MVA2009 IAPR Conference on Machine Vision Applications, May 20-22, 2009, Yokohama, JAPAN

Voting based Video Classification Using Clustering and Learning

Kei KIKUCHI Seiji HOTTA
Tokyo University of Agriculture and Technology
2-24-16 Naka-cho, Koganei, Tokyo, 184-8588 Japan
kei.kikuchi522@gmail.com

Abstract

In this paper, we propose video classification using linear manifolds (affine subspaces). In our method, we represent videos belonging to a same class by several linear manifolds using k-varieties clustering. When a test video is given, each frame of it votes for the class to which its nearest linear manifold belongs. According to this voting, the test video is classified into the class that achieves the majority votes. For improving accuracy, a way of adopting generalized learning vector quantization for our video classification is also presented. The performance of our video classification is verified with experiments on short videos downloaded from Web.

1 Introduction

According to the rapid use of videos on Web, needs for content-based video applications such as retrieval [1, 2], copy-retrieval [3], and harmful contents detection [4] have increased in recent years. For realizing these needs, we have to develop techniques for processing a large amount of videos with low computational cost. In this paper, we focus on nonverbal video classification problems. In other words, a test video is classified into its corresponding class without text tags. Of course, it is difficult to represent a video by a single label. Hence, we focus on short videos constructed with a single topic such as videos in YouTube [5]. In this paper, unlabeled and labeled videos are called test and training videos, respectively.

In this paper, we propose video classification using linear manifolds (affine subspaces). By using linear manifold representation, video sequences are compressed effectively and distances between videos can be measured with low computational cost. In our method, videos belonging to a same class are represented by several linear manifolds using k-varieties clustering [6]. We call these clusters *subclasses* in a class. When a test video is given, each frame of it (test frames) votes for the class to which its nearest subclass belongs. According to this voting, the test video is classified into the class that achieves the majority votes. By using our classification, test videos can be classified without specific preprocessing such as shot segmentation. However, subclasses obtained with k-varieties clustering do not guarantee high accuracy. Hence, we apply a learning rule based on generalized learning vector quantization (GLVQ) [7] to subclasses for improving accuracy. The performance of our video classification is verified with experiments on short videos downloaded from Web.

2 Related work

Several video classification methods such as [8, 9] have been proposed in the past. However, the most related work to our video classification is [8], so in this section, let us start with a brief review of the video classification method described in [8].

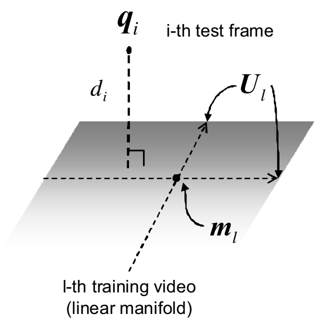

Figure 1: Concept of projection distance.

Let C and N be the numbers of classes and training videos, respectively. Assume that the lth training video ($l = 1, ..., N$) consisting of n_l frames has a single class label y_l. Let $\boldsymbol{x}_s^{(l)} = (x_{s1}^{(l)} \cdots x_{sd}^{(l)})^\top$ ($s = 1, ..., n_l; l = 1, ..., N$) be the d-dimensional feature vector of the sth frame of the lth training video. Consequently, the lth training video is represented by a set of feature vectors (i.e., a matrix) $\mathbf{X}_l = (\boldsymbol{x}_1^{(l)} \cdots \boldsymbol{x}_{n_l}^{(l)}) \in \mathbb{R}^{d \times n_l}$.

In [8], each training video is represented by a single linear manifold obtained with principal component analysis (PCA). That is, the lth training video is represented by a linear manifold spanned by its origin $\boldsymbol{m}_l$ and bases $\mathbf{U}_l$, where $\boldsymbol{m}_l$ and $\mathbf{U}_l$ are the mean vector of $\boldsymbol{x}_s^{(l)}$ (i.e., $\boldsymbol{m}_l = \sum_{s=1}^{n_l} \boldsymbol{x}_s^{(l)}/n_l$) and the eigenvectors corresponding to the r-largest eigenvalues obtained by eigenvalue decomposition of the covariance matrix $\mathbf{C}_l = \sum_{s=1}^{n_l}(\boldsymbol{x}_s^{(l)} - \boldsymbol{m}_l)(\boldsymbol{x}_s^{(l)} - \boldsymbol{m}_l)^\top/n_l$, respectively.

