Nanjing University in TRECVID 2009

Yang Yang, Jingwei Xiao, Kang Lin, Gangshan Wu, Tongwei Ren, Yaqiong Wang
State Key Lab for Novel Software Technology, Nanjing University
{yang,xiaojw,lingkang,rtw,wyq}@graphics.nju.edu.cn, gswu@nju.edu.cn

Abstract

This report describes the details of our system for copy detection task in TRECVID 2009. We submitted 4 runs, 2 for video-only queries and 2 for video + audio queries. Details are listed below. Each group is with same technique but different parameters.

NJU.v.balanced.1, **NJU.v.nofa.1**: SURF feature is used. Videos are indexed with feature points. We adopt LSH as indexing technique.

NJU.m.balanced.1, NJU.m.nofa.1: SURF feature is used. Feature points are made to BOF before videos are indexed. A post process of SURF points pairing is included.

For video-only queries, LSH (Locality-sensitive Hash) was built using SURF points but the approach cost too much memory. To save time and space cost, we generated a global feature for each frame using bag-of-features when dealing with video + audio queries, and ignored audio information in both query and reference media. Because it's our first time participation, our goal is to complete whole process of TRECVID copy detection task, get experience of the key technologies in CBCD.

Keywords

Content-based copy detection, Speeded Up Robust Features, Bag-of-features, Locality-Sensitive Hashing, Approximate Nearest Neighbor Search

1. Introduction

Video copy detection technologies are recently considered rather useful especially on copyright protection and advertising monitoring. The main task is to find similar video copies from the data base, using image/video features and other computer vision approaches. This kind of technologies can also be used in redundancy reduction of huge database.

TRECVID, which is held by NIST, is an evaluation program in video retrieval domain and video copy detection is also included.

This is our first time participant of TRECVID Copy Detection task. With regard to the special requirement of this task, we adopted Speeded Up Robust Features (SURF) [1][4] as visual feature. We chose two different schemes of indexing in video-only and video + audio task. They are Locality-sensitive Hashing and Random Kd-tree Searching. We wish to evaluate the performance of these two schemes on huge data set. The experiment results brought us some experience.

Our system was implemented with C++ and OpenCV(Open Computer Vision Library) [5] library. Unfortunately, the TRECVID Copy Detection Sample Generator [9] from INRIA-LEAR failed to return expected video queries on any of our computers. So, we had no chance to refine our system before submitting results.

2. Video-only Task

Our system extracts SURF feature points from keyframes selected from reference videos. In offline phase, SURF points from reference video are extracted and stored. The system then constructs an index for them. In querying phase, features are extracted in the same way as reference videos. In order to find the related video in the reference video set, for each SURF point of query video, the system finds its nearest neighbors among points of reference video. Then, a voting procedure was set up to select the qualified videos. The general scheme of our system in video only task is show in figure 1.

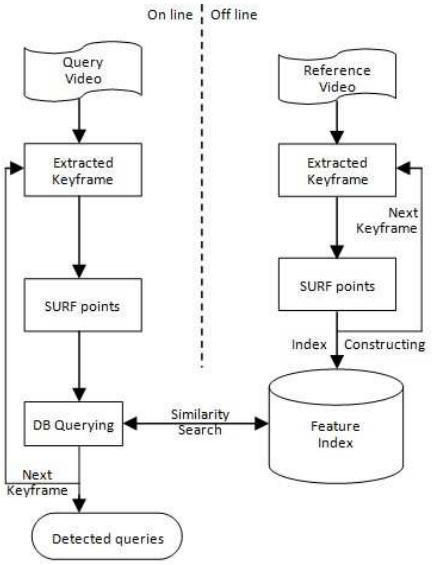


Figure 1.
Procedure of our method in video-only task

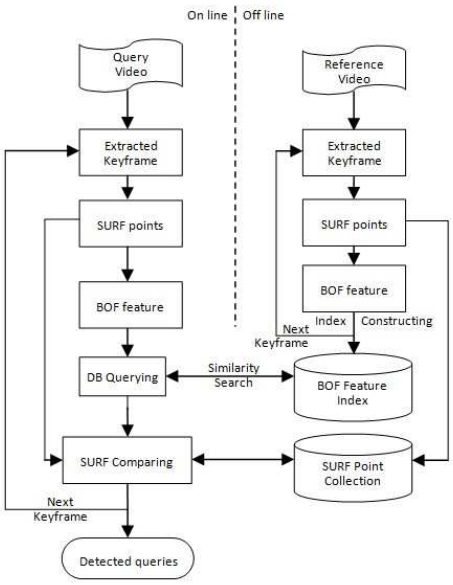


Figure 2.
Procedure of our method in video + audio task

2.1 Frame Feature Extraction

The reason why we adopt Speeded Up Robust Features is that its extraction speed is relatively fast. Because it is scale-invariant and robust, most of the transformation can be deal with. An upright version of SURF (U-SURF) was used for there's no rotation in the 10 transformations. We extracted the "I" fames in videos as keyframes where we extract SURF points from, avoiding prediction and compensation in compressed domain. The sample rate is about 2 fps.

