Video Retrieval based on Tracked Features Quantization

Hiroaki Kubo, Julien Pilet, Hideo Saito.
Graduate School of Science and Technology
Keio University
Tokyo Japan
{*khiro, julien, saito*}*@hvrl.ics.keio.ac.jp*

Shin'ichi Satoh.
National Institute of Informatics
Tokyo Japan
satoh@nii.ac.jp

***Abstract*—In this paper, we present an image retrieval method based on feature tracking. Feature tracks are summarized into a compact discreet value and used for video indexing purpose. As opposed to existing space-time features, we do not make any assumption on the motion visible on the indexed videos. As a result, given an example query, our system is able to retrieve related videos from a large database. We evaluated our system with the copy detection benchmark MUSCLE-VCD-2007. We also ran retrieval experiment on hours of TV broadcast.**

***Keywords*-Video retrieval, Content based retrieval, Descriptor, Quantization.**

I. Introduction

With the quickly growing amount of video material available on-line, organizing and retrieving sequences is of primary importance. The classical approach to video retrieval is to rely on keywords. The indexed material have to be tagged beforehand, potentially automatically [4]. However, in some cases, it can be difficult to represent with keywords a query to a video database. Another option is to use an example as the query, which can be more intuitive. Some methods, given a single image as a query, efficiently search through a large database of images to extract relevant ones [7], [5]. These approaches detect feature points on the images and index them using quantized descriptors. Using such an approach to index a video sequence requires indexing independently every frame or key-frames, which has drawbacks. The redundancy between frames is poorly exploited, yielding large index tables. Inspired by the success of image retrieval techniques, we propose a video retrieval system that quantizes tracks of features. The descriptors along the track are summarized together into a single index entry. Doing so has the advantage of handling properly the redundancy between consecutive frames. It can also easily exploit a video sequence as a query, even in the case of a static scene filmed by different cameras.

To index a sequence based on a feature tracked over multiple frames, we rely on a method that follows two steps of quantization [6]. First, a pre-built k-mean tree quantizes the descriptors at a frame level. They are then grouped along their tracks into histograms. These histograms, in turn, are quantized to capture the different appearances the point can take.

Capturing trajectories of image features is an efficient approach to video copy detection [2], [9] and to action recognition [8]. However, these methods fail to handle cases in which the same scene is shot by different cameras, following separate paths. Our approach aims at describing a single physical point visible on several frames, by summarizing all the available descriptors into a single one. Relying on such a feature track representation is a new approach to recognize and measure similarities of video. As a result, our method can retrieve videos of a static or dynamic scene shot from different viewpoints.

The rest of the paper is organized as follow. We first give a brief overview of the way to quantize tracks [6]. Section III presents the new representation of videos. We present video retrieval results in Section IV.

II. Characterizing tracks of features

This section summarizes a technique to describe tracks of features [6]. We rely on the well known Scale Invariant Feature Transform (SIFT [3]) to obtain local features and their corresponding 128 dimensions descriptors. Our method quantizes the descriptors using a K-mean tree, as suggested by [5]. The K-mean tree is built by recursively clustering sample descriptors acquired during a first training phase. Our tree has at most 4 branches at every node and has about 60000 leafs. This approach reduces the feature description to a single discreet value.

SIFT features are then matched from frame to frame, using simple normalized cross-correlation (Fig.1a). The result is a set of stable tracks of features forming a sequence of K-mean tree leafs. Counting the number of occurrences of each leaf in the sequence produces a sparse vector of roughly $60K$ dimensions, most of which being zero. This vector, or histogram, is an estimation of the distribution of the descriptors caused by the underlying real physical point. It takes into account the instability of the feature detector. Such a vector is depicted by Fig. 1(b), under the tree, using gray levels to represent its values.

During a second training phase, we collect many of these histograms and cluster them using agglomerative clustering.

