LAMP-TR-132 May 2006
CAR-TR-1013
CS-TR-4804
UMIACS-TR-2006-25

SOFTCBIR: OBJECT SEARCHING IN VIDEOS COMBINING KEYPOINT MATCHING AND GRADUATED ASSIGNMENT

Ming Luo, Daniel DeMenthon, Xiaodong Yu, David Doermann

Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
ming@cs.umd.edu, daniel@cfar.umd.edu

Abstract

This paper proposes a new approach to object searching in video databases, *SoftCBIR*, which combines a keypoint matching algorithm and a graduated assignment algorithm based on *softassign*. Compared with previous approaches, SoftCBIR is an innovative combination of two powerful techniques: 1) An energy minimization algorithm is applied to match two groups of keypoints while accounting for both their similarity in descriptor space and the consistency of their geometric configuration. The algorithm computes correspondence and pose transformation between two groups of keypoints iteratively and alternately toward an optimal result. The objective energy function combines normalized distance errors in descriptor space and in the spatial domain. 2) Initial individual keypoint matching relies on Approximate K-Nearest Neighbor (ANN) search. ANN achieves much more accurate initial keypoint matching results in the descriptor space than K-means labeling. Experiments prove the effectiveness of our approach, and demonstrate the performance improvements rising from the combination of the two proposed techniques in the SoftCBIR algorithm.

The support of this research by the Department of Defense under contract MDA-9040-2C-0406 is gratefully acknowledged.

1 Introduction

As digital video is becoming more prevalent in people's life, content-based video analysis has grown into a very active research area. Object searching is a relatively new topic [17, 13, 23, 24, 11, 12, 14, 3, 18, 20] in content-based video analysis, and consists of searching desired objects in videos by query words or examples, according to three approaches: 1) textual only, 2) visual only, 3) textual and visual. In this paper, we have focused on the visual approach only.

Almost all researchers agree that an object can be represented by a constellation of keypoints with a specific geometric configuration [17, 13, 23, 24, 3, 18, 20]. The mainstream approach of object searching [17, 13, 23, 24, 20] is composed of three steps: detecting keypoints, computing descriptors, and matching. In the first step, keypoints are detected and located with specific position, scale and orientation. The second step computes descriptors at keypoint locations. The third step matches keypoints by their descriptors. The matching between two keypoints is evaluated by the distance between their descriptors in the descriptor vector space. An object is represented by a local group of keypoints. To match two objects, their groups of keypoints are matched based on certain criteria, including not only the matching between individual keypoints, but also some measures based on comparisons of the geometric configurations between the two groups. For the above three steps, researchers have some agreement about the first step and the third step. For the first step, typically, keypoints are detected at extrema positions in a scale-space. For the third step, typically, people enforce either *neighborhood* consistency or geometric consistency between the two objects. (Authors have used the term *spatial consistency* to express the concept of neighborhood consistency, but we avoid it as this seems confusing when examined at the same time as geometric consistency.) Neighborhood consistency enforces the fact that a group of neighbor keypoints in one object should be mapped into a group of neighbor keypoints in the other object [23]. Geometric consistency assumes there exists a planar affine transform or homography between the points of the two objects. Geometric consistency is stronger than neighborhood consistency, but more complicated to compute. On the other hand, for step 2, the reported methods are very diverse. In 2004, Mikolajczyk and Schmid [19] presented a comparative study of the available local descriptors: Harris-Affine detector [8], shape context [2], steerable filters [4], differential invariants [15], spin images [16], complex filters [22], moment invariant [7], SIFT [17], PCA-SIFT [13], and cross-correlation of different types of keypoints. That work concludes that SIFT performs best in the comparison experiments. However, the performance of PCA-SIFT is not far behind. In addition, its acronym is somewhat a misnomer in the sense that it does not use the histograms of gradient directions in subregions around interest points that define the SIFT approach. Instead, the feature vector is obtained by PCA analysis of the raw vectors of gradient values around interest points. Furthermore, the PCA step results in much smaller feature vectors (typically 20 dimensions instead of 128 dimensions).

Sivic and Zisserman proposed several video mining approaches, including Video Google [23] in 2003 and a related approach [24] in 2004. They use Lowe's keypoint detector and SIFT descriptor [17], and had some success in extracting objects in videos. But there are three shortcomings in their work. 1) In [23] and [24], for matching between groups of keypoints, the authors argue that computing the parameters of the planar homography between the two groups using iterations is time-consuming, and they adopt a simpler testing criterion, computing the number of matched individual keypoint pairs between the two groups after neighborhood consistency filtering. If

this number is above a certain threshold, the two groups are thought to be matched. Geometric consistency between groups is not considered. This simplification will produce false object matches under certain circumstances. 2) In [23] and [24], the authors use K-means clustering to perform a vector quantization on the set of keypoint descriptors extracted from the keyframes of videos in the database. The number of clusters K is set manually. K-means clusters are given unique labels, and each keypoint is given the label of the cluster it belongs to. In the matching, keypoints with identical labels are matched. There are two problems in this method. First, the K-means algorithm is not very suitable for vector quantization in large and high-dimensional data sets, due to its two inherent drawbacks: dependency on the initial state and degeneracy [25]. The initial state includes the selection of initial centers and the value of K. K-means is non-deterministic, thus final results depend upon the initial state, because the mean squared error often converges to a local minimum [10]. But the local minimum cannot guarantee an ideal Voronoi graph-like vector quantization result. Instead, the K clusters in the K-means result may be elongated in the high-dimensional space. Additionally, the value of K is also critical to the clustering result, and obtaining an optimal K for a given data set is an NP-hard problem [25]. When the distribution of the data set is unknown, the optimal K is hard to attain. A manually assigned value of K may not be suitable for the data, falsely splitting one good cluster into two, or missing some obvious splits, or both. The risk of degeneracy implies that the clustering may end with some meaningless sparse clusters. Second, K-means is not very suitable for object searching. There may be a lot of keypoints lying around the borders between clusters, because K-means does not account for point density. Thus, it does not hold that two keypoints assigned the same labels are among the closest neighbors in the descriptor vector space. This fact will cause errors in a matching algorithm based on K-means results. 3) They use "raw" SIFT descriptors, which are 128-dimensional vectors. Considering that there are hundreds or thousands of keypoints in a single ordinary video frame, the distance computation between SIFT descriptors in the 128-dimensional vector space for thousands of frames is overwhelming. Such a high-dimensional descriptor is impractical for a usable video analysis system, which requires near real-time response to users' interactions.

In this paper, we describe a new approach to object searching in videos called *SoftCBIR* that addresses the above three problems. It combines two innovative components. (1) An energy minimization framework is proposed to update the pose transformation and correspondence alternately and iteratively. We accomplish this by extending the ideas of softassign [6]. Softassign is a probabilistic framework that lets all the available data participate in the updates of the pose transformation and correspondence. So it is less vulnerable to the selection of the initial state than the incremental method of [20]. The energy minimization combines the use of keypoint matching and softassign to evaluate the matching between two groups of keypoints, by evaluating the similarity in keypoint descriptor vector space and the geometric consistency between two groups of keypoints simultaneously. The iteration gives an optimal result for both correspondence and pose transformation. (2) We use Approximate K-Nearest Neighbor (ANN) [1] to perform the initial individual keypoint matching in the descriptor vector space instead of the K-means used by [23] and [24]. ANN searching is much more accurate than searching by K-means labeling. Although naïve nearest neighbor would be very time-consuming, ANN uses a balanced box-decomposition (BBD) tree, which is efficient to build and search. It also provides a parameter to adapt the tradeoff between accuracy and efficiency. Evaluation shows that it can achieve both fairly high

accuracy and efficiency [1]. The SoftCBIR algorithm is the novel combination of these two components. In addition, since it uses PCA-SIFT, it handles feature vectors with more manageable numbers of dimensions.

The rest of the paper is organized as follows: Section 2 gives an overview of our system. Sections 3 and 4 present details about the energy minimization algorithm and ANN. Section 5 shows results of experiments and evaluations. Conclusions and future work are addressed in Section 6.

2 Overview

In a preprocessing stage, one keyframe is extracted for each shot in the video database, and keypoints (including their x-position, y-position, scale and orientation) and their descriptors are extracted using the PCA-SIFT approach for each keyframe. Then a BBD tree is built using the whole set of descriptors for all the videos. Below is an outline of the process for our object searching approach once the user opens a query image and draws a rectangle to define a query object.