In a classification phase, when a test video consisting of n frames with d-dimensional feature vectors $\mathbf{Q} = (\boldsymbol{q}_1 \cdots \boldsymbol{q}_n) \in \mathbb{R}^{d \times n}$ is given, the test video is classified into the class to which the nearest linear manifold belongs. In practice, the projection distance between the ith frame of the test video (denoted as $\boldsymbol{q}_i$) and the lth training video (cf. Fig. 1) can be measured by

$$d_i = \|\boldsymbol{q}_i - \boldsymbol{m}_l\|^2 - \|\mathbf{U}_l^\top(\boldsymbol{q}_i - \boldsymbol{m}_l)\|^2, \qquad (1)$$

so the total sum of projection distances is given by $D_l = \sum_{i=1}^{n} d_i$. In [8], this total sum of distances D_l is regarded as the distance between the test video and the lth training one, so the class of the test video (denoted as ω) is estimated by the nearest neighbor rule:

$$\min_{l=1,...,N}\{D_l\} = D_{l^*} \Rightarrow \omega = y_{l^*}. \qquad (2)$$

By using this method, training videos can be stored by small memory requirement, and test videos are classified

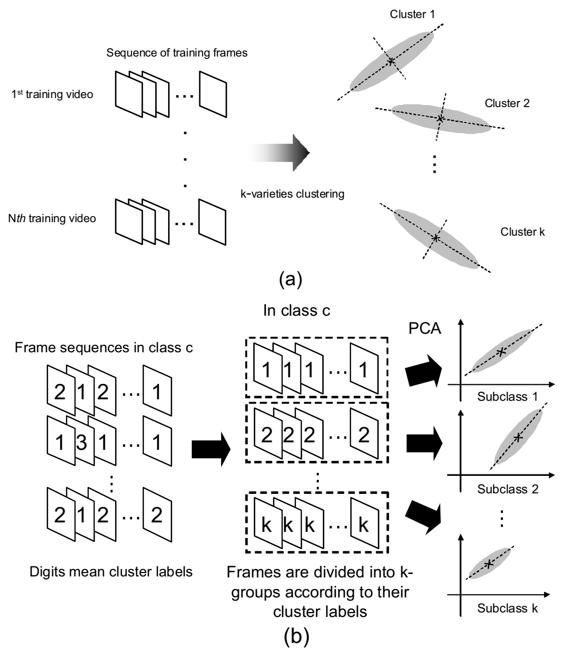
Sequence of training frames
1st training video
Nth training video
k-varieties clustering
Cluster 1
Cluster 2
Cluster k
(a)
In class c
PCA
Frame sequences in class c
Subclass 1
Subclass 2
Subclass k
Digits mean cluster labels
Frames are divided into k-groups according to their cluster labels
(b)

Figure 2: Clustering concept for our video classification.

without specific preprocessing such as shot segmentation. However, this method requires high classification cost when the number of training videos is large. For overcoming this difficulty, more simple representation such as a single linear manifold per class can be applied, but such simple representation yields accuracy deterioration. Thus, we propose here video classification using clustering and learning for improving classification accuracy and reducing memory requirement as small as possible.

3 Our video classification

3.1 Clustering

In our video classification, training videos in a same class are represented with k linear manifolds by using the hard k-varieties clustering (kVC) which is the crisp version of fuzzy c-varieties clustering [6]. By using this clustering, we can use clusters obtained by kVC for classification directly. First, we adopt kVC to all training frames with no class distinction, i.e., the first step for our video classification is described as follows:

Initialization: Choose the number of clusters k. Let $y_s^{(l)}$ be the cluster label of the training frame $\boldsymbol{x}_s^{(l)}$. Assign cluster labels $(1, ..., k)$ to $y_s^{(l)}$ randomly.

Step1: Form the linear manifold of cluster j $(j = 1, ..., k)$ by applying PCA to the frames of which cluster labels $y_s^{(l)}$ are equal to j.

Step2: Classify each frame to its nearest cluster by using the projection distance. That is, measure the projection distance between $\boldsymbol{x}_s^{(l)}$ and cluster j by

$$d_j = \|\boldsymbol{x}_s^{(l)} - \boldsymbol{m}_j\|^2 - \|\mathbf{U}_j^\top(\boldsymbol{x}_s^{(l)} - \boldsymbol{m}_j)\|^2, \qquad (3)$$

where $\boldsymbol{m}_j$ and $\mathbf{U}_j$ are the origin and the orthonormal base of the linear manifold for cluster j. When the nearest cluster is j^* (i.e., $j^* = \arg\min_j d_j$), $y_s^{(l)} = j^*$.

Step3: Iterate **Step1** and **Step2** until all cluster labels $y_s^{(l)}$ are unchanged.

By the above algorithm, each training frame is assigned to its nearest cluster (cf. Fig. 2 (a)). However, these clusters are not used for classification directly, so class-specific k clusters are reconstructed according to cluster labels. In practice, in class c $(c = 1, ..., C)$, the linear manifold of the jth cluster for class c is formed by applying PCA to training frames belonging to class c of which cluster labels are j (cf. Fig. 2 (b)). We call such cluster the jth subclass in class c.

3.2 Classification rule

In [8], the total sum of projection distances is used for classification. However, such distance values sometimes will be large because the number of dimensions of linear manifolds is limited. Hence, we adopt a voting-based rule to our video classification. When a test video $\mathbf{Q} = (\boldsymbol{q}_1 \cdots \boldsymbol{q}_n)$ is given, the projection distance between the ith test frame $\boldsymbol{q}_i$ and all subclasses is measured by

$$d_j^{(c)} = \|\boldsymbol{q}_i - \boldsymbol{m}_j^{(c)}\|^2 - \|\mathbf{U}_j^{(c)\top}(\boldsymbol{q}_i - \boldsymbol{m}_j^{(c)})\|^2, \qquad (4)$$

where $\boldsymbol{m}_j^{(c)}$ and $\mathbf{U}_j^{(c)}$ are the origin and the orthonormal base of the jth $(j = 1, ..., k)$ subclass of class c $(c = 1, ..., C)$. When the nearest subclass of the ith test frame belongs to class c^* (i.e., $c^* = \arg\min_c d_j^{(c)}$), a vote is added to class c^*. This voting is done through all test frames. Let V_c be the total number of votes for class c, the class of the test video is estimated by the majority voting:

$$\omega = \arg\max_{c=1,\ldots,C} V_c. \qquad (5)$$

4 Learning rule for video classification

The subclasses obtained with clustering described above do not guarantee high accuracy, so we apply the learning algorithm called *generalized learning vector quantization* (GLVQ) [7] to our classification. In our classification, a test video is classified by voting, and this voting is done based on the projection distance between a test frame and its nearest linear manifold. Hence, we apply GLVQ to subclasses frame-by-frame.

Let $\boldsymbol{x}$ be a labeled input frame for training. Let $\boldsymbol{m}_1$ and $\mathbf{U}_1$ be the mean vector and the orthonormal basis of the nearest subclass $\mathcal{M}_1$ that belongs to the same class of $\boldsymbol{x}$. In contrast, let $\boldsymbol{m}_2$ and $\mathbf{U}_2$ be the mean vector and the orthogonal basis of the nearest subclass $\mathcal{M}_2$ that belongs to a different class from $\boldsymbol{x}$. Let us consider the relative distance difference $\mu(\boldsymbol{x})$ defined as follows:

$$\mu(\boldsymbol{x}) = \frac{d_1 - d_2}{d_1 + d_2}, \qquad (6)$$

where d_1 and d_2 are the projection distance values from $\boldsymbol{x}$ to $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. The above $\mu(\boldsymbol{x})$ satisfies $-1 < \mu(\boldsymbol{x}) < 1$. If $\mu(\boldsymbol{x})$ is negative, $\boldsymbol{x}$ is classified correctly; otherwise, $\boldsymbol{x}$ is misclassified. For improving accuracy, $\mu(\boldsymbol{x})$ should decrease for all input frames. Thus, a criterion for learning is formulated as minimization of a cost function defined as follows:

$$S = \sum_{j=1}^{n_x} f(\mu(\boldsymbol{x}_j)), \qquad (7)$$

where $\boldsymbol{x}_j$ and n_x are the jth labeled frame and the number of labeled frames for training, respectively. The function $f(\mu)$ is a nonlinear monotonically increasing function. In

[7], a sigmoid function $f(\mu, t) = 1/(1 - e^{-\mu t})$ is used for a such function, where t is learning time. When a sigmoid function is used, $\partial f/\partial \mu$ is derived as $f(\mu, t)\{1 - f(\mu, t)\}$. To minimize S, $\boldsymbol{m}_i$ and $\mathbf{U}_i (i = 1, 2)$ are updated based on a steepest descent method by the following equations:

$$\boldsymbol{m}_i \leftarrow \boldsymbol{m}_i - (-1)^i \alpha \frac{\partial f}{\partial \mu} \frac{d_{3-i}}{d_1 + d_2} (\boldsymbol{y} - \mathbf{U}_i \mathbf{U}_i^\top \boldsymbol{y}), \tag{8}$$

$$\mathbf{U}_i \leftarrow \mathbf{U}_i - (-1)^i \alpha \frac{\partial f}{\partial \mu} \frac{d_{3-i}}{d_1 + d_2} (\boldsymbol{y}\boldsymbol{y}^\top \mathbf{U}_i), \tag{9}$$

where α is small positive constant $(0 < \alpha < 1)$, and $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{m}_i$. In practice, the updated $\mathbf{U}_i$ is not an orthogonal matrix, so the Gram-Schmidt process is applied to $\mathbf{U}_i$ for orthogonalization in each training step. After above learning, test videos are classified based on voting using trained subclasses. As shown in the equations (8) and (9), manifolds are updated by attractive and repulsive forces from $\boldsymbol{x}$. As mentioned in [7], the convergence condition of GLVQ depends on the relation of these forces, i.e., GLVQ converges when attractive forces are larger than repulsive ones. This relation of forces is dependent on the definition of μ, and we can confirm Eq. (6) satisfies this condition as well as the original GLVQ algorithm. Hence, it can be expected that the above manifold learning converges to a good solution.

5 Experimental results

We tested the proposed method on the videos downloaded from MoCoVideo [10]. The video uploaded on MoCoVideo has a single text tag, so we downloaded videos from the site and constructed a dataset with them. Consequently, we have gotten the dataset that consists of 750 videos formed by 15 classes, i.e., soccer, talk show, car, comedy show, air show, TV games, basketball, table tennis, swimming, surfing, baseball, news, night scene, train, and bike race. The average length of a video is 260 sec. In experiments, we reduced the frame resolution of all videos to 6×8, 9×12 and 12×16 pixels with a uniform quantization method. Furthermore, we reduced the number of frames of all videos to 100 frames. In this experiment, we used color pixel values of each frame as feature vectors. So, the number of dimensions of each feature vector were $6 \times 8 \times 3 = 144$, $9 \times 12 \times 3 = 324$, and $12 \times 16 \times 3 = 576$, respectively. All algorithms were implemented with MATLAB on a standard PC that has Pentium 2 GHz CPU and 2 GB RAM.

5.1 Error rates with respect to k

First, we investigated error rates with respect to the number of subclasses k. Error rates were estimated by 10-fold cross-validation with varying from $k = 1$ to $k = 7$. In all subclasses, the number of dimensions of linear manifolds were fixed with $r = 10$. For learning, $\alpha = 10^{-5}$ was used for a positive constant.

Figure 3 illustrates error rates with respect to the number of subclasses. In this figure, horizontal and vertical axes denote the number of subclasses and error rates, respectively. The dotted line indicates the error rate obtained by our method without learning. In contrast, the solid line indicates the error rate obtained by our method with learning. As shown in this figure, the error rates decreased with the number of subclasses increased. This result shows that subclass representation of videos is effective for improving accuracy. In addition, by learning, the error rates was improved about 10% in every k.

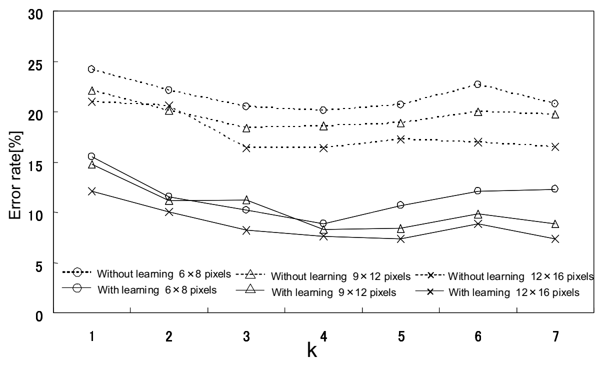

Figure 3: Error rates with respect to # subclasses.

5.2 Error rates of video classification

Next, we compared our method with *other video classification methods*: Hausdorff distance measure [11], support vector machine (SVM) [12], projection distance-based classification [8], and genre classification using shot segmentation [9].

For matching two videos by a Hausdorff distance, we represent videos with representative frames obtained by k-means clustering [11]. In our experiments, error rates of Hausdorff distance classification were evaluated by varying the number of clusters.

When SVM is used for video classification, we have to represent a video as a single feature vector. Here, we represent a video as a histogram. First, all frames were clustered into k clusters by using k-means clustering. Next, we computed a histogram by counting frequencies of each cluster in a video. By this processing, a video was represented as a k-dimensional histogram $\boldsymbol{h} = (h_1 \cdots h_k)^\top$. Such histogram was used as feature vectors for video classification. For SVM, we used the Gaussian kernel $K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\alpha\|\boldsymbol{x}-\boldsymbol{y}\|^2}$ for nonlinear mapping.

In genre classification using shot segmentation, the video was represented as 11-dimensional feature vector [9]. First, each video was segmented into shots using the method described in [13]. Next, 11 feature vectors were extracted from each shot of videos. After that, a C4.5 decision tree was applied to build a classifier.

Table 1 shows the minimum error rate of each method and its standard deviation in each frame resolution. As shown in this table, our method with learning outperformed the other classifiers drastically. In conventional classifiers using clustering, videos are represented by centroids or representative frames. Such compression tends to lose discriminative information, so it is difficult to achieve high accuracy. In contrast, manifold-based classifiers can represent various frames by linear combinations of orthonormal bases. In other words, a representation capacity of linear manifolds is larger than vectorial features. Note that, however, the error rate of our method with learning was lower than the projection-distance-based methods. The difference between them is a way of classification, i.e., distance-based or voting-based. In distance-based methods, projection distances between some frames and their correct classes sometimes gets larger unexpectedly because the number of dimensions of linear manifolds is limited. Consequently, some test videos are misclassified by summing projection distances. In contrast, voting-based classification is robust against unexpected misclassification of some test frames, and such misclassification can be reduced easily by frame-by-frame learning.

Table 1: Error rates of each method [%].

classifier	6×8	9×12	12×16
Our method without learning ($k = 4$)	20.1 ± 4.2	18.4 ± 4.5	16.4 ± 5.1
Our method with learning ($k = 4$)	8.9 ± 3.3	8.4 ± 4.0	7.3 ± 3.9
Hausdorff ($k = 11$)	34.3 ± 5.4	31.2 ± 5.1	29.3 ± 4.7
SVM ($\alpha = 0.003, k = 18$)	21.2 ± 3.9	18.2 ± 3.2	16.5 ± 4.1
projection distance [8]	25.6 ± 5.1	22.6 ± 4.7	20.0 ± 5.6
projection distance with learning [8]	19.6 ± 3.9	16.6 ± 3.2	15.8 ± 3.7
genre classification [9]	22.7 ± 3.2	20.9 ± 3.4	19.9 ± 2.8

Table 2: Classification time [sec].

classifier	6×8	9×12	12×16
Our method ($k = 4$)	**0.1**	0.4	1.1
projection distance [8]	1.4	2.2	3.9

5.3 Classification and learning time

Next, we compared classification cost of our method with that of the projection distance-based method [8]. First, we measured the mean classification time per video. The number of dimensions of linear manifolds were fixed with $r = 10$. Table 2 shows the mean classification time of each method. As shown in this table, when the frame resolution was 6×8 pixels, our method was about 10 times faster than the projection distance-based method. The reason for this is the difference of the number of linear manifolds. That is, the number of linear manifolds in the projection distance-based method is equal to the number of all training video N. In contrast, our method requires only k linear manifolds in each class ($k \times C$). Hence, our method can classify videos by lower classification cost than the projection distance-based method.