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.794

IEEE computer society

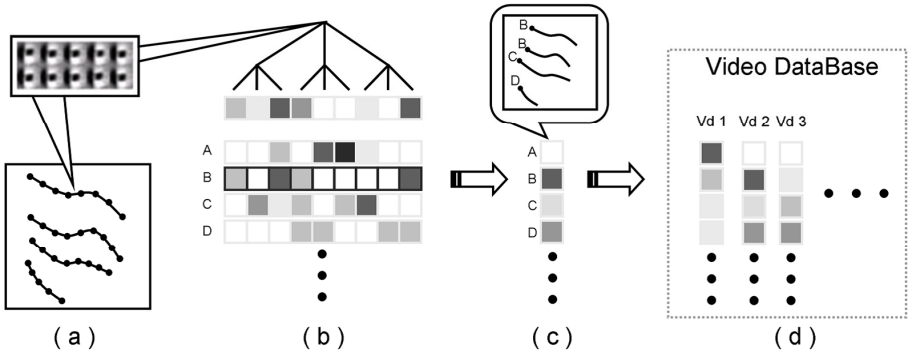

Figure 1. Structure of our method. (a) Keypoints are tracked from frame to frame. Descriptors are extracted at each frame. (b) The descriptors are quantized using a k-mean tree. The descriptors of a track distributes on the tree's leafs, forming an histogram. This histogram is in turn vector quantized with the pre-computed prototypes called A,B,C, and D on this figure. In this example, the track's histogram resembles B. (c) Following a similar process, each track of the video is assigned to a prototype. The vector counting the number of occurrences for each prototype describes the video sequence. (d) During retrieval, our system compares this description with the ones of the indexed videos.

About $30K$ representative prototypes remain. At runtime, the system determines which prototype best matches the observed tracks. Each track can then be described by a stable discreet value we call Feature Track ID (FTID). Fig. 1(c) denotes FTIDs with A,B,C, and D.

III. Video Representation

In this section, we describe how to represent, index, and retrieve videos using the previously presented FTID. Basically, FTIDs are used as *words* in a *bag of word* model.

Given a video sequence A, our system extracts a number of feature tracks and assigns FTIDs to them. Let $\hat{a}_i$ denote the number of occurrences of FTID i in the sequence A. Our system then normalizes the histogram:

$$a_i = \max\left(\frac{\hat{a}_i}{\max_j(\hat{a}_j)} - t, 0\right) . \quad (1)$$

In other words, we calculate Feature Track ID for all tracks in the video and count them to make an histogram. At the end of the sequence, the histogram is normalized with its maximum value. We then set a threshold to eliminate too small bins to represent videos. In our case, the threshold is set to $t = 0.1$, to ignore bins which did not reach 10% of the largest one.

The indexing process corresponds to compute Eq. 1 for each video. For retrieval, the query video Q is compared to index sequences using cosine similarity. For the particular video X, the similarity is:

$$S(q, x) = \frac{qx}{\|q\|\|x\|} . \quad (2)$$

Where q and x are the FTID histogram of video Q and X. Computing the similarity for all indexed videos can be achieved quickly using inverted tables. For each FTID i, a table lists the indexed movies in which the bin i is greater than 0. By considering only the movies listed in tables corresponding the non-zero dimensions of the query q, the system has to process only a limited number of sequences.

IV. Experiments

Here we describe the two experiments we conducted to evaluate our system. Because no common video retrieval benchmark exists, we tested our system's ability to achieve copy detection.

We tested our system with the benchmark of MUSCLE-VCD-2007 [1]. Although our method is not designed for copy detection, our system achieves over 70 percent of correctness.

To test the video retrieval capabilities of our method, we indexed about three video database and ran queries on it. The results of video retrieval are shown in Fig:2.

A. Copy Detection

The copy detection task consists in determining whether or not the query video has been copied in one of the 101 indexed videos, and if yes, which in which one. The benchmark contains 15 query videos which either do not contain any common part with the database, or contains modified portions of it.

To index the 101 movies, we first divide them into short shots by comparing the color histogram and number of tracked points between consecutive frames. We also split the query sequence with the same method. Using Eq.2, each shot finds the best matching one in the database and votes for the corresponding movie. We also average similarity best scores to obtain a movie level score. Thresholding it at 0.3 tells whether the query is copied in the database or not.

