計測自動制御学会東北支部 第 223 回研究集会 (2005.7.27)
資料番号 223-7

Combining Spatial, Color and Motion Features for Video Object Segmentation

○ Hariadi Mochamad*, Norihito Numa*, Hui Chien Loy* and Takafumi Aoki*

*Graduate School of Information Sciences, Tohoku University

Keywords: video object segmentation, motion vector, MPEG, learning vector quantization, phase-only correlation

Contact : Graduate School of Information Sciences, Tohoku University
6-6-05, Aramaki Aza Aoba Sendai-Shi, 980-8579 Japan
Phone: +81-22-795-7169, Fax: +81-22-263-9308,
E-mail: mochar@aoki.ecei.tohoku.ac.jp

1. Introduction

The rapid growth in the number of multimedia applications has resulted in new demands of multimedia data processing, such as the flexibility to access the content of video data in the semantic level. These demands are supported by the emerging video standards MPEG-4 and MPEG-7 which are not only concentrating on efficient compression methods, but also provide the user with greater flexibility to access and manipulate the content of multimedia data (i.e. the video contents).

In order to obtain the content of a video sequence, an input video sequence must first be segmented into an appropriate set of arbitrarily shaped semantic objects. The problem is how to extract the shape information of the semantic object inside the video sequences. Many video object segmentation algorithms have been developed over the years. These algorithms can be broadly classified into two types i.e., *automatic segmentation* (e.g., [1]), and *semi-automatic* segmentation (e.g.[2]). The automatic segmentation method tracks the semantic object from *a-priori* information such as, color texture and motion. Until now, there is no guarantee that any of the automatic segmentation result will be semantically meaningful, since a semantically meaningful object may have multiple colors, textures and motions [3]. On the other hand, the semi-automatic method offers more feasible solutions. This method involves human assistance to provide the knowledge of semantically meaningful object on an initial frame, then a tracking mechanism is employed to segment the semantic object on the successive frames.

We have proposed a framework of semi-automatic video object segmentation based on Learning Vector Quantization (LVQ) [4]. In our previous work, we presented the video object segmentation using the combination of spatial feature (pixel position coordinates) and color features (YUV) for LVQ-based video object segmentation. The problem of the previous work is that the algorithm is not ro-

bust for some cases such as the object color is similar to its background (the object tree in Fig. 1) and the object is moving too fast (the object player's hand in Fig. 2). Addressing to these problem, we improve our semi-automatic video object segmentation by combining motion feature with spatial and color features for classifying each pixel into object or background classes. To provide the motion feature we employ Phase-Only Correlation (POC) function [5, 6] – an image matching technique using the phase components in 2-Dimensional Discrete Fourier Transforms (2-D DFT)of given images. Due to the complexities of motion in a video sequence, to estimate motion requires the high-accuracy matching of small image blocks. We employ hierarchical search strategy to find the best matching image block. Experimental observation shows that POC-based hierarchical searchthis method exhibit better matching performance than the method using the combination of SAD (Sum of Absolute Differences) and full search strategy in general [7].

Our experiments with MPEG-4 standard video sequences demonstrate that the proposed method is robust to video sequences which are not suited with our previous work.

2. Fundamentals of the Algorithm

This section gives a brief explanation of our previous work where the video object segmentation method only used the spatial and color features. Next, we explain the phase-based image matching and hierarchical search strategy to find the motion vector between two adjacent frames. The last part explains how to combine the spatial, color and motion features into one framework.

2.1 Previous Work

We have developed a LVQ based video object segmentation using spatial and color features [4]. We use a 5-D feature vector $x \in \Re^5$ to represent each pixel of a video frame. Pixel coordinates (s_1, s_2) and YUV color components (y, u, v) are integrated together to form a single feature vector. To prevent domination by any one feature, all features should be normalized. The Y component of YUV color space y is normalized as

$$\tilde{y} = \frac{y - \mu(y)}{\sigma(y)}, \tag{1}$$

where $\tilde{y}$ is the normalization of y, $\mu(y)$ is the mean of y and $\sigma(y)$ is the standard deviation of y. This normalization is also done for u, v, s_1 and s_2 features.