In addition, we measured mean time for learning linear manifolds. The number of iteration for learning was fixed with 100 times. Table 3 shows the learning time of each method. As shown in this table, when the frame resolution was 6×8 pixels, our method required about 17000 seconds, but the projection distance-based method required over 50000 seconds. As shown in Table 1 and Table 3, error rates decreased slightly according with the frame resolutions increased, but the learning costs were also increased drastically. Hence, we can conclude that 6×8 pixels were enough for our video classification.

5.4 Effectiveness of learning

Finally, we investigated the effectiveness of learning described in Section **4** for voting-based classification. As an example, in Fig. 4, we show a voted classes of each frame of a video belonging to class 4. In this figure, the frame resolution was 6×8 pixels. Figure 4 (a) shows the result obtained by our method without learning. In contrast, Fig. 4 (b) shows the result obtained by our method with learning. In these figures, horizontal and vertical axes denote frame numbers and their voted classes, respectively. As shown in Fig. 4 (a), many frames were misclassified into different classes. In contrast, as shown in Fig. 4 (b), almost of all frames were classified into the correct class (class 4). Hence, it was verified that frame-by-frame learning described in Section **4** is effective for improving classification accuracy of our voting-style classification.

Table 3: Learning time of each method with learning [sec].

classifier	6×8	9×12	12×16
Our method ($k = 4$)	**16582**	33563	98772
projection distance [8]	57343	77865	135629

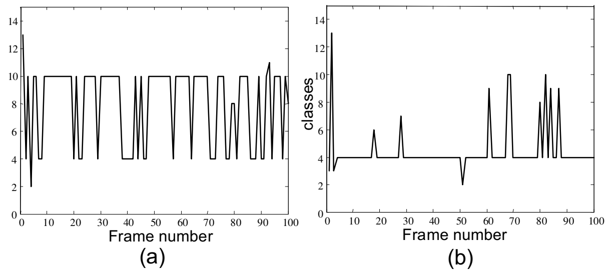

Figure 4: Effectiveness of learning.

6 Conclusion

This paper presented video classification using linear manifolds and learning. In our method, a test video is classified into the class that achieves the majority votes from test frames. For improving accuracy, we applied a learning algorithm to our method. The performance of our video classification was verified with experiments on short videos downloaded from Web. Experimental result showed that our method outperformed conventional video classification methods. Future work will be dedicated to apply other compression methods such as probabilistic PCA [14] to our video representation.

References

[1] Y. A. Aslandogan and C. T. Yu, "Techniques and systems for image and video retrieval," IEEE Trans. on knowledge and data engineering, vol. 11, no. 1, pp. 56–63, 1999.

[2] Z. Xiong, *et al.*, "Semantic retrieval of video - review of research on video retrieval in meetings, movies and broadcast news, and sports," IEEE Signal Processing Magazine, vol. 23, no. 2, pp. 18–27, 2006.

[3] A. Joly, O. Buisson, and C. Frelicot, "Content-based copy retrieval using distortion-based probabilistic similarity search," IEEE Trans. on Multimedia, vol. 9, no. 2, pp. 293–306, 2007.

[4] D. Koizumi, *et al.*, "A method of filtering hazardous images on WWW image search systems," J. IPS. Japan, vol. 47, No. SIG8, pp. 147–156, 2006.

[5] `http://www.youtube.com/`

[6] J. C. Bezdek "Pattern Recognition with Fuzzy Objective Function Algorithms," Plenum Press, 1981.

[7] A. Sato and K. Yamada, "Generalized learning vector quantization," NIPS, pp. 423–429, 1995.

[8] K. Kikuchi and S. Hotta, "Video Classification Using Linear Subspace Methods," Proc. of KJPR2007, pp.15-20, 2007.

[9] B-T. Truong, S. Venkatesh, and C. Dorai, "Automatic genre identification for content-based video categorization," ICPR, vol. 4, pp. 230–233, 2000.

[10] `http://www.mocovideo.jp/`

[11] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance," Trans. on PAMI, vol. 15, pp. 850-863, 1993.

[12] C.C. Chang and C.J. Lin,"LIBSVM: A library for support vector machines," 2001.

[13] B. T. Truong, *et al.*, "New enhancements to cut, fade, and dissolve detection processes in video segmentation," Proc. of ACM Multimedia, pp. 219-227, Nov. 2000.

[14] M. Tipping and C. Bishop, "Probabilistic principal component analysis," Journal of the Royal Statistical Society, pp. 611-622, 1999.