1. Extract the keypoints and their descriptors in the user's query rectangle.
2. For each keypoint in the query rectangle, search its K nearest neighbor keypoints as matched keypoints with ANN using the BBD tree built in the preprocessing stage. In our experiments, K is set to 14 in ANN.
3. Perform the energy minimization algorithm *for each keyframe containing a subset of the matched keypoints obtained by ANN.*
4. For each keyframe, define a *matching score* equal to the number of final matched keypoints after the energy minimization algorithm, and determine a rectangle framing all matched keypoints as the matched object area.
5. Rank the keyframes in the database according to their matching scores.
6. Submit ranked results: the user can choose to view top-ranked keyframes.

3 Energy minimization

Graduated assignment is an iterative registration algorithm that in particular is used to register two sets of 2D points in medical images [6]. For two groups of points in 2D or 3D space, it computes the correspondence and pose transformation iteratively and simultaneously, using a fast deterministic annealing mechanism. The objective is to minimize an energy function, which is a sum of distance errors between pairs of matched keypoints in the spatial domain. In our object searching framework, after achieving an initial individual matching of keypoints, it is necessary to check the geometric consistency between the two groups of keypoints because of the following two reasons: 1) Some individual matchings of keypoints are wrong. 2) Correct individual matching of keypoints cannot guarantee that the two groups of keypoints are matched when they exhibit similar geometric configurations. This geometric consistency problem is more complex than the purely spatial matching problem that the original softassign algorithm was meant to solve. Indeed

we should consider not only the distance error in the spatial domain, but also the distance error in the descriptor vector space of keypoints. In the energy minimization of our SoftCBIR algorithm, we combine these two distances into a new energy function and apply deterministic annealing iterations similar to those of softassign to find video frames containing groups of keypoints that match the keypoints of query rectangles with similar individual descriptors and similar geometric configurations.

Consider a query image and a test video frame. The query image has a list of J keypoints $p_j^{(1)} = (x_j^{(1)}, y_j^{(1)})$ with descriptors $d_j^{(1)}$ in the descriptor space. The test video frame has a list of k keypoints $p_k^{(2)} = (x_k^{(2)}, y_k^{(2)})$ with descriptors $d_k^{(2)}$ in the descriptor space.

We use a correspondence matrix M to indicate the correspondence between the J keypoints in the query image and the K keypoints in the test video frame. M_{jk} indicates the probability of correspondence between $p_j^{(1)}$ and $p_k^{(2)}$. The matrix M is a $(J+1) \times (K+1)$ matrix because an additional slack row and an additional slack column are needed to indicate when no acceptable correspondence has been found for a keypoint in the query image or in the test frame respectively. At each step of iteration, the Sinkhorn algorithm, an iterative normalization of rows and columns, is applied to M to insure that each row and each column sum up to one.

We look for an affine transform A from the group of keypoints in the query image to the group of keypoints in the test frame, defined as

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Thus the point $A\, p_j^{(1)}$ is the corresponding spatial position for $p_j^{(1)}$ after the affine transform.

The objective energy function E is then defined as

$$E = \sum_{j=1}^{J} \sum_{k=1}^{K} M_{jk} \left(\frac{D_s^2(A\, p_j^{(1)}, p_k^{(2)})}{\alpha_s} + \frac{D_d^2(d_j^{(1)}, d_k^{(2)})}{\alpha_d} \right) \tag{2}$$

where $D_s(\cdot, \cdot)$ is the Euclidean distance function in the spatial domain and $D_d(\cdot, \cdot)$ is the Euclidean distance function in the descriptor vector space, and α_s and α_d are normalization parameters for the two distances.

The energy minimization used by SoftCBIR is a deterministic annealing procedure in which a temperature $T = 1/\beta$ is initially high so that small local minima are first smoothed out at the expense of accuracy and is progressively decreased so that, once a large minimum is found, landscape details are provided for increased accuracy. At each temperature level, the first step is to update M using the current squared spatial distance errors $D_s^2(A\ p_j^{(1)}, p_k^{(2)})$ and squared descriptor distances $D_d^2(d_j^{(1)}, d_k^{(2)})$. The second step is to update the affine transformation A by solving in closed form for the affine transform parameters that set the derivative $\partial E / \partial A = 0$, thereby minimizing the energy. The code of this component of our SoftCBIR algorithm can be summarized as follows:

Inputs:

1. For the query image, a list of J keypoints $p_j^{(1)} = (x_j^{(1)}, y_j^{(1)})$ with descriptors $d_j^{(1)}$ in the descriptor space.

2. For the test video frame, a list of K keypoints $p_j^{(2)} = (x_j^{(2)}, y_j^{(2)})$ with descriptors $d_k^{(2)}$ in the descriptor space.

Initialize slack elements of assignment matrix M to $\gamma = 1/(max\{J, K\} + 1)$, β to β_0.

Initialize A to an expected pose transformation (see below for details).

Do A until $\beta > \beta_{final}$ or $E < E\_threshold$ (deterministic annealing loop)

- Compute combined squared distances $D_{jk}^2 = D_s^2(Ap_j^{(1)}, p_k^{(2)})/\alpha_s + D_d^2(d_j^{(1)}, d_k^{(2)})/\alpha_d$
- Update $M_{jk} = \gamma e^{-\beta D_{jk}^2}$
- **Do B until** ΔM **small** (Sinkhorn algorithm)

  Update Matrix M by normalizing each row except slack row: $M_{jk} = M_{jk} / \sum_{k=1}^{K+1} M_{jk}$

  Update Matrix M by normalizing each column except slack column: $M_{jk} = M_{jk} / \sum_{j=1}^{J+1} M_{jk}$
- **End Do B**
- Compute energy $E = \sum_{j=1}^{J} \sum_{k=1}^{K} M_{jk}(D_{jk}^2)$
- Update A by minimizing the energy E, i.e., solve the equation $\partial E/\partial A = 0$ with the parameters of A as variables (see below for details)
- $\beta = \beta_{update}\beta$

End Do A

Outputs: Affine matrix A and correspondence matrix M between the groups of keypoints in query image and test video frame.

In other words, the energy minimization in SoftCBIR produces a high probability of correspondence between pairs of keypoints that are close in descriptor space and also in the spatial domain after affine transformation, thanks to the Gaussian relationship between distance and correspondence of the update step $M_{jk} = \gamma e^{-\beta D_{jk}^2}$. Note the multiplying term β, which is the inverse of the temperature. In the first iterations, β is small, so that the algorithm does not pay much attention to distances, and probabilities tend to be equally distributed between several possible pairings between keypoints. It is only after several iterations, when the affine transformation becomes more correctly calculated, that the parameter β becomes large enough and the algorithm starts to fully account for the distances between points, which in turn improves the affine transform calculation. In our experiments, we use $\beta_0 = 1.2$, $\beta_{update} = 1.05$, and $\beta_{final} = 10$. Loop B within loop A enforces normalization to one for the probabilities of each row and each column. The combined result of normalization and exponentiation of distances is a winner-take-all mechanism in which one term in each row and each column tends to grow closer to one at the expense of the other terms. If the combined distances between a transformed keypoint and all the keypoints of its row remains large, then the winner becomes the extra term of the slack column, therefore a large value in the slack column indicates that no match was found for a keypoint of the query image. Similarly, elements of the slack row are large when keypoints of a video frame do not correspond to any keypoints of the query image.