2.2 Feature Points Indexing

In video-only task, our system adopted Locality-sensitive Hashing to construct point index. We used one version of implementations called "E2LSH" [6] by MIT. To accelerate the voting scheme, we tried to return all points in the bins which query points was hashed to, without linear comparing.

2.3 Dealing with Flip (vertical mirroring)

Both position and local descriptor of SURF points are being changed while vertical mirroring. For an upright surf point, vertical mirroring just changes sequence of some dimensions, and signs of X-Haar responses. By changing sequence of some dimensions and negating some values, the descriptor can be flipped. Our system automatically generates flipped descriptor while extracting from queries. The flipped ones participate voting right after the original ones.

2.4 Result Analysis

Our results were shown in figure 3 and seem not so good. After we submitted the results, we started looking for the problems. One of the reasons is from our index scheme.

LSH is a multi-dimensional index technique which is very popular recently. The index structure is presented as hash tables. The number of hash tables is one of the parameters which determine the LSH performance. We had done a simple evaluation on LSH after we submit the result. We found that if the parameters of hash tables are very close to optimal ones, the points in one hash bin are pretty close. There's no problem returning all elements in one bin. Otherwise, the result would be unacceptable.

A hash table may cost a bit of memory depending on the count of points. Since our system directly build index using SURF points, single PC can hardly hold the memory used by searching and voting scheme, even a server. We adopt a group of 5 PCs to handle this, each with 4GB memory. Reference data were split into 5 pieces and deployed. Voting results were joined together to generate the final result. Though we run it under such configuration, we can still keep no more 10 hash tables in memory. This is far away from the optimal settings, because for 64 dimensions, LSH required about 160 hash tables. In order to reduce the requirement, we adopted feature of SURF-16, reducing 64 dimensions to 16. We have done some statics on each dimension using TRECVID videos, finding that rounding bins made much less contribution to descriptor distance than center bins. These approaches leaded us to final result, but not a satisfying result.

3. Video + Audio Task

Since the approach we used in video-only was not so good and successful. We made a lot of changes in visual feature processing and indexing. After SURF points extracted from keyframes, instead of building index directly, we generate a global feature for each keyframe. Index is built with global features. Query keyframes are being indexed to find similar frames. Then, a further step of SURF point comparing is performed. The procedure is shown in figure 2.

3.1 Global Feature

The global feature we generated is called "visual bag-of-words" or "bag-of-features" [3], which is a histogram of SURF descriptors on a chosen set of bins. We selected the bins by performing a K-MEANS algorithm on a sampled set of SURF points from reference data. Feature histogram was calculated by:

$$H(x) = \sum_{i=0}^{n} G(c_x, p_i),$$

$$G(c, p) = \begin{cases} \frac{1}{\sqrt{\pi\sigma}} e^{-\frac{|c-p|^2}{\sigma}} & ,|c-p| < 3\sigma \\ 0 & ,otherwise \end{cases}$$

in which c_x is the x_{th} bin center, and p_i is the i_{th} SURF point in point set. Since SURF point has a Laplacian sign marking the point to be positive or negative, two kind global features were generated, one from positive points and one from negative ones.

3.2 Index Construction

Indexing scheme we chose this time is random kd-tree search algorithm [7] provided by toolkit of "FLANN" [8]. Unlike LSH, kd-tree's memory cost just depends on the count of SURF points (approximately a little more than the memory to store SURF points), and more stable. We first apply the parameter selection algorithm provided in [7] on a sampled dataset to obtain optimized parameters.

3.3 SURF Comparing

In order to get a more accurate similarity between query frame and reference frame, a further step comparing was performed using SURF points. To accelerate, we used a kd-tree search algorithm implemented by "ANN tool kit" [9] instead of linear searching. The new similarity was measured by points paired.

3.4 Audio Feature Extraction

We intended to extract MFCC coefficient as our audio feature. However, there're some problems with the mpeg audio decoder. We have configured it for several days but it still failed to work. So, we had to drop audio in our work, in order to submit a result before deadline.

3.5 Result Analysis

The results are shown in figure 4, also disappointing. We're still looking for the reasons. It is probably because we did not set the search parameters of FLANN well. The parameters were calculated by training on the sampled feature set with algorithm provided by [7]. Maybe the sample rate is too small to represent the whole feature set, and the algorithm cannot work on the feature set well.

4. Conclusion

This report describes our content-based video copy detection system in TRECVID 2009. It is the first time of our participation in TrecVID, and the results are rather disappointing, however, the experiences we get are very useful. Because copy detection is a very important technique in video monitoring and copyright protecting, our lab will keep on working at this and hope we can do better next time.

5. Reference

[1] H Bay, A Ess, T Tuytelaars, L Van Gool. "Speeded-Up Robust Features (SURF)". In CVIU'2008.

[2] A Andoni, P Indyk. "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions". