our approach the object boundary is able to be refined in an efficient and scalable manner.

It is worthy pointing out that because of the important role played by the boundary seeds in the proposed refining algorithm, we need to guarantee the integer coordinates for them such that they are available as image pixels. This is the reason why we introduce the one pixel overlapping between the classification blocks in Section IV.

VI. Experimental Results

To test the effectiveness and robustness of the proposed approach, we apply it to five standard MPEG-4 test video sequences, which exhibit certain varieties of temporal and spatial characteristics. These sequences are *Akiyo*, *Mom*, *Mom & Daughter*, *Silent*, and *Flower Garden*. The segmentation and tracking performance is evaluated on both subjective and objective basis. Also the average processing speed per frame is presented to demonstrate the efficiency of our approach.

A. Average Run Time

During the training phase, the unconstrained optimization algorithm proposed in [20] is adopted to minimize the cost function of (5). The parameter C in (5) is empirically chosen as $C = 10^7$ for the first layer, and $C = 10^4$ for the second or higher layers. All experiments are carried out on a Pentium IV 2.5-GHz PC and the average execution time is shown in Table I. Comparing with the pixel-by-pixel classification method whose run time is around 4.65 s for a 176 × 144 frame and 3.969 s for a 180 × 120 frame, the proposed method is about ten times

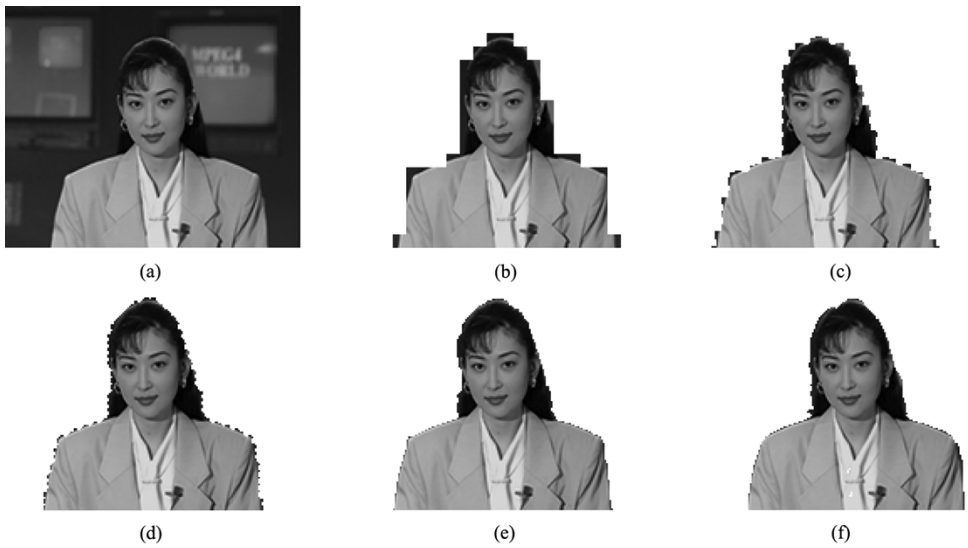


Fig. 15. Tracking results of different boundary resolutions when $N_0 = 3$ (the block size is 9×9). (a) Training frame (frame #1). (b) Tracking result after initial block-level classification. (c) Tracking result after the first iteration of the pyramid boundary refining algorithm. (d) Tracking result after the second iteration of the pyramid boundary refining algorithm. (e) Tracking result after the third iteration (pixel-wise) of the pyramid boundary refining algorithm. (f) Tracking result obtained by pixel-by-pixel classification.

TABLE I
AVERAGE AND STANDARD DEVIATION OF RUN TIME PER FRAME

	Akiyo	Mom	Mom & Daughter	Silent	Flower Garden
average (msec)	396.1	332.5	378.9	413.4	439.3
std (msec)	0.008	0.013	0.014	0.022	0.030
frame size (pixels)	176 x 144	180 x 120	180 x 120	176 x 144	176 x 144

Fig. 16. Tracking results of *Akiyo*. (a) Tracking result of frame 12. (b) Tracking result of frame 134. (c) Tracking result of frame 220.

faster. Run time analysis shows that the feature extraction operation takes nearly 99.7% of the whole run time, and DCT is the major contributor. While the implementation of the algorithm can be further optimized, it should be mentioned that so far we have only considered the intra-frame information for segmentation. The run time is expected to be reduced significantly when the temporal redundancy is utilized, by which one can reduce the number of pixels whose class labels have to be obtained through feature extraction and the multilayer procedures. The potential of that reduction will be discussed later in the Section VII.