Table I shows the retrieval scores. Our method achieves a classification score of 73 percent. Most of the computation time is spent extracting SIFT features. Our method failed to handle mirroring and flipping effects, since the SIFT descriptor is not designed for invariance to such transformations.

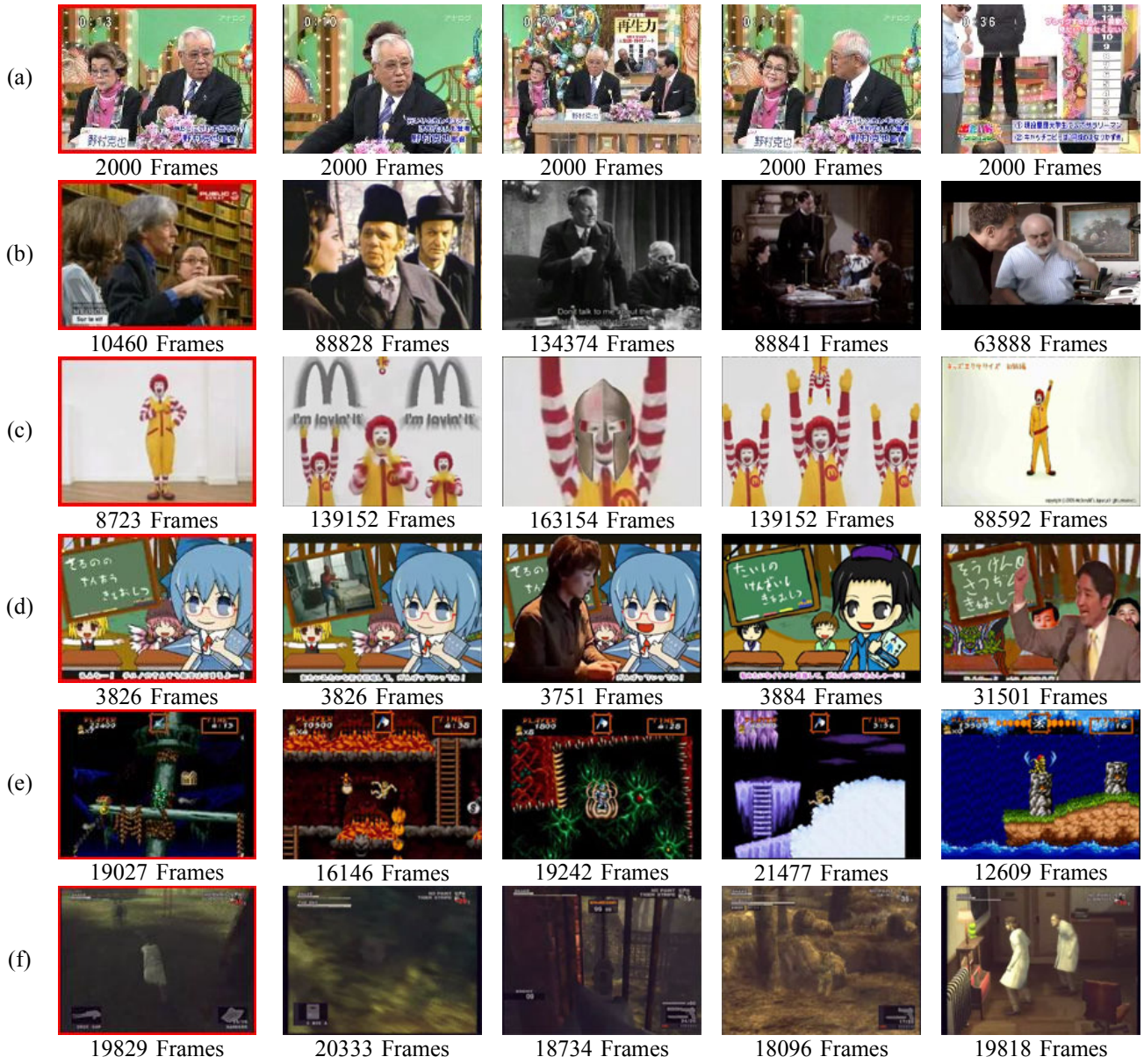

Figure 2. Video Retrieval Results. Videos are represented in this figure by manually selected frame. The leftmost image of each row is the query video. The few first results appear on its right, in the retrieval order. (a) The system retrieves shots of a TV show, from a 3 hours TV recording. (b) The system retrieves scenes with a few people from the MUSCLE-VCD-2007 database (about 100 hours). (c) Our system retrieves several videos of Ronald Mac Donald from a database of 80 shorts movies downloaded on the internet (total: 6 hours). (d) Our method finds modified versions of a cartoon from the same database. (e) and (f) The system retrieves some gameplay videos of same series.

	Score	Search Time
Best Run(IBM-1)	0.83	44min
Average	0.7	54min
Our method	0.73	80min

Table I
RETRIEVAL SCORE AND TIME

B. *Video Retrieval*

We show some video retrieval results calculated by our method in Fig2.

In our first experiment, we recorded 3 hours of TV programs, including news, variety, and commercials. We split the video in parts of 2000 frames and handle them independently. Fig. 2(a) depicts an example query selected among the 3 hours recording. The query shot is part of a variety show. The system retrieves other parts of the same show. In this example, many feature track could be matched on the background.

We also conducted a retrieval experiment using the MUSCLE-VCD-2007 as a database. As opposed to the copy

detection experiment, we used each video directly with the similarity score of Eq. 2. We use a copy detection query which does not have ground truth in database. Because of the variety of MUSCLE-VCD-2007 database, the contents of results are independent, but these videos have similar portions which we show in Fig2(b).

In addition, we tested our method on 6 hours of video we downloaded from Nico-Nico Douga. The database has 80 sequences, some of which showing modified versions of others. Several video might also share the same subject. In both case, our method succeeded in retrieving the related videos. As shown by Fig. 2(c), different sequences showing the same clown could be