While this approach is better than using pixel position only or color information only, it is not robust enough to handle a wide variety of video sequences. This is due to the different properties of spatial and color information. We introduce a parameter K to adjust the balance between pixel coordinates and YUV color components to form a single 5-D feature vector:

$$\boldsymbol{x} = (\tilde{s_1}, \tilde{s_2}, K\tilde{y}, K\tilde{u}, K\tilde{v}). \tag{2}$$

We create segmentation result by employing Vector Quantization (VQ) to 5-D feature vectors, and classify all pixels into object or background classes. The LVQ algorithm is then use to provide optimal determination of the appropriate set of codebook vectors and their classes that correctly approximates the segmentation result

The problem of the previous work is that the algorithm is not robust for some cases such as the condition in Fig. 1 and Fig. 2. In Fig. 1), the object (tree) color is similar to its background (home

Fig. 1 The object (tree) has similar color with background.

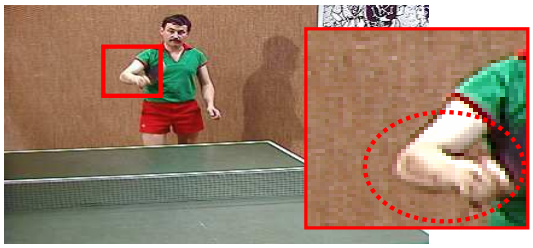

Fig. 2 The movement of object (player's hand) is too fast.

and small trees) and in Fig. 2 the player's hand is moving too fast. Another problem is that we have to adjust parameter K to achieve good segmentation results. We need to try a set of parameter K to decide the best K for each video sequence. Furthermore, the optimal parameter K is obtain by evaluating the last frame error or the error of the next key-frame [4]. This method is not feasible and take a lot of time. In our proposed algorithm, we found that by moving the codebook vectors spatial components based on their valid motion vectors, we can omit parameter K, thus we do not depend on the next key-frame to define the best K again.

2.2 Motion Estimation using Phase-Based Image Matching

Consider two $N_1 \times N_2$ images, $f(n_1, n_2)$ and $g(n_1, n_2)$, where we assume that the index ranges are $n_1 = -M_1, \cdots, M_1$ and $n_2 = -M_2, \cdots, M_2$ for mathematical simplicity, and hence $N_1 = 2M_1 + 1$ and $N_2 = 2M_2 + 1$. Let $F(k_1, k_2)$ and $G(k_1, k_2)$ denote the 2D Discrete Fourier Transforms (2D DFTs) of the two images.

Refer to the original POC function, the cross-phase spectrum (or normalized cross spectrum) $\hat{R}(k_1, k_2)$ is defined as

$$\begin{aligned} \hat{R}(k_1, k_2) &= \frac{F(k_1, k_2)\overline{G(k_1, k_2)}}{|F(k_1, k_2)\overline{G(k_1, k_2)}|} \\ &= e^{j\theta(k_1, k_2)}, \end{aligned} \tag{3}$$

where $\overline{G(k_1, k_2)}$ denotes the complex conjugate of $G(k_1, k_2)$ and $\theta(k_1, k_2) = \theta_F(k_1, k_2) - \theta_G(k_1, k_2)$. The Phase-Only Correlation (POC) function $\hat{r}(n_1, n_2)$ is the 2D Inverse Discrete Fourier Transform (2D IDFT) of $\hat{R}(k_1, k_2)$ and is given by

$$\hat{r}(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1 k_2} \hat{R}(k_1, k_2) W_{N_1}^{-k_1 n_1} W_{N_2}^{-k_2 n_2} \tag{4}$$

where $\sum_{k_1 k_2}$ denotes $\sum_{k_1=-M_1}^{M_1} \sum_{k_2=-M_2}^{M_2}$.

Now consider $f_c(x_1, x_2)$ as a 2D image defined in continuous space with real-number indices x_1 and x_2. Let δ_1 and δ_2 represent sub-pixel displacements of $f_c(x_1, x_2)$ to x_1 and x_2 directions, respectively. So, the displaced image can be represented as $f_c(x_1 - \delta_1, x_2 - \delta_2)$. Assume that $f(n_1, n_2)$ and $g(n_1, n_2)$ are spatially sampled images of $f_c(x_1, x_2)$ and $f_c(x_1 - \delta_1, x_2 - \delta_2)$, and are defined as

$$\begin{aligned} f(n_1, n_2) &= f_c(x_1, x_2)|_{x_1 = n_1 T_1, x_2 = n_2 T_2}, \\ g(n_1, n_2) &= f_c(x_1 - \delta_1, x_2 - \delta_2)|_{x_1 = n_1 T_1, x_2 = n_2 T_2}, \end{aligned}$$

where T_1 and T_2 are the spatial sampling intervals, and index ranges are given by $n_1 = -M_1, \cdots, M_1$ and $n_2 = -M_2, \cdots, M_2$. The POC function $\hat{r}(n_1, n_2)$ between $f(n_1, n_2)$ and $g(n_1, n_2)$ will be given by

$$\begin{aligned} &\hat{r}(n_1, n_2) \\ &\simeq \frac{\alpha}{N_1 N_2} \frac{\sin\{\pi(n_1 + \delta_1)\}}{\sin\{\frac{\pi}{N_1}(n_1 + \delta_1)\}} \frac{\sin\{\pi(n_2 + \delta_2)\}}{\sin\{\frac{\pi}{N_2}(n_2 + \delta_2)\}} \end{aligned} \tag{5}$$

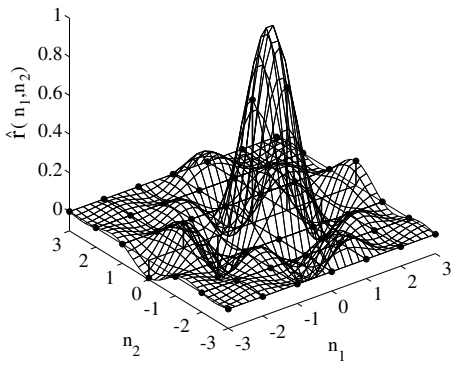

Fig. 3 Function fitting for estimating the peak position.

where $\alpha \leqq 1$.

We use eq. (5) — the closed-form peak model of the POC function — directly for estimating the peak position by function fitting. By calculating the POC function for two images $f(n_1, n_2)$ and $g(n_1, n_2)$, we can obtain a data array of $\hat{r}(n_1, n_2)$ for each discrete index (n_1, n_2), where $n_1 = -M_1, \cdots, M_1$ and $n_2 = -M_2, \cdots, M_2$. It is possible to find the location of the peak that may exist between image pixels by fitting the function (5) to the calculated data array around the correlation peak, where α, δ_1, and δ_2 are fitting parameters. Figure 3 shows an example, where eq. (5) is fitted to the data array of $\hat{r}(n_1, n_2)$. A spectral weighting technique and a windowing technique are used to reduce aliasing and noise effects and to overcome the periodicity effect in DFT, respectively.

2.3 Hierarchical Search Method

The hierarchical search method employs POC function above. Then, this algorithm employs (i) a coarse-to-fine strategy using image pyramids for robust correspondence search with POC function, and (ii) a sub-pixel window alignment technique for finding a pair of corresponding points with sub-pixel displacement accuracy. In the first stage (i), we estimate the the corresponding points search with pixel-level accuracy using phase-based hierarchical search with coarse-to-fine strategy. Thus, the estimation error becomes less than 1 pixel for every corresponding point. The second stage (i-i) of the algorithm is to recursively improve the sub-pixel accuracy of corresponding points by adjusting the location of the window function with sub-pixel accuracy. As a result, the coordinates of corresponding points are obtained with sub-pixel accuracy. (See [6] for deeper discussion and analysis of for subsection 2.3 and 2.2).

2.4 Combining Spatial, Color and Motion Features

Our proposed method combine the motion vector into the 5-D feature vector from our previous method, in which the feature components are pixel coordinates and color in YUV color space. Our basic idea is to classify each pixel not only base on color and coordinate features, but also base on motion feature. In this case our proposed algorithm assumes that every pixel of the frame has its motion vector. However, since it is not feasible to do motion estimation for every pixel, we take a set of reference points. Let frame I and J as two adjacent frames, set the reference points $\boldsymbol{m} = (m_1, m_2)$ in I, where H is the amount of reference pixels and $\boldsymbol{m}_h (h = 1, 2, \cdots, H)$. Find the motion vectors $\boldsymbol{d}(\boldsymbol{m})$ using POC based hierarchical search motion estimation for all reference points $\boldsymbol{m}_h$. Given a feature vectors $\boldsymbol{u} = (u_1, u_2)$ in image I, we find the reference point $\boldsymbol{m}_b$ that is the nearest to $\boldsymbol{u}$ in terms

of Euclidean distance, where

$$b = \arg\min_{h} \{\|\boldsymbol{u} - \boldsymbol{m}_h\|\}, \quad (6)$$

Thus, the motion vector of $\boldsymbol{u}$ is given by the motion of $\boldsymbol{m}_b$ i.e. $\boldsymbol{d}(\boldsymbol{m}_b)$.

To combine spatial, color and motion features, a 6-D feature vector is created in which the 6th component is normalized motion vector $\boldsymbol{d}(\tilde{\boldsymbol{m}})$, and the spatial components are $\tilde{s_1}$ and $\tilde{s_2}$. Due to the coarse result caused by the usage of reference pixel for motion vector, we need to introduce a weight parameter τ to motion vector. Thus, we form a 6-D feature vector with τ for motion vector adjustment as $\boldsymbol{x} = (\tilde{y}, \tilde{u}, \tilde{v}, \tilde{s_1}, \tilde{s_2}, \tau\boldsymbol{d}(\tilde{\boldsymbol{m}}))$.

To classify each pixel using pixel-wise classification by Vector Quantization (VQ), we also need to find the motion vectors of codebook vectors spatial components. Let the spatial components of codebook vectors be $\boldsymbol{j} = (j_1, j_2)$ in I. Find the motion vector of $\boldsymbol{j}$ using POC based hierarchical search motion estimation and move $\boldsymbol{j}$ with respect to its motion vector as follows. Let $\boldsymbol{j}$ as the codebook vector spatial feature in frame I and $\boldsymbol{d}(\boldsymbol{j})$ as its motion vector. Then the updated spatial components of codebook vector frame J is

$$\boldsymbol{q}(\boldsymbol{j}) = \boldsymbol{j} + \boldsymbol{d}(\boldsymbol{j}) \quad (7)$$

VQ uses P codebook vectors $\boldsymbol{p}_i \in \Re^6 (i = 1, 2, \cdots, P)$ to group the feature vectors $\boldsymbol{x} \in \Re^6$ (corresponding to pixels of a frame) into P cluster. In our case, each codebook vector $\boldsymbol{p}_i$ has its own class label (object or background). We classify each pixel of a frame into object class or background class based on VQ as follows. Given a feature vector $\boldsymbol{x}$, we find the codebook vector $\boldsymbol{p}_c$ that is nearest to $\boldsymbol{x}$ in terms of Euclidean distance, where

$$c = \arg\min_{i} \{\|\boldsymbol{x} - \boldsymbol{p}_i\|\}. \quad (8)$$

in which $\boldsymbol{x} = (\tilde{y}, \tilde{u}, \tilde{v}, \tilde{s_1}, \tilde{s_2}, \tau\boldsymbol{d}(\tilde{\boldsymbol{m}}))$ and $\boldsymbol{p}_i = (\tilde{y}, \tilde{u}, \tilde{v}, q(\tilde{j})_1, q(\tilde{j})_2, \tau\boldsymbol{d}(\tilde{\boldsymbol{j}}))$.

The class of the pixel is given by the class of its nearest codebook vector $\boldsymbol{p}_c$. In other words, the P codebook vectors partition (or quantize) the 6-D vector space into P nearest neighbor regions known as *Voronoi regions*. Within the Voronoi region of $\boldsymbol{p}_i$, the feature vectors (pixels) are labeled with the class of $\boldsymbol{p}_i$.

Using the function fitting eq. (5) we get the POC peak level α. To define the validity of a motion vector, we set a certain threshold value T for α as follows:

$$\alpha \quad \begin{cases} \geq T & \text{motion vector is valid} \\ < T & \text{motion vector is not valid} \end{cases} \quad (9)$$

This is done to both reference pixels and codebook vectors motion vectors.

The validity of motion vector is implemented on LVQ algorithm. The LVQ algorithm starts with the random initialization of P codebook vectors $\boldsymbol{p}_i$ $(i = 1, 2, \cdots, P)$. In practice, this is done by choosing P pixels from the video frame randomly as initial codebook vectors. Each codebook vector is given the majority class of pixels within its Voronoi region.

For supervised learning of LVQ, an individual learning rate $\gamma_i(t)$ is assigned to each codebook vector $\boldsymbol{p}_i$. Given a feature vector $\boldsymbol{x}$ whose classification is known in advance, if the motion vector is not valid then the following formula is used to update the nearest codebook vector $\boldsymbol{p}_c$:

$$\boldsymbol{p}_c(t+1) = \boldsymbol{p}_c(t) + s(t)\gamma_c(t)\left[\boldsymbol{x}(t) - \boldsymbol{p}_c(t)\right], \quad (10)$$

where t and $\gamma_c(t)$ denote the training step and

learning rate for $\boldsymbol{p}_c$, respectively, and

$$s(t) = \begin{cases} +1 & \text{if } x(t) \text{ and } p_c \\ & \text{belong to the same class,} \\ -1 & \text{if } x(t) \text{ and } p_c \\ & \text{belong to different classes.} \end{cases} \quad (11)$$

The other codebook vectors $p_i (i \neq c)$ remain the same. Hence, if the class label of p_c matches that of x, then p_c moves toward x. Otherwise, it moves away from x. This training algorithm is called LVQ1. For fast convergence of eq.(11), we can optimize the learning rate $\gamma_c(t)$ using the following equation:

$$\gamma_c(t) = \frac{\gamma_c(t-1)}{1 + s(t)\gamma_c(t-1)}. \quad (12)$$

The combination of LVQ1 with this optimization formula is called Optimized Learning Vector Quantization 1 (OLVQ1) [8]. During training of the codebook vectors, the supervised learning process of OLVQ1 is repeated for many training steps changing the feature vector x each time to determine the optimal set of codebook vectors.

3. Video Object Segmentation System

This section describes the video object segmentation framework for a temporal video segment consisting of many frames. The system flowchart is shown in Fig. 4. The LVQ codebook vectors are trained in the key frame to approximate the object shape as described in the previous section. For the subsequent frames, the object is segmented automatically. For each frame, the LVQ codebook vectors obtained in its previous frame are used to classify the pixels of the current frame into object or background. The classified pixels are then used as training data for supervised learning to update the LVQ codebook vectors for segmenting the next frame, hence propagating the semantic information. This process is repeated until the last frame. Described below are the steps of the flowchart in Fig. 4.

A. Manually define the object of interest

The semantic object is manually defined with user assistance in a particular key frame. Each pixel of the key frame is manually classified as either "object" or "background". This manual segmentation provides the initial training data for the LVQ codebook vectors in Step D.

B. Normalize spatial and color feature vectors

In this step, Each pixel of the key frame is represented by a 6-D feature vector. In this step, motion feature has not been included yet. Therefore, we set the motion vector to 0 value. To reduce domination of the feature vector by any one component, each component is normalized by parameters μ and σ, as described in 2.1.

C. Initialize LVQ codebook vectors

Initialization of codebook vectors is done by randomly choosing P pixels from the key frame as codebook vectors. Each codebook vector is assigned the majority class label of pixels within its Voronoi region, as explained in subsection 2.4.

D. Train LVQ codebook vectors

Training of codebook vectors is performed to learn the shape of the segmented object. The codebook vectors are trained using OLVQ1 algorithm described in subsection 2.4. We use 20000 total training steps and the initial learning rate $\gamma_i(0)$ is 0.4 for all codebook vectors p_i. To save computation time, only codebook vectors which do not have valid motion vector are trained. However, to update the color information, we train all codebook vectors for every

4 or 5 frames.

E. Get the next frame

The next frame in the temporal segment is loaded.

F. Set reference pixels for estimating motion

Take a set of reference points in image I. In our experiments, we take a reference pixel for every 8 pixels.

G. Estimating motion for reference pixels

Estimating motion for every reference pixel by searching the corresponding point in sub-pixel level in frame J using POC-based hierarchical search method (subsection 2.2).

H. Estimating motion for codebook vectors spatial components

Estimating motion for every codebook vector spatial components by searching the corresponding point in sub-pixel level in frame J using POC-based hierarchical search method (subsection 2.2).

I. Move codebook vectors spatial components with respect to its motion vectors

In this step, we only use the valid motion vector by setting a threshold value for POC peak value α (eq:11). In our experiment, we set the threshold to 0.5. If the α is below $0,5$ then we consider the motion vector to 0 (no movement).

J. Normalize all feature vectors

In this step, the motion feature is included as described in eq. 2. The feature vectors of the loaded video frame are normalized with parameters μ, σ and τ, as described in subsection 2.1.

K. Pixel-wise classification by VQ

Pixel-wise classification of the video frame into object class or background class is done to create the segmentation result. The pixel wise classification by VQ, as described in subsection 2.4, is carried out using the codebook vectors trained in the previous frame. Each pixel is labeled object or background depending on the class of its nearest codebook vector.

L. Removal of segmentation noise

Color similarities between the background and the object may sometimes result in segmentation noise in the form of small, scattered disjoint regions. This is because the projection of a 6-D Voronoi region onto 2-D image plane is not necessarily continuous. To reduce this noise, we introduce a post-processing step using median filtering for removing the small regions. Thus, we obtain the segmented object.

M. End of temporal segment?

The algorithm terminates if the end of the temporal segment is reached. Otherwise, the segmentation result of Step L is used to update the LVQ codebook vectors for classifying the next frame (repeat from steps D to M).

In practice, to reduce computation time, instead of using all the pixels of the video frame, we employ a rectangular window that encloses the object of interest by a small margin. The size of the margin is dependent on the expected movement of the object between frames. In our experiments, we used a margin size of 32 pixels.

4. Experiment and Evaluation

In our experiments, we evaluate our proposed algorithm using the following MPEG standard test video sequences: *Flower Garden*, *Horse Riding* and *Table Tennis*. The object of interest is manually defined in the first frame, and then the proposed algorithm is used to automatically segment the object. The number of codebook vectors depends on the size of the object to be segmented. In our ex-

periments, we employ one codebook vector for every 128 pixels of the rectangular window (described at the end of Section 3) that encloses the object of interest. Thus, a rectangular window of size 256 $\times$ 256 will require 512 codebook vectors. The segmentation accuracy is evaluated by comparing the segmentation result of our proposed algorithm with that of manual segmentation [1]. The segmentation error for each frame is given by the following formula:

$$\text{error} = \frac{\text{total number of misclassified pixels}}{\text{total number of pixels in a frame}} \quad (13)$$

Figure 5, 6, and 7 show the error rate comparison between previous algorithm (without motion) and proposed algorithm (with motion).

5. Conclusion and Future Work

This paper proposed a framework of video object segmentation using spatial (pixel position), color (YUV) and motion features. We use phase-based image matching with hierarchical search method to provide motion vectors. Our proposed approach is more robust to video sequences which have similar color between object and background, and the movement of video object is too fast. We found that by moving all codebook vectors spatial components with respect to their motion vectors, we can omit adjusting parameter K for weigh adjustment of color feature. In the LVQ training step, we do not train codebook vectors with valid motion vectors for 4 until 5 subsequence frames, which will reduced computation time. Experimental results demonstrate that our proposed algorithm gives more robust segmentation than the previous method. Nevertheless, some motion vectors with low POC peak level ($T < 0.5$) occur. These motion vectors are consider as not valid motion vectors which will reduce the performance of segmentation.

Our next plan is to develop an algorithm to decide the replacement value for not valid motion vector. We also plan to implement our algorithm into real worlds applications.

[1] We manually segmented all frames of each video sequence that was used in our experiments

References

1) X. H., Y. A.A., and K. M.R., "Automatic moving object extraction for content-based applications," IEEE Trans. Circuits Syst. Video Technol., vol.14 no. 6, June 2004.

2) S. Sun, D. Haynor, and Y. Kim, "Semiautomatic video object segmentation using vsnakes," IEEE Trans. Circuits Syst. Video Technol., vol.13 no. 1, Jan. 2003.

3) A.C. Bovik, "The hand book of image and video processing," Academic Press Limited, 1st edition, May 1998.

4) H. Mochamad, H.C. Loy, and T. Aoki, "Semiautomatic video object segmentation using LVQ with color and spatial features," IEICE Trans. Inf. Syst. Special Sect. on Recent Advances in Circuits and Systems, vol.E88-D no. 7, July 2005.

5) K. Takita, T. Aoki, Y. Sasaki, T. Higuchi, and K. Kobayashi, "High-accuracy subpixel image registration based on phase-only correlation," IEICE Trans. on Fundamentals, vol.E86-A No.8, Aug. 2003.

6) K. Takita, M. Muquit, T. Aoki, and T. Higuchi, "A sub-pixel correspondence search technique for computer vision applications," IEICE Trans. on Fundamentals, vol.E87-A No.8, Aug. 2004.

7) H.C. Loy and T. Aoki, "Robust motion estimation for video sequences based on phase-only correlation," The Sixth IASTED Int. Conf. Signal and Image Proc., Aug. 2004.

8) T. Kohonen, "Self-Organizing Maps 3rd edition," Springer-Verlag, Berlin Heidelberg, Germany, 2001.

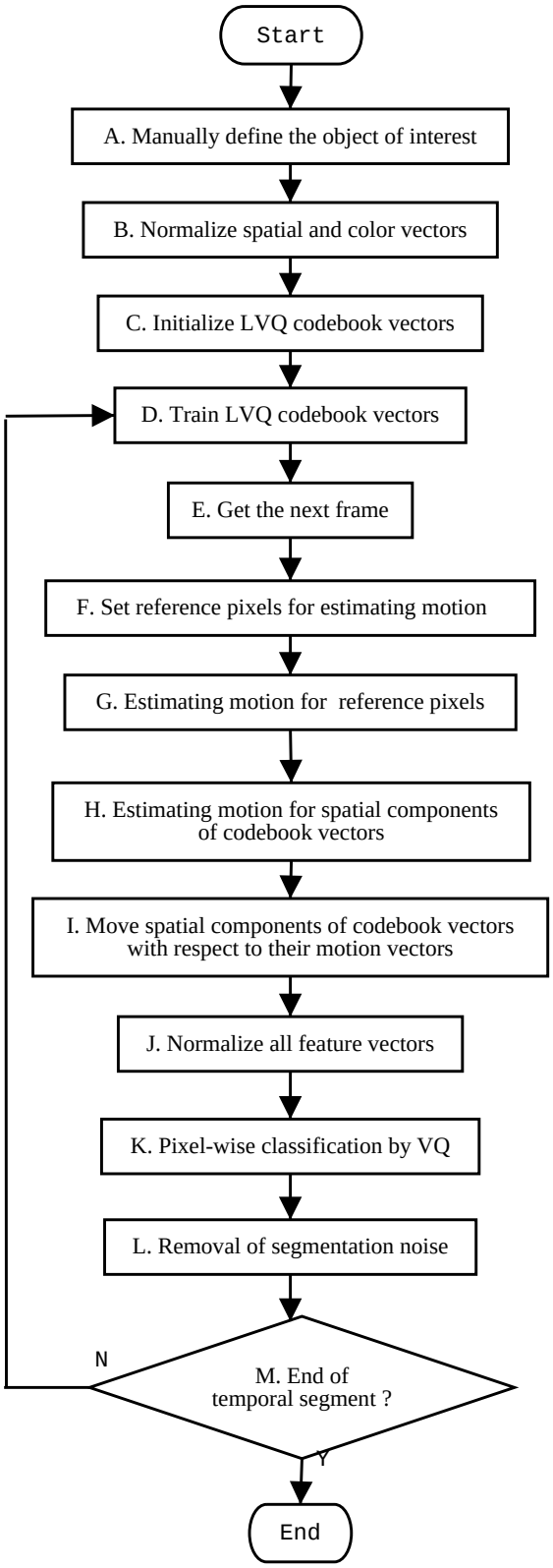

Fig. 4 Video object segmentation flowchart.

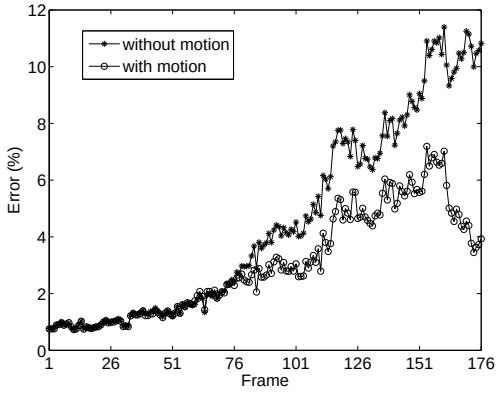

Fig. 5 Frame-by-frame error rate for *Horse Riding* video sequence without and with motion vectors.

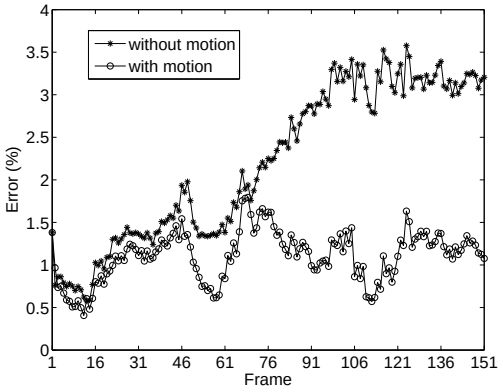

Fig. 6 Frame-by-frame error rate for *Table Tennis* video sequence without and with motion vectors.

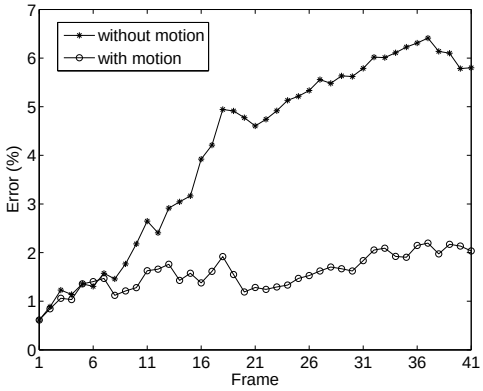

Fig. 7 Frame-by-frame error rate for *Flower Garden* video sequence without and with motion vectors.

Horse Riding video sequence at frame 2, 50, 100, 150 and 176

Segmentation results without motion vectors

Segmentation results with motion vectors

Flower Garden video sequence at frame 2, 15, 27, 35 and 40

Segmentation results without motion vectors

Segmentation results with motion vectors

Table Tennis video sequence at frame 2, 40, 80, 120 and 150

Segmentation results without motion vectors

Segmentation results with motion vectors

Fig. 8 Video Object segmentation results without and with motion vector for video sequence *Horse Riding*, *Flower Garden* and *Table Tennis*.