B. Subjective Evaluation

Akiyo and *Mom* belong to the typical head-and-should type of sequences. The objects which are the anchored-women in the scene exhibit slow and smooth motion against a stationary background. The performance of our approach is satisfactory even when the objects undergo considerable deformation, as shown in Figs. 16 and 17.

Mom & Daughter is another typical head-and-should type of sequence. However, it exhibits much more complex motion characteristics than *Akiyo* and *Mom*. If mom and daughter are considered as a single object, we have to deal with its noncoherent motions: the mom's head and shoulder move slowly, the daughter stays nearly still for most of the time, and the mom's left hand even disappears in the middle of the sequence. Nevertheless the proposed approach performs well too, as shown in Fig. 18.

The third test sequence is *Silent*, in which a woman makes a number of different gestures. If *Mom & Daughter* is characterized as the combination of "slow motion (mom) along with still motion (daughter) over a simple background," the *Silent* can be considered as the combination of "rapid motion (woman's hands) and slow motion (woman's body) over a textured background." Several tracking results are provided in Fig. 19, showing the effectiveness of our approach.

Among the four sequences tested in the experiments, *Flower Garden* is perhaps the most challenging one. Unlike the previous

Fig. 17. Tracking results of *Mom*. (a) Tracking result of frame 1. (b) Ttracking result of frame 118. (c) Tracking result of frame 138.

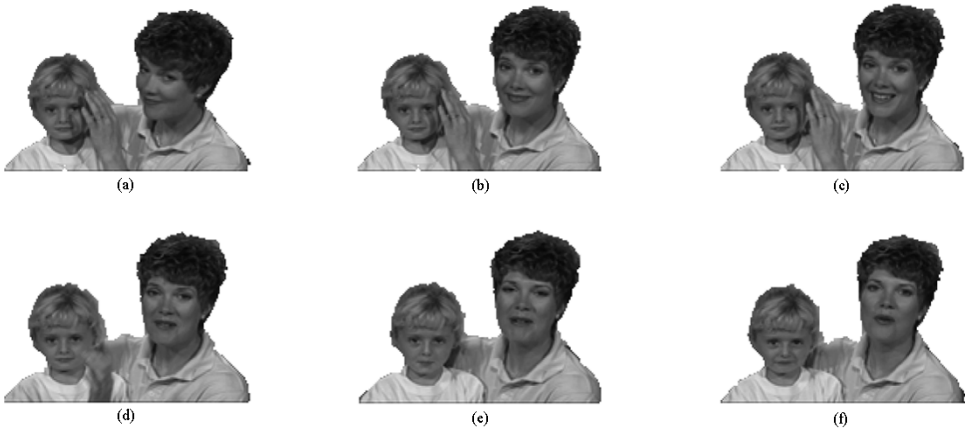


Fig. 18. Tracking results of *Mom & Daughter*. (a) Tracking result of frame 2. (b) Tracking result of frame 11. (c) Tracking result of frame 16. (d) Tracking result of frame 24. (e) Tracking result of frame 43. (f) Tracking result of frame 103.

Fig. 19. Tracking results of *Silent*. (a) Tracking result of frame 33. (b) Tracking result of frame 64. (c) Tracking result of frame 123. (d) Tracking result of frame 145. (e) Tracking result of frame 171. (f) Tracking result of frame 220.

video-conference kind of sequences, it displays a natural scene that is rich of colors and textures with a nonstationary camera. In addition, the houses as the selected object to track are only partially viewable for quite a few frames. The presence of occlusion adds another difficulty to this sequence. Other approaches such as template matching and motion tracking may fail in this case. In contrast, our approach can survive this problem because the un-occluded portion that exhibits the features of the object is still recognizable by our approach. The tracking results of the *Flower Garden* sequence, shown in Fig. 20, demonstrate this advantage. As we can see from Fig. 20, some portions of the houses are uncovered and extracted correctly as the camera moves along. At the same time, the newly occluded area is identified as background and does not appear in the tracking mask. When the occlusion finally disappears, the entire houses emerge as a complete object.

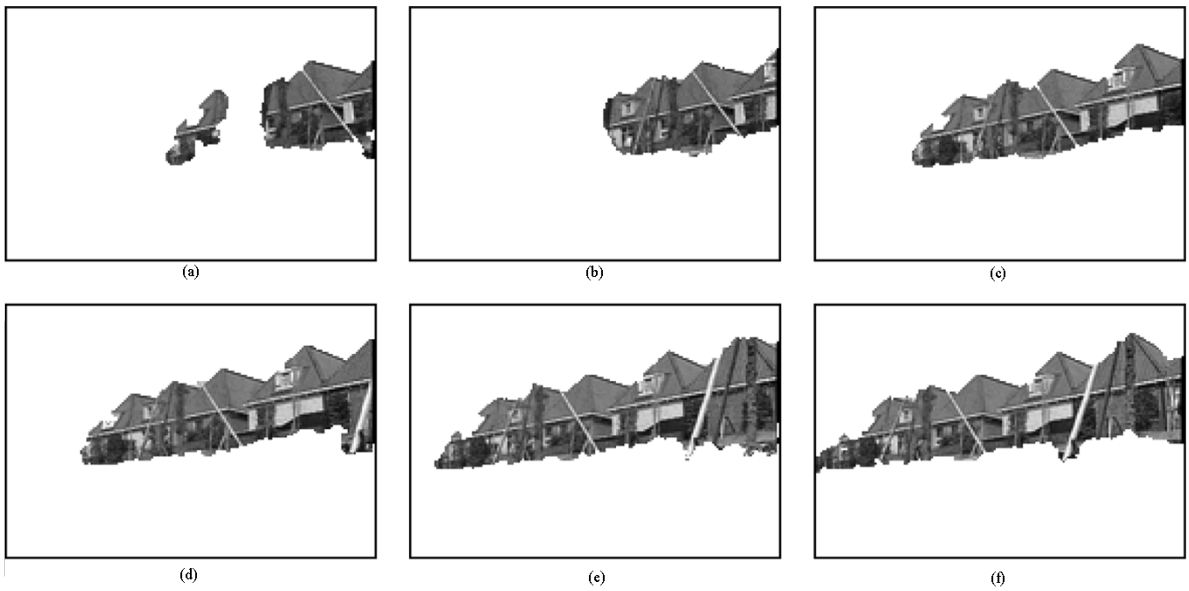


Fig. 20. Tracking results of *Flower Garden*. (a) Tracking result of frame 2. (b) Tracking result of frame 12. (c) Tracking mask of frame 65. (d) Tracking result of frame 86. (e) Tracking result of frame 126. (f) Tracking result of frame 149.

It can also be observed that even when the camera is in motion and the training is only done once, the tracking results are still of good quality. We believe this is because there is no significant change of the video content in the *Flower Garden* so that the information captured by the first frame is rich enough to generate a classifier that is robust for the rest of the sequence. Otherwise, a retraining may be necessary. To do so, a scene change module should be incorporated into the system to detect the change of the video content, which can be measured by the difference of the color or texture histogram between frames, and signal the necessity of the retraining when the difference is significant.

C. Objective Evaluation

In the previous subsection, the proposed segmentation/tracking approach is evaluated on a subjective basis. In this subsection, we introduce an objective criterion to assess the performance quantitatively. Among the criteria available in the literature [36], the one proposed by Wollborn and Mech [37] has been widely adopted. Let $\mathrm{VO}_n^{\mathrm{est}}$ and $\mathrm{VO}_n^{\mathrm{ref}}$ denote the estimated and the reference binary object mask of frame n. Then according to [37], the spatial distortion of $\mathrm{VO}_n^{\mathrm{est}}$ is defined as

$$d\left(\mathrm{VO}_n^{\mathrm{est}}, \mathrm{VO}_n^{\mathrm{ref}}\right) = \frac{\sum_{(x,y)} \mathrm{VO}_n^{\mathrm{est}}(x,y) \bigoplus \mathrm{VO}_n^{\mathrm{ref}}(x,y)}{\sum_{(x,y)} \mathrm{VO}_n^{\mathrm{ref}}(x,y)}, \tag{19}$$

where the $\bigoplus$ is the binary XOR operation [8].

Note that the numerator is equal to the number of wrongly classified pixels while the denominator is the number of pixels per frame. Fundamentally (19) is a measurement of the classification error which makes it very suitable to evaluate our approach.

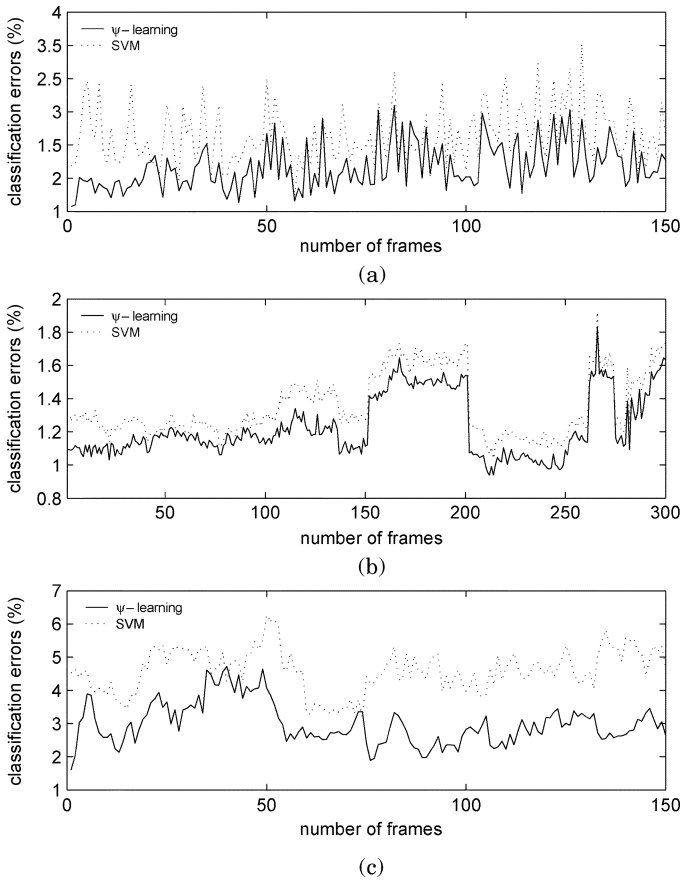


Fig. 21. Segmentation error with respect to frame numbers. (a) *Mom & Daughter* sequence. (b) *Akiyo* sequence. (c) *Flower Garden* sequence.

Fig. 21 shows the segmentation error of the *Mom & Daughter*, *Akiyo*, and *Flower Garden* sequences. Also the error rates using SVM as the classifier are provided for the comparison purpose. Evidently ψ-learning outperforms SVM for the proposed segmentation and tracking approach.

VII. Conclusion

VO extraction, as a prerequisite of the emerging content-based video technologies, is a very important yet very challenging task. In this paper, we present a novel semiautomatic approach that handles single VO extraction as a binary classification problem. By this approach, we are able to overcome some limitations of the conventional tracking methods and deliver an improved performance for various video sequences. The proposed method has following features.

1) A multilayer ψ-learning mechanism is proposed to achieve high classification accuracy even when the sequences contain complicated content.
2) Block-level classification is introduced to deal with the inefficiency of pixel-by-pixel classification.
3) A pyramid boundary refining method is incorporated to obtain the pixel-wise object boundary in a fast and scalable manner.

Experimental results demonstrate that the proposed method can successfully extract the object of interest from video sequences that exhibit various spatial and temporal characteristics. Nevertheless, the object boundaries are not always perfectly located due to the classification error. One possible solution is to extract the edges points that exist within a small distance from the contour of the extracted object and connect them as the refined boundary. More computational cost, of course, has to be paid for this purpose.

The proposed approach only relies on the spatial information. Video sequences, however, provide temporal information which should be useful for object and background separation. Therefore, one of the future research directions is to take advantage of the temporal redundancy between frames to further improve the efficiency of our algorithm. For example, we do not have to go through the operations of DCT and multilayer classification to determine the class label whenever a boundary seed $p(x, y)$ is encountered. Instead, we can first check the difference of the chrominance (or the intensity for grayscale videos) between two consecutive frames and average them in the block centering at $p(x, y)$. If the difference is small which implies little motion around the pixel, the new class label (of the current frame) can be the same as the old one (obtained in the previous frame). By doing so, the number of DCT operation which is the major contributor to the computational complexity of our algorithm will be reduced and so will be the run time.

Another research topic in the future is to extend the proposed approach to multiple object tracking. In analogy to binary classification, an N object tracking problem can be formulated as an $(N + 1)$-category classification problem. That is, one class for the background and N classes for the objects of interest. Most of the mechanisms presented in the paper, such as the block classification and the boundary refining algorithm, are still applicable although additional methods will have to be developed.

References

[1] L. Chiariglione, "MPEG and multimedia communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 5–18, Feb. 1997.

[2] *Overview of the MPEG-4 Version 1 Standard*, ISO-/IEC JTC1/SC29/WG11, Oct. 1997.

[3] D. Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 539–546, Sep. 1998.

[4] I. Kompatsiaris and M. G. Strintzis, "Spatialtemporal segmentation and tracking of objects for visualization of videoconference image sequences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 8, pp. 1388–1403, Dec. 2000.

[5] Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 800–810, Aug. 2001.

[6] 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, no. 5, pp. 776–781, Aug. 2000.

[7] A. Neri, S. Colonnese, G. Russo, and P. Talone, "Automatic moving objects and background separation," *Signal Proces.*, vol. 66, no. 2, pp. 219–232, 1998.

[8] C. Kim and J. N. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 2, pp. 122–129, Feb. 2002.

[9] S. Y. Chien, S. Y. Ma, and L. G. Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, pp. 577–586, Jul. 2002.

[10] A. Alatan, L. Onural, M. Wollborn, R. Mech, E. Tuncel, and T. Sikora, "Image sequence analysis for emerging interactive multimedia services—the European COST 211 framework," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 7, pp. 802–813, Nov. 1998.

[11] 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. 572–584, Sep. 1998.

[12] D. G. Prerz, C. Gu, and M. T. Sun, "Semantic video object extraction using four-band watershed and partition lattice operators," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 5, pp. 603–618, May 2001.

[13] S. Sun, D. R. Haynor, and Y. Kim, "Semiautomatic video object segmentation using VSnake," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 1, pp. 75–82, Jan. 2003.

[14] C. He, J. Dong, Y. F. Zheng, and S. C. Ahalt, "Object tracking using the Gabor wavelet transform and the golden section algorithm," *IEEE Trans. Multimedia*, vol. 4, no. 4, pp. 528–538, Dec. 2002.

[15] Y. Altunbasak and A. M. Tekalp, "Occlusion-adaptive, content-based mesh design and forward tracking," *IEEE Trans. Image Process.*, vol. 6, no. 9, pp. 1270–1280, Sep. 1997.

[16] P. V. Beek, A. M. Tekalp, N. Zhuang, I. Celasun, and M. Xia, "Hierarchical 2-D mesh representation, tracking and compression for object-based video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 2, pp. 353–369, Mar. 1999.

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

[18] Y. Zhong, A. K. Jain, and M. P. Dubuisson-Jolly, "Object tracking using deformable templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 5, pp. 544–549, May 2000.

[19] H. Wang and M. Brady, "Real-time corner detection algorithm for motion estimation," *Image Vis. Comput.*, vol. 13, pp. 695–703, Nov. 1995.

[20] X. Shen, G. Tseng, X. Zhang, and W. H. Wong, "On ψ-learning," *J. Amer. Statist. Assoc.*, vol. 98, no. 463, pp. 724–734, 2003.