retrieved, despite changes in scene, background, and actions. In the example of Fig. 2(d), several cartoon videos share a similar background. Our method successfully retrieved all the related videos, in spite of modified frame contents, newly added characters, and other changes. In the example of Fig. 2(e) and (f), our method retrieved some gameplay videos of same series.

V. Discussion and Conclusions

We presented a representation of videos using FTID-histogram which is able to capture the instability of feature points. By efficiently quantizing feature track information, our approach can represent videos in an efficient and compact manner. Because of FTID's robustness to keypoint's instability, our approach can extract discriminative features from videos. In addition, as depicted by Fig. 2, our method can handle severe transformations of videos.

Our method, as most feature-based image retrieval approaches, ignores the spatial relationship of features in the similarity score, thereby decreasing the quality of the retrieved results. In future work, we consider to use the relative locations of features to improve retrieval.

References

[1] N. B. J. Law-To, A. Joly. Muscle-vcd-2007: a live benchmark for video copy detection, 2007. http://www-rocq.inria.fr/imedia/civr-bench/.

[2] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa. Robust voting algorithm based on labels of behavior for video copy detection. In *Proceedings of the 14th annual ACM international conference on Multimedia*, page 844. ACM, 2006.

[3] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[4] R. Morzinger, R. Sorschag, G. Thallinger, and S. Lindstaedt. Automatic image annotation using visual content and folksonomies. In *Proceedings of the Metadata Mining for Image Understanding Workshop at VISAPP2008, Funchal, Madeira, Portugal*, 2008.

[5] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition*, 2006.

[6] J. Pilet and H.Saito. Virtually augmenting hundreds of real pictures: An approach based on learning, retrieval, and tracking. In *IEEE Virtual Reality*, Boston, MA, March 2010.

[7] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, volume 2, pages 1470–1477. Citeseer, 2003.

[8] J. Sun, X. Wu, S. Yan, L. Cheong, T. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

[9] X. Wu, Y. Zhang, Y. Wu, J. Guo, and J. Li. Invariant visual patterns for video copy detection. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 1–4, 2008.