At each step, the parameters of the affine transformation A that minimize the energy can be found in closed form by solving the two 3×3 systems:

$$\begin{bmatrix} \sum_{j,k} \frac{2M_{j,k}{x_j^{(1)}}^2}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}y_j^{(1)}}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}}{\alpha_s} \\ \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}y_j^{(1)}}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}{y_j^{(1)}}^2}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}y_j^{(1)}}{\alpha_s} \\ \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}y_j^{(1)}}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}}{\alpha_s} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}x_k^{(2)}}{\alpha_s} \\ \sum_{j,k} \frac{2M_{j,k}y_j^{(1)}x_k^{(2)}}{\alpha_s} \\ \sum_{j,k} \frac{2M_{j,k}x_k^{(2)}}{\alpha_s} \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \sum_{j,k} \frac{2M_{j,k}{x_j^{(1)}}^2}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}y_j^{(1)}}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}}{\alpha_s} \\ \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}y_j^{(1)}}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}{y_j^{(1)}}^2}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}y_j^{(1)}}{\alpha_s} \\ \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}y_j^{(1)}}{\alpha_s}, \sum_{j,k} \frac{2M_{j,k}}{\alpha_s} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \sum_{j,k} \frac{2M_{j,k}x_j^{(1)}y_k^{(2)}}{\alpha_s} \\ \sum_{j,k} \frac{2M_{j,k}y_j^{(1)}y_k^{(2)}}{\alpha_s} \\ \sum_{j,k} \frac{2M_{j,k}y_k^{(2)}}{\alpha_s} \end{bmatrix} \tag{4}$$

After the energy minimization stage, we transform the correspondence matrix M, which expresses probabilities of correspondences between keypoints of the query image and the test video frame, into a hard assignment matrix with only zeros and ones: if M_{jk} is the unique maximum in both the j^{th} row and the k^{th} column and M_{jk} is larger than a threshold $\theta = 0.9$, then M_{jk} is set to one, and if this term is not in a slack row or column, the keypoint $p_j^{(1)}$ and $p_k^{(2)}$ are considered to be matched keypoints.

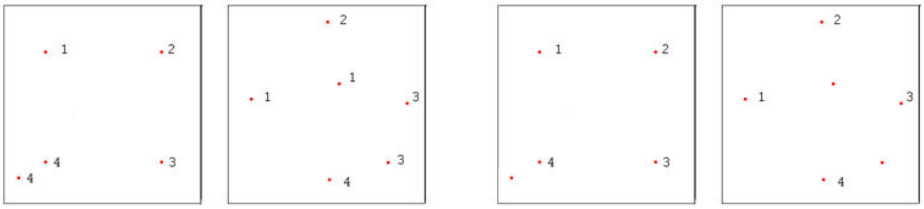
At this stage we adopt the number of surviving matched keypoints as the matching score. The frames are ranked by their matching score and the ranked list is delivered as the object searching result.

The deterministic annealing algorithm is slow if the affine matrix A is initialized with random values. To speed up the convergence of the algorithm, we estimate an initial affine matrix A using the initial matched keypoints provided by ANN. The routine for producing this initial value for A is similar to one run of the loop A in the pseudocode, in which M is computed using only distances in the descriptor space, i.e. as $M_{jk} = \gamma e^{-\beta\, D_d^2(d_j^{(1)}, d_k^{(2)})/\alpha_d}$.

In Section 2, we explained that the K used for ANN is 14. For each keypoint inside the query rectangle of the query image, we find its 14 nearest neighbors in the keypoint descriptor space among all the keypoints in all the keyframes in the database. So in each test frame, for each query keypoint, we may find several matched keypoints. In other words, in the initialization stage of SoftCBIR algorithm, there can be one-to-many or many-to-one matches. But after the energy minimization, the thresholding of the correspondence matrix into a hard assignment matrix guarantees one-one matches. Fig. 1 illustrates this mechanism.

4 ANN

ANN search is used to generate candidate matches as inputs to the energy minimization algorithm. As mentioned, K-means has inherent drawbacks for keypoint matching. Nearest neighbor (NN) search is better suited for this task. Given a set S of n data points in a d-dimension feature space F, the goal of NN searching is to find the data points in S closest to a given query point q in F. But the brute-force search requires $O(dn)$ time, which is unacceptable in many practical problems. A K-d tree can be used to address the NN problem efficiently [5] and find K nearest neighbors



Query image Result frame (a) Query image Result frame (b)

Figure 1: A diagram example of matched keypoints before energy minimization (a) and after it (b). In (a), identical numbers indicate they are nearest neighbors found by ANN. In (b) identical numbers indicate they belong to matched pairs.

in logarithmic time. But in fact, it is logarithmic only for fairly restricted input distributions in low-dimensional space. For certain inputs or in high-dimensional space, the K-d tree search can be linear in time. In light of these difficulties, an alternative approach is to find the approximate nearest neighbors (ANN) [1], which is formulated as: given a set S of n data points and a query point q in a d-dimension feature space F, find a p in S such that p is a $(1+\varepsilon)$-approximate nearest neighbor of q.

$$dist(p, q) \leq (1 + \varepsilon) dist(p*, q). \tag{5}$$

where $p*$ is the true nearest neighbor to q. Arya et al. [1] show that in $O(d\, n \log n)$ time it is possible to construct a data structure of size $O(d\, n)$ such that an ANN p can be reported in $O(c_{d,\varepsilon} \log n)$ time for a constant $c_{d,\varepsilon} \leq d \lceil 1 + 6d/\varepsilon \rceil^d$ and any Minkowski metric. The data structure used in [1] is a balanced box-decomposition (BBD) tree that hierarchally decomposes the n data points into a collection of cells, each of which is either a d-dimensional rectangle or a set-theoretic difference of two rectangles, one enclosed within the other. In our system, we build a BBD tree to store the whole set of keypoints obtained from all the frames of the videos. In ANN, we use $\varepsilon = 0.1$ to increase speed without adversely reducing accuracy. The K in ANN is an adjustible parameter of our system. It is easily adjustable by users to get suitable for a database they have. Generally, the more smilar images in the database, the larger K should be.

5 Experiments and Evaluation

5.1 Data

We use ABC news videos provided by TRECVID 2004 [9] as our training data and testing data. Using the ground truth of shot boundaries provided by TRECVID 2004, we extract one keyframe (the middle frame) for each given shot, and extract their keypoints and PCA-SIFT descriptors. Table 1 describes the training set and testing set. A BBD tree is built using all these PCA-SIFT descriptors. Both the training set and the testing set contain not only news but also commercials.

The evaluation of our framework is conducted at the object level. Ground truth objects were manually labeled and framed with rectangles by five students working independently. These

Table 1: Training set and Testing set

	Training set	Testing set
Files	"19981210_ABCa.mpg" "19981221_ABCa.mpg"	"19981004_ABCa.mpg" "19981021_ABCa.mpg" "19981109_ABCa.mpg" "19981126_ABCa.mpg"
Length	56 minutes and 47 seconds	1 hour 53 minutes and 40 seconds
Number of keyframes	844	1,735
Number of keypoints	391,213	818,997

objects have diverse semantics, including human faces, superimposed texts, logos, buildings, animation figures, banners, computers, maps, ties, etc. Only those objects that occurred in at least two different keyframes were labeled. The resulting total number of labeled objects is 222. The number of occurrences of objects varies from 2 to 32.

5.2 Method

In the evaluation of our object searching approach, we use all the instances of all the objects labeled in the ground truth as queries in turn. Results are delivered as ranked frame lists, along with rectangles indicating the detected matching object area. For each query, the query frame itself is automatically filtered out of the result list. For each result frame, if the detected object area overlaps an instance of the current query object, it is counted as a correct result; otherwise a false positive is reported.

5.3 Results

We present our results with three methods, (1) Mean Precision Recall Curve, (2) Mean R-Precision and (3) Mean Average Precision over all objects [21]. To analyze the contributions of the energy minimization algorithm and ANN to our SoftCBIR approach, we perform the following three experiments: (1) with energy minimization and ANN (our proposed approach), (2) with ANN and without energy minimization algorithm, (3) with energy minimization algorithm and without ANN (using K-means instead). Fig. 2 illustrates the comparison between the three results with the mean precision recall curves. In Fig. 2, the solid curve is for (1), the dashed curve is for (2), and the dotted curve is for (3). It is clear that our approach combining both energy minimization and ANN outperforms the other two by a large margin. Table 2 tabulates the detailed values on the curve of our approach. Table 3 compares the Mean R-Precisions and Mean Average Precision of the three experimental results. For general purpose object searching in a large-scale video database, the results of our approach are encouraging. In this experiment, the K for K-means is chosen as 11700 because this value yields the highest results.

Fig. 3 to Fig. 6 show some examples of valid frames returned by SoftCBIR. The results in Fig. 3, 4 and 6 are perfect, in the sense that all the relevant frames listed in ground truth are delivered at the top of the ranking. Fig. 5 shows two false positives - Result 11 and Result 12. Fig. 7 shows four queried objects that failed to return any correct results. From Fig. 7 we can

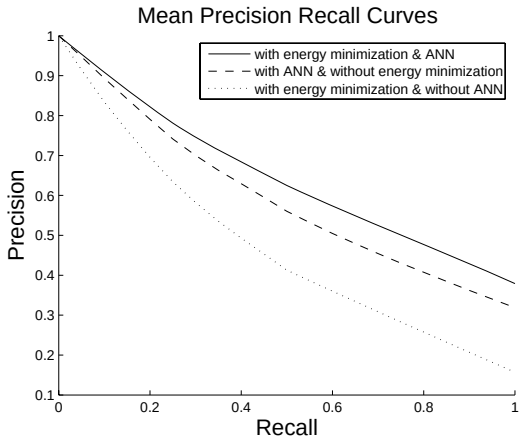


Figure 2: Comparison between approaches in Mean Precision Recall Curve

Table 2: Points on Mean Precision Recall Curve of our approach

Recall	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Precision	1.00	0.95	0.91	0.86	0.82	0.78	0.74	0.71	0.68	0.65	0.62
Recall	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00	
Precision	0.60	0.57	0.55	0.52	0.50	0.48	0.45	0.43	0.40	0.38	

see that our approach does not work well for the following types of objects: 1) Objects with no or few keypoints because of characteristics of the object such as blur, low resolution or shading, for example in (a)-(c). 2) Objects with perspective change beyond 30 degrees, for example, (c). 3) Objects with too high-level semantic meaning, for example, (d).

5.4 Speed

We tested the system speed using all the instances of all the objects in the ground truth as queries in turn. For each such query, we search it in the whole testing set containing 1735 images. On a computer with Pentium 4 CPU 2.4GHz and 1GB RAM, the average running time of one object searching task is 28.71 seconds, the minimum time is less than 3 second, and the maximum time is 170 seconds. With the same hardware conditions, if David Lowe's raw SIFT descriptor are used, the average running time of one object-searching task is 45 hours. This is very slow because of the constant $c_{d,\varepsilon} \leq d\lceil 1 + 6d/\varepsilon \rceil^d$ in Section 4, which increases exponentially with d

Table 3: Comparison in Mean R-Precision and Mean Average Precision

	Mean R-Precision	Mean Average Precision
With energy minimization and ANN	0.478	0.505
With ANN, without energy minimization	0.411	0.452
With energy minimization, without ANN	0.206	0.242

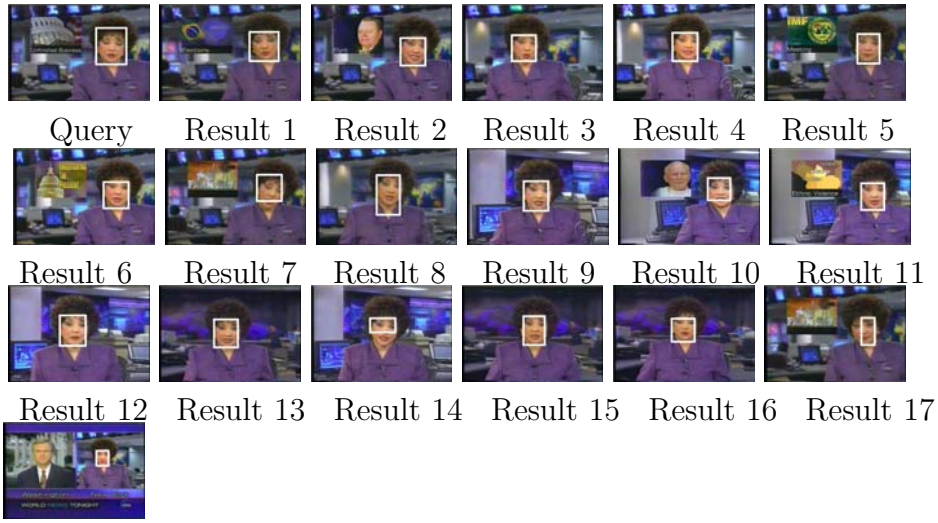


Figure 3: Search results for the object "Carole Simpson"

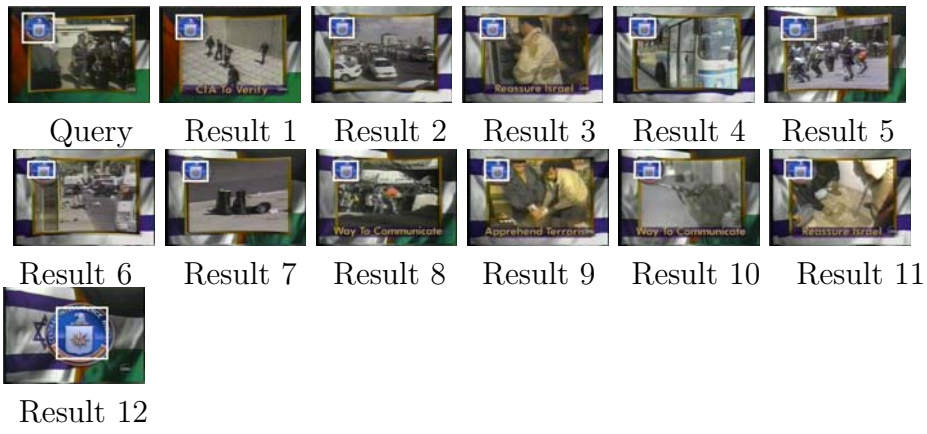


Figure 4: Search results for the object "CIA seal"

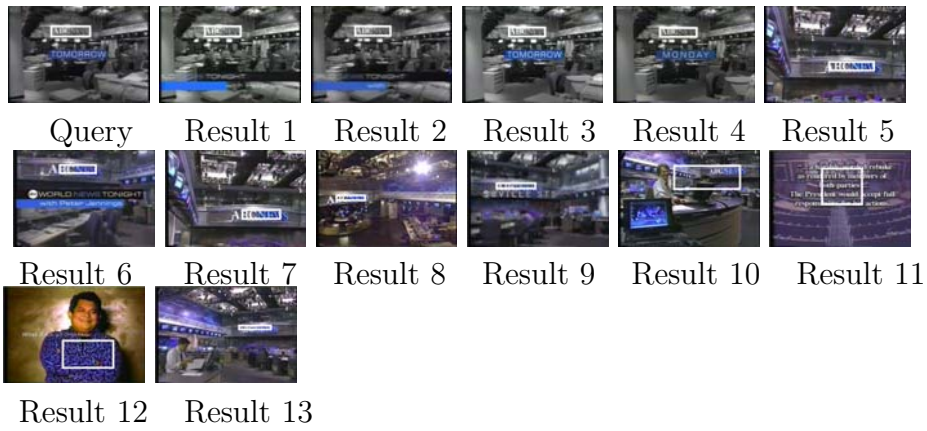


Figure 5: Search results for the object "ABCNEWS logo"

Figure 6: Search results for the object "Superimposed text in Aetna advertisement"

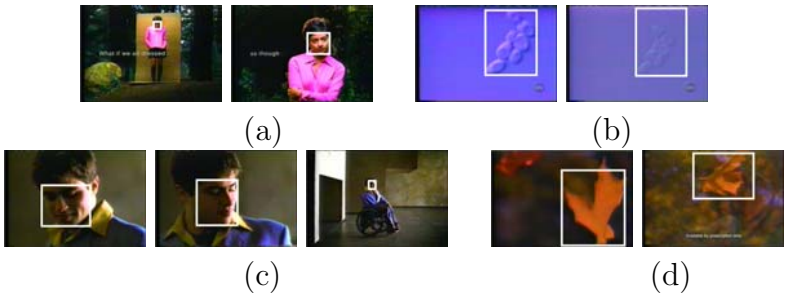


Figure 7: Unsuitable objects

(the dimension of descriptors). This comparison proves the advantage of using the smaller number of feature dimensions provided by PCA-SIFT in combination with efficient K-NN in real-world applications.

6 Conclusions and future work

In this paper, we proposed a new keypoint based object searching approach called SoftCBIR. Our main contributions include developing the energy minimization algorithm to enforce optimal geometric consistency and descriptor matching between two groups of keypoints, and using the Approximate Nearest Neighbor (ANN) for the keypoint matching instead of K-means. Experiments on ABC news videos with 1,735 keyframes in the testing set prove the effectiveness of our approach. We also pointed out object types that are not suitable for our approach.

In future work, we will explore the following directions: 1) Expand the current data set to a larger subset of the TRECVID videos, 2) Try to improve keypoint detection and representation, to increase robustness to 3D projection and non-rigid transformations, and 3) Provide mechanisms to bridge the gap between the text queries of TRECVID and the image queries of SoftCBIR, for example by using internet search to provide images that best illustrate the concepts of the text queries.

7 Appendix 1

We also developed a heuristic approach to check the neighborhood consistency and geometry consistency. This approach is faster than the energy minimization method. Its drawback is

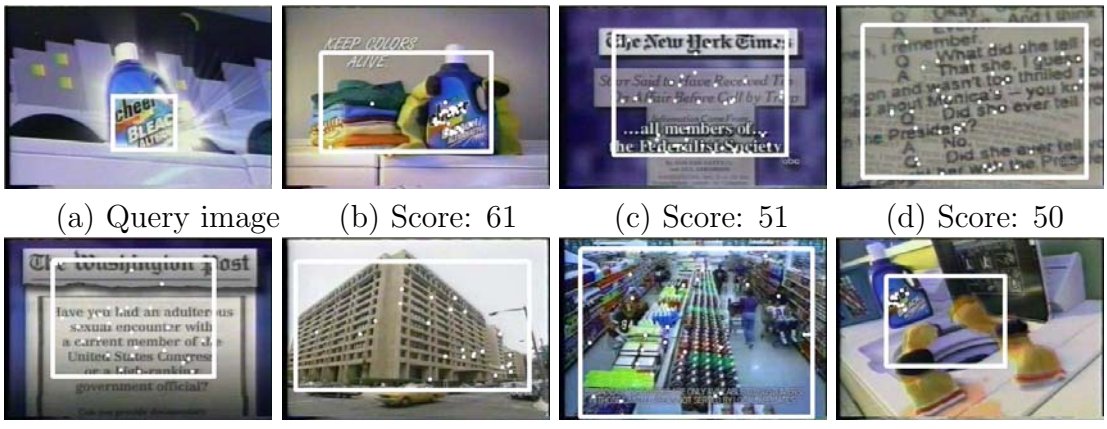
that it assumes that the scale changes between the query region and the candidate regions are not large. This heuristic approach includes a neighborhood consistency filter and a geometry consistency filter.

7.1 Neighborhood consistency filter

For object matching, only those matched keypoint pairs with enough supports from their neighborhoods are reliable. Therefore two filters are employed to check neighborhood consistency, one for filtering out neighborhood outliers and one for filtering out key points that do not satisfy the local mapping constraints. With these two filters, we effectively filter out false alarms or give them low scores for the ranking, and refine the scores and object locations for the remaining correct results.

7.1.1 Filtering out neighborhood outliers

Among the matched keypoints, a neighborhood outlier in the result image is unreliable. Although the KNN based PCA-SIFT matching is generally good, there are still errors in individual keypoint pairings. Some of these pairings lie in the result image by themselves without supporting neighbor pairings, thus should be filtered out. Fig. 8 and Fig. 9 show such an example. Fig. 8 and Fig. 9 are results of object searching in a key frame database (including 440 keyframes) from a video in TRECVID 2004. In Fig. 8, (a) is the query image with the logo of a detergent brand as the query object. The top seven ranked object searching results without neighborhood consistency filtering are shown from (b) to (h). The rectangles locate the detected object and white points indicate matched keypoints positions. Fig. 8 also shows their scores. It is easy to see that only (b) and (h) are correct results, and others are false alarms. Even in (b) and (h), the detected object regions are too large compared to the true object. Fig. 9 shows the top four ranked results after filtering out neighborhood outliers. Here the first two results are the correct ones. The score of the third result is much lower than those of the first two results. Furthermore the detected object regions in Fig. 9 (a) and (b) are much better than those in Fig. 8 (b) and (h).



(a) Query image (b) Score: 61 (c) Score: 51 (d) Score: 50

(e) Score: 44 (f) Score: 37 (g) Score: 35 (h) Score: 34

Figure 8: Object search example for criteria without neighborhood consistency filter

(a) Score: 52 (b) Score: 30 (c) Score: 16 (d) Score: 6

Figure 9: Object search example for criteria with filtering out of neighborhood outliers

We now turn to describing the algorithm in detail. Suppose the set of matched keypoints is S in a result image. For a keypoint (x, y) in S, we compute the local density of matched keypoints in the image plane around (x, y) using (6). If the density is less than a threshold, keypoint (x, y) is removed from the matched keypoint set.

$$density(x, y) = e^{-\frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i-x)^2+(y_i-y)^2}} \tag{6}$$

where $(x_1, y_1), ..., (x_n, y_n)$ are the closest n keypoints to (x, y) in S with 2D Euclidean distance, with n being an adjustable system parameter. In our current system, n=2.

7.1.2 Examining local mapping

Filtering out neighborhood outliers is the first step of neighborhood consistency filtering. After removing neighborhood outliers, there may still be matches between keypoint pairs which do not satisfy neighborhood consistency. For each matched keypoint pair, we examine the local mapping around them in their respective neighborhoods. If there are no enough support pairings from their neighborhoods for their matching, this pair will be filtered out. Fig. 10 and Fig. 11 show such an example.

(a) Query image (b) Image 24 (c) Image 46

(d) Result 1, Image 46, score: 478 (e) Result 2, Image 24, score: 310

Figure 10: Object search example without examining local mapping

Fig. 10 and Fig. 11 are also results of object searching in the same keyframe database as Fig. 8 and Fig. 9. In Fig. 10, (a) is the query image with as query object the title of The New

(a) Result 1, Image 24, score: 212 (b) Result 2, Image 46, score: 95

Figure 11: Object search example while examining local mapping

York Times in gothic font, (b) and (c) are two keyframes in the database with frame IDs 24 and 46 respectively. Image 24 and the query image share exactly the same object – "The New York Times". And Image 46 and the query image share similar objects, such as text with the same gothic font. The top two ranked object searching results without examining local mapping are shown in (d) and (e), where Image 46 is ranked first and Image 24 is ranked second. This result is encouraging because the scores of Image 46 and Image 24 are higher than those of other frames in the database, so we are successfully capturing one important feature of the object - gothic font. On the other hand, the result is not perfect because Image 46 gets higher score than Image 24, while the opposite should be true. Fig. 11 shows the top 2-ranked object searching results after examining local mapping in (a) and (b). Here Image 24 is ranked first and Image 46 is ranked second. And Image 24 gets a much higher score than Image 46. This result better reflects our intuition. And the location and size of the detected object region in Fig. 11 (a) are much better than for Fig. 10 (e).

As for the detailed algorithm, for a result image, suppose the set of matched keypoints in the query image is S, and that in the result image it is S'. For each matching pair of keypoints (K, K'), with K in S and K' in S', suppose their scales are σ and σ' respectively. We generate adaptive sizes of neighborhoods (noted as N and N') around K and K' respectively in their images.

$$\begin{aligned} N &= \{(x_i, y_i) | 1 \leq i \leq n\} \\ N' &= \{(x'_i, y'_i) | 1 \leq i \leq n'\} \\ \frac{n'}{n} &= \frac{\sigma'}{\sigma} \end{aligned} \tag{7}$$

where (x_i, y_i) are the closest n keypoints to K in S in 2D Euclidean distance, and (x'_i, y'_i) are the closest n' keypoints to K' in S' in 2D Euclidean distance. $Min\{n, n'\}$ is an adjustable system parameter for our system. For our current system, $Min\{n, n'\}$=5. That is, if $\sigma < \sigma$', n=5, n' is computed by (3), otherwise n'=5, n is computed by (7). If the number of matched pairs (K_1, K_2), with K_1 in N and K_2 in N', exceeds a threshold, we conclude that there is a local mapping between K and K' and we keep them; otherwise we delete (K, K') from the matched keypoint set. The threshold here is also an adjustable system parameter; currently it is 2.

7.2 Geometry consistency filter

After neighborhood consistency filtering, we have more accurate matched keypoint sets in each result image. But there still exist some sets of matched points that do not satisfy geometry consistency. Computing the homography by iteration is a computationally expensive approach. Fortunately we can make use of the local scale and orientation features of keypoints to simplify

the checking of geometry consistency. For each individually matched keypoint pair, $K1$ in query image and $K2$ in result image, suppose (σ_1,θ_1) records the scale and orientation of K_1, and (σ_2,θ_2) records those of K_2; we compute the scale change ratio σ and orientation change angle θ.

$$\sigma = \sigma_2/\sigma_1 \tag{8}$$

$$\theta = \theta_2 - \theta_1 \tag{9}$$

These two parameters (σ,θ) are the scaling (zooming) parameter and rotation parameter for the transformation from K_1 to K_2. If the two groups of keypoints really represent the same object, they should conform to a single planar homography transformation (assuming their depth is small compared to their distance to the camera), so the (σ,θ) set should be compact in the 2-dimensional parameter space. Therefore, we filter out those individual keypoint pair matches that are sparsely distributed in the two-dimensional parameter space (σ,θ). As for the detailed algorithm, we build a 2D parameter vector (logσ,θ) and normalize it into $(N_\sigma, N_\theta) \in$[0,1]$\times$[0,1] as described by (10) and (11).

$$N_\sigma = \begin{cases} 0, \ldots\ldots if(\sigma < 1/5) \\ \frac{\log\sigma + \log 5}{2\log 5}, if(1/5 < \sigma < 5) \\ 1, \ldots\ldots if(\sigma > 5) \end{cases} \tag{10}$$

$$N_\theta = \frac{\theta + \pi}{2\pi}, where\theta \in [-\pi, \pi] \tag{11}$$

For a parameter point (N_σ, N_θ), we compute the local density of parameter points around it in the parameter plane as (12) and (13). If the density is lower than a threshold, we filter it out from the matched keypoint set.

$$density(N_\sigma, N_\theta) = e^{-\frac{1}{m}\sum_{i=1}^{m} D((N_\sigma,N_\theta),(N_\sigma^i,N_\theta^i))} \tag{12}$$

$$D((N_\sigma, N_\theta), (N_\sigma^i, N_\theta^i)) = \sqrt{(N_\sigma^i - N_\sigma)^2 + min\{(N_\theta^i - N_\theta)^2, (N_\theta^i - N_\theta \pm 1)^2\}} \tag{13}$$

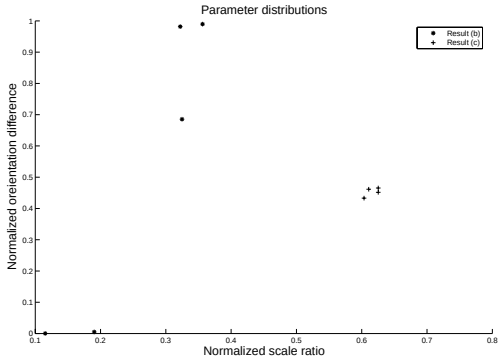
in which D($\cdot$,$\cdot$) is a distance function in the (N_σ, N_θ) parameter plane. Here (N_σ^1, N_θ^1),...,(N_σ^m, N_θ^m) are the closest m parameter points to (N_σ, N_θ) in S with the distance function D($\cdot$,$\cdot$), with m as an adjustable system parameter. In our current system, m=5. The special distance function in (13) is designed such that accounts for the fact that [-π,π] range is circular. (In normalized range, 1 represents 2π).

Fig. 12 shows such an example. In Fig. 12, (a) is the query image with query object as the word "YES"; (b) and (c) are two tied object-searching results without geometry consistency filter. They both have five matched keypoints. But they present different patterns in the 2D parameter space (N_σ, N_θ) as (d). With a geometry consistency filter, result 2 will have higher rank than result 1, because parameters of matched keypoints in result 2 are much more compact in the 2D parameter space than those in result 1.

7.3 Ranking

After neighborhood consistency filtering and geometry consistency filtering, the remaining groups of matched keypoints in the result frames are considered to be really matched objects. At this stage we again adopt a very simple way to rank frames by the number of matched keypoints that survived these filtering steps. We deliver this ranked list as the object searching result.

(a) Query image (b) Result 1 (c) Result 2



(d) Parameter distribution

Figure 12: Object search example for criteria with geometry consistency filter

8 Appendix 2

For object searching in videos, we propose a framework for shot boundary detection and keyframe extraction. It handles detection of cuts, dissolves and fades. Generally, it computes pixel-to-neighbor image differences in the videos. The cut detection applies the so-called *second-max ratio criterion* in a sequential image buffer. The dissolve detection is based on a skipping image difference and linearity error in a sequential image buffer. We also filter out the effects of camera flash. For each detected shot, we extract the middle frame as its representative image. Once keyframes are extracted, object searching in videos is equivalent to object searching in the database of keyframe images.

8.1 Introduction

Shot boundary detection is an essential elementary component of video analysis. There exist a lot of different shot boundary detection methods in the literature. In this appendix, we explain the methods we use in our video research. It is designed in a straightforward way from the pixel-to-neighbor image differences in video sequences.

Generally, there are three kinds of shot boundaries: cut, dissolve and wipe. A cut is an abrupt transition between shots which is naturally formed by the video capturing process. A dissolve is a gradual transition between shots, which is an effect added by video editors where two adjacent shots are partly overlapped, while the frame intensities of the first shot are decreased to zero and the frame intensities of the second shot are increased from zero. In fade-in and fade-out, the two shots are not overlapped but the variations of frame intensities in the two adjacent shots are similar to those in a dissolve. Therefore we use the same detection method and complete it with a post-processing step to account for the fact that the shots are not overlapped. A wipe is a digital

video effect also generated by video editors that can have many different forms. In a wipe, one new shot pushes away an old shot. In this appendix we only describe algorithms for cut detection, dissolve and fade detection.

8.2 Cut Detection

8.2.1 Second-Max Ratio Criterion

The image difference between two adjacent frames can be a good cue for detecting cuts. But motion within one shot may be so large that it can also cause a noticeable image difference and can be confused as a cut. To deal with motion, we use the second-max ratio criterion in the image difference sequence to detect cuts. The second-max ratio, $R(t)$, is defined as

$$R(t) = \frac{d(t)}{max_{t' \in [t-w_1, t+w_1], t' \neq t} d(t')} \tag{14}$$

in which $d(t) = ||I(t+1) - I(t)||$ is the pixel-to-neighbor image difference between two adjacent frames and is described in the next section, w_1 is the half-width of a sliding window, and max $d(t')$ is the maximum among all image differences $d(t)$ between adjacent frames in the sliding window, excluding the frame at time $= t$ considered at the numerator of the expression. Therefore, when that frame is located at a maximum of $d(t)$, the denominator selects the second maximum in the sliding window, since the first one is excluded from consideration. If the second maximum is not small, as is likely for high level of motion, then the ratio remains small. In case of a cut, there is no large second maximum, so the ratio becomes large when frame t is just past the cut. Fig. 13 demonstrates the effectiveness of $R(t)$. In this figure, there is a segment (around position c) with large motion. A series of large responses in are shown in $d(t)$. But in $R(t)$, these are correctly eliminated. All peaks in $R(t)$ correspond to real cuts.

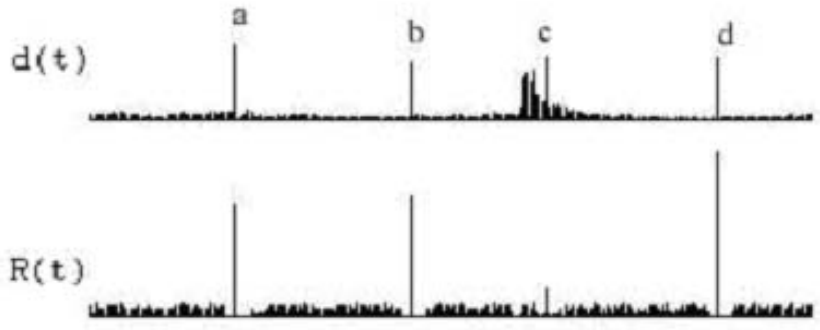


Figure 13: Effectiveness of second-max ratio criterion in detecting cuts. Note that the peaks due to motion that appear in $d(t)$ do not appear in $R(t)$

8.2.2 Image Difference Computation

There are many ways to compute the difference between two images. The following three are popular:

1. Pixel-to-pixel difference

2. Pixel-to-neighborhood difference

3. Histogram difference

We can think of the image difference as a distance between two vectors in a feature space. In the first and second methods, the features are in the pixel space, with the color of each pixel as one component of each feature. In the third method the features are in histogram space, with each histogram bin a component of each feature. The first method is not robust to motion in videos. The second method is an advanced version of the first method with a degree of motion compensation. The third method can be computed by the Euclidean distance or any other defined histogram distance in the histogram space. One of the drawbacks of the third method is that it totally discards the spatial distribution of the images. Our experiments support our analysis that the second method, pixel-to-neighborhood difference, is the best for evaluating image difference. In our implementation we compute the pixel-to-neighborhood difference as

$$\begin{aligned} d(t) &= ||I(t+1) - I(t)|| \\ &= c\sum_{i,j}(min_{k\in[i-w_2,i+w_2],l\in[j-w_2,j+w_2]}(\sum_{m\in R,G,B}|I_{k,l}(t+1)[m] - I_{i,j}(t)[m]|)) \end{aligned} \tag{15}$$

in which c is a scalar constant, w_2 is the semi-width of the motion compensation searching window, $I_{i,j}(t)[m]$ is the channel value corresponding to m for the pixel position (i, j) of the frame image $I(t)$, with $m \in R, G, B$.

8.3 Dissolve Detection

In dissolve detection, we cannot use the image difference between two adjacent frames, because it is small during the dissolve. But there will be a large image difference between two frames if we skip an interval (such as a 25 frame interval). This skipping image difference can be used as a cue to dissolve detection. But it is not a unique criterion for finding a dissolve. It is very common that a peak of the skipping image difference in a sequence is not a dissolve (it can be a cut, a wipe, a shot with large motion, or the combinations of these events). Therefore we need to add other criteria for reliable dissolve detection. Another criterion we can use is the degree of linearity of a sequence of frames. This is because, in the mechanism of dissolve effect generation, most video editing processors use a linear combination of the signal before the dissolve transition (noted as signal A) and the signal after the transition (noted as signal B). Thus in this model the dissolve is a simultaneous fade-out of the signal A and fade-in of signal B. This implies that, during the dissolve transition, images change linearly. Hence the degree of linearity can be selected as another criterion to dissolve detection using its generative properties. Technically, it can be evaluated from the normalized linear error within a sequence of frames. We propose a method for dissolve detection which combines the above two criteria. Suppose we currently have a sequence of w images which is a segment in the video sequence starting with $I(t)$ and ending with $I(t+w-1)$. We define the current skipping image difference value as

$$
\begin{aligned}
D(t) &= ||I(t+w-1) - I(t)|| \\
&= c\sum_{i,j}(min_{k\in[i-w_2,i+w_2],l\in[j-w_2,j+w_2]}(\sum_{m\in R,G,B}|I_{k,l}(t+w-1)[m] - I_{i,j}(t)[m]|))
\end{aligned}
\tag{16}
$$

We define the current normalized linear error along this part of video as

$$
LE(t) = \frac{\sum_{i=0}^{w-1}||I(t+1) - ((1-\frac{i}{w-1})I(t) + \frac{i}{w-1}I(t+w-1))||}{||I(t+w-1) - I(t)||} \tag{17}
$$

We detect a dissolve by the simultaneous presence of a peak of $D(t)$ and a valley of $LE(t)$. Fig. 14 shows an example of detected dissolves from a news video. Here we have four dissolves that all satisfy our dissolve model. Fig. 14 also illustrates that we can easily estimate the start frame number and end frame number of a dissolve transition.

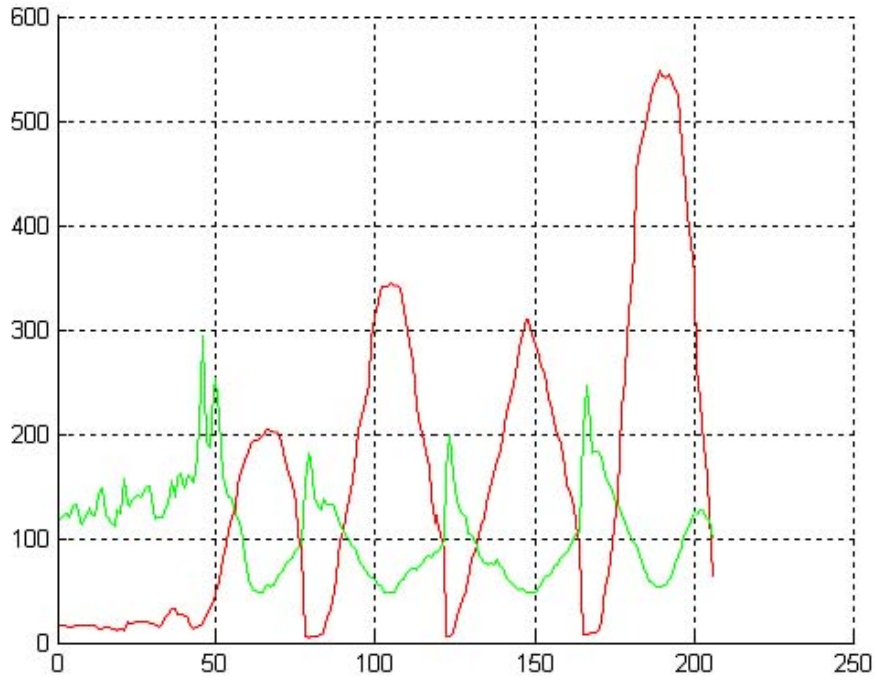


Figure 14: Four dissolves with simultaneous peak of $D(t)$ (darker red curve) and valley of $LE(t)$ (lighter green curve)

In (17) , we use the linearity assumption in RGB color space. One concern could be that in analog television and in MPEG encoding, other color spaces are used. In which color space is the dissolve signal linear? In fact, these three color-spaces are equivalent in linearity, because there exists a linear transform between each pair of the three color-spaces. We also did some

experiments to show this equivalence. For example, Fig. 15 shows the $LE(t)$ curves for RGB color space and YUV color space and their shapes are similar. In our implementation, we compute the linearity errors in the RGB color space.

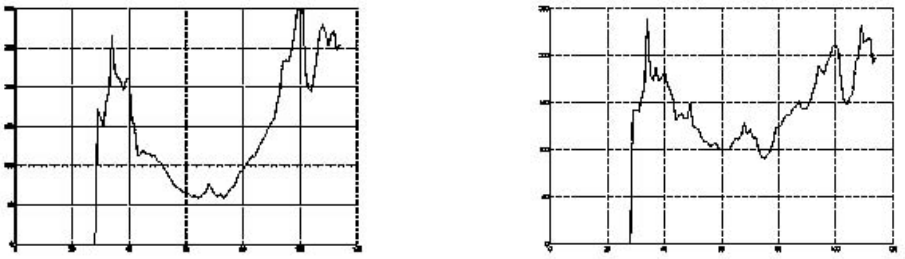


Figure 15: $LE(t)$ in RGB color space (left picture) and YUV color space (right picture)

8.4 Elimination of Flash

There may be a lot of camera flash in videos, especially in news videos. A high-intensity light of very short duration is produced. Flash will cause a false alarm for cut detection, because it will increase the global brightness of one or more frames dramatically. Our method to eliminate false alarms caused by flash from candidate cuts uses the following two facts: (1) the duration of flash is very short, generally only one or two frames, (2) the images before a flash are similar to the images after the flash, if they are in the same shot. Following the method described above, we can get candidate cuts by computing $R(t)$. These candidate cuts will be filtered by our post-processing module to eliminate false alarms caused by flash. For a candidate cut position, noted as t, we examine the following four values: $||I(t+1) - I(t-2)||$, $||I(t+1) - I(t-1)||$, $||I(t+2) - I(t)||$, $||I(t+3) - I(t)||$. If any of these four values is less than a threshold, this candidate is considered to be a false alarm caused by flash, and filtered out from candidate cuts. This algorithm performs quite well in experiments, and filters out most of the false alarms caused by flash. Also notice that, in the special cases when the flashlight happens to be at the real cut boundary position, our algorithm also works. It will not filter out the cut because the images before the cut and those after the cut have large differences.

8.5 Merging Adjacent Fade-Out and Fade-In

Fade-out and fade-in are very common in videos. Fade-out is a dissolve from an image (noted as image A) to a black frame, while fade-in is a dissolve from a black frame to an image (noted as image B). In most cases, a fade-out will be followed by a fade-in. In the framework introduced above, we will get two dissolve transitions in this case. One is the preceding fade-out, and the other is the subsequent fade-in. But this is not a reasonable output. In the shot boundary detection output, we should deliver only one dissolve from image A to image B. Therefore we have a post-processing module to merge adjacent fade-out and fade-in into one dissolve transition. In this post-processing module, for a candidate dissolve transition, we analyze the ratio of the number of black frames during the transition divided by the total number of frames of the transition. If this ratio is over 0.5, we consider the transition to be a fade-in or fade-out. We merge two such

Table 4: Experimental Results

Run-id	Recall for all	Precision for all	Recall for cut	Precision for cut	Recall for dissolve	Precision for dissolve	F-recall for dissolve	F-precision for dissolve
D0	0.802	0.847	0.900	0.877	0.595	0.765	0.567	0.792
D1	0.816	0.833	0.900	0.880	0.639	0.719	0.536	0.795
D2	0.809	0.842	0.900	0.877	0.617	0.750	0.568	0.795
D3	0.846	0.693	0.923	0.733	0.685	0.599	0.554	0.741
D4	0.731	0.908	0.864	0.921	0.450	0.859	0.656	0.787

close detected fade-in / fade-out transition components to one new dissolve transition. In our implementation, a frame is declared to be a black frame if at least 90% or all the pixels are such that $R < 25$, $G < 25$ and $B < 25$.

8.6 Implementation

In our implementation, we combined our algorithms with an MPEG1 decoder. In the decoding, we keep a buffer of 25 sequential frames. It is a cyclic pointer array. Each pointer in the array is pointing to an image structure. Every time a new frame is decoded, the frame buffer is updated correspondingly and the features $d(t)$, $R(t)$, $D(t)$ and $LE(t)$ are computed based on the current frame buffer. The update rule is that the newly decoded frame comes in the buffer and pushes out the pointer for the oldest frame. The array is called cyclic because if the last decoded frame is replaced at the end of the array, the current newly decoded frame will be replaced at the start position of the array. This array is very convenient for memory management and the only overhead is a pointer pointing to the currently processed position in the buffer.

8.7 Experimental Results

We submitted five runs to TRECVID 2004. Our run IDs are D0, D1, . . . , D4. Generally, they are obtained by the same algorithms for feature extraction and shot boundary detection decision. The difference between the five runs we submitted is with the different values of parameters in our algorithms, including the w, w_1, w_2 parameters introduced above, the thresholds for $R(t)$, and the thresholds for finding the peaks of $D(t)$ and valleys of $LE(t)$. The results are shown in Table 4. They show the tradeoff between precision and recall. F-recall and F-precision in Table 4 are frame-based recall and frame-based precision respectively.

As a more thorough analysis, we find that 29% of all missed gradual transitions are caused by wipes. Wipes are difficult to detect effectively in videos because they are of various types and the pixel-to-neightbor image differences have different properties for different styles of wipes. We also find that 25% of all missed cut transitions are caused by extremely short dissolves. In TRECVID 2004 data, there are a lot of extremely short dissolves involving only three frames. These transitions are detected as dissolves in our submission, but in the TRECVID ground truth, they are counted as cuts.

8.8 Future Work

Papers submitted by other participants propose different methods for doing shot boundary detection. Tsinghua University [26], RMIT University [27], and FX Palo Alto Lab [28] obtained the best results. Tsinghua University considers motion information in shot boundary detection. RMIT University proposes an impressive PrePostRatio to detect gradual transitions. FX Palo Alto Lab views shot boundary detection as a general supervised classification problem rather than an ad-hoc peak detection. All these ideas are illuminating. Combining them to our work would be helpful. Moreover, our own methods can still be improved. For example, in the current implementation, the width of the sliding window is fixed at 25 frames. An adaptive sliding window width may perform better.

References

1. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, An Optimal algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions, Journal of ACM, Vol. 45, pp. 891–923, 1998.

2. S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24(2), pp. 509–522, 2002.

3. R. Fergus, P. Perona, and A. Zisserman, Object Class Recognition by Unsupervised Scale-invariant Learning, Vol. 2, pp. 264–271, Proceedings of International Conference on Computer Vision, 2003.

4. W. Freeman and E. Adelson. The design and use of steerable filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13(6), pp. 891–906, 1991.

5. J. H. Friedman, J. L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software, Vol. 3(3) pp. 209–226, 1977.

6. S. Gold, A. Rangarajan, C.P.Lu, S. Pappu, andE. Mjolsness, New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence, Pattern Recognition, vol. 31, pp. 1019 C 1031, 1998.

7. L. Van Gool, T. Moons, and D. Ungureanu. Affine / photometric invariants for planar intensity patterns. Proceedings of European Conference on Computer Vision, pp. 642–651, 1996.

8. C. Harris and M. Stephens. A combined corner and edge detector. Alvey Vision Conference, pp. 147–151, 1988.

9. http://www-nlpir.nist.gov/projects/tv2004/tv2004.html

10. A. Jain and R. Dubes, Algorithms for Cluster Data, Prentice Hall, Inc., 1988.

11. F. Jurie and C. Schmid, Scale-invariant shape features for recognition of object categories, Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, pp. 90–96, 2004.

12. T. Kadir, A. Zisserman, and M. Brady, An affine invariant salient region detector, Proceedings of European Conference on Computer Vision, Vol. 1, pp. 228-241, 2004.

13. Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. Proceedings of the Conference on Computer Vision and Pattern Recognition, Volume I, pp. 511–517. 2004.

14. Y. Ke, R. Skthankar, and L. Huston, Efficient Near-duplicate Detection and Sub-image Retrieval, Proceddings of ACM International Conference on Multimedia, 2004.

15. J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. Biological Cybernetics, vol. 55, pp. 367–375, 1987.

16. S. Lazebnik, C. Schmid, and J. Ponce. Sparse texture representation using affine-invariant neighborhoods. Proceedings of the Conference on Computer Vision and Pattern Recognition, Vol. 2, pp. 319–324, 2003.

17. D. G. Lowe. Object recognition from local scale-invariant features. Proceedings of International Conference on Computer Vision, pp. 1150–1157, 1999.

18. D.G. Lowe, Distinctive Image Features from Scale- Invariant Keypoints, International Journal of Computer Vision, Vol. 60, pp. 91-110, 2003.

19. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004.

20. P.Moreels and P. Perona. Common-Frame Model for Object Recognition. Advances in Neural Information Processing Systems, 2004.

21. G. Salton and M. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, 1983.

22. F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets. Proceedings of European Conference on Computer Vision, pp. 414-431, 2002.

23. J. Sivic and A Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. Proceedings of the International Conference on Computer Vision, Vol. 2, pp. 1470–1477, 2003.

24. J. Sivic and A. Zisserman. Video Data Mining Using Configurations of Viewpoint Invariant Regions. Proceedings of the Conference on Computer Vision and Pattern Recognition, Vol. 1, pp. 488–495, 2004.

25. H. Spath, Clustering Analysis algorithms for Data Reduction and Classification of Objects, Ellis Horwood, Chichester, 1980.

26. J. Yuan, W. Zheng, Z. Tong, L. Chen, D. Wang, D. Ding, J. Wu, J. Li, F. Lin, B. Zhang. Tsinghua University at TRECVID 2004: Shot Boundary Detection and High-level Feature Extraction, TRECVID 2004.

27. T. Volkmer, S. Tahaghoghi, H. E. Williams. RMIT University at TRECVID 2004, TRECVID 2004.

28. M. Cooper, T. Liu, E. Rieffel. Shot Boundary Detection Combining Similarity Analysis and Classification, TRECVID 2004.

Acknowledgement

Special thanks to Xu Liu, Mei Huang, Xue Mei, Yang Yu and Zhe Lin for helping me label ground truth. Thanks to Eugene Borovikov and Raphael Bousquet for their advices during system implementation.