[21] A. Doulamis, N. Doulamis, K. Ntalianis, and S. Kollias, "An efficient fully unsupervised video object segmentation scheme using an adaptive neural-network classifier architecture," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 616–630, May 2003.

[22] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst.*, vol. 13, no. 4, pp. 18–28, Jul.–Aug. 1998.

[23] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *Proc. 1997 IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition*, 1997, pp. 130–136.

[24] M. Pontil and A. Verri, "Support vector machines for 3-D object recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 6, pp. 637–646, Jun. 1998.

[25] A. Tefas, C. Kotropoulos, and I. Pitas, "Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 7, pp. 735–746, Jul. 2001.

[26] D. J. Sebald and J. A. Bucklew, "Support vector machine techniques for nonlinear equalization," *IEEE Trans. Signal Process.*, vol. 48, no. 11, pp. 3217–3226, Nov. 2000.

[27] S. Chen, A. K. Samingan, and L. Hanzo, "Support vector machine multiuser receiver for DS-CDMA signals in multipath channels," *IEEE Trans. Neural Netw.*, vol. 12, no. 3, pp. 604–611, May 2001.
[28] T. S. Huang, X. S. Zhou, M. Nakazato, Y. Wu, and I. Cohen, "Learning in content-based image retrieval," in *Proc. 2nd Int. Conf. Development and Learning*, Jun. 2002, pp. 155–162.
[29] S. Tong and E. Change, "Support vector machine active learning for image retrieval," in *Proc. ACM Int. Conf. Multimedia*, Oct. 2001, pp. 107–118.
[30] G. D. Guo, A. K. Jain, W. Y. Ma, and H. J. Zhang, "Learning similarity measure for natural image retrieval with relevance feedback," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 811–820, Jul. 2002.
[31] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.
[32] C. Cortes and V. N. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[33] V. N. Vapnik, *The Natural of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[34] R. Cappelli, D. Maio, and D. Maltoni, "Multiple KL for pattern representation and classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 9, pp. 977–996, Sep. 2001.
[35] J. Weng, "Cresceptron and SHOSLIF: toward comprehensive visual learning," in *Early Visual Learning*, S. K. Nayar and T. Poggio, Eds. Oxford, U.K.: Oxford Univ. Press, 1996.
[36] P. L. Correia and F. Pereira, "Objective evaluation of video segmentation quality," *IEEE Trans. Image Process.*, vol. 12, no. 2, pp. 186–200, Feb. 2003.
[37] *Refined procedure for objective evaluation of video generation algorithms*, ISO/IEC JTC1/SC29/WG11 M3448, Mar. 1998.

Yi Liu (S'03) received the B.S. and M.S. degrees from the Department of Information Science and Electronics Engineering, Zhejiang University, Hangzhou, China, in 1997 and 2000, respectively. She is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, Ohio State University, Columbus.

Her research interests include machine learning, pattern recognition, and their applications in the area of multimedia signal processing.

Yuan F. Zheng (S'82–M'86–SM'90–F'97) received the B.S. degree from Tsinghua University, Beijing, China, in 1970, and the M.S. and Ph.D. degrees in electrical engineering from The Ohio State University, Columbus, in 1980 and 1984, respectively.

From 1984 to 1989, he was with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC. Since August 1989, he has been with The Ohio State University, where he is a Professor in the Department of Electrical Engineering. During 2004–2005, he is on leave at the Shanghai Jiao Tong University, Shanghai, China. His research interests include two aspects. One is in wavelet transform for image and video compression for Internet and satellite communications. Current efforts focus on content-based compression, 3-D wavelet transformation, video object tracking, and content-based retransmission in Internet communications. The other is in robotics, which includes robots for biological applications, multiple robots coordination, legged robots, human–robot coordination, and personal robotics.

Prof. Zheng was an Associate Editor of the *International Journal of Intelligent Automation and Soft Computing*. He was Vice-President for Technical Affairs of the IEEE Robotics and Automation Society from 1996 to 1999. He was an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION between 1995–1997. He was the Program Chair of the 1999 IEEE International Conference on Robotics and Automation, held in Detroit, MI, on May 10–15, 1999. Professor Zheng received the Presidential Young Investigator Award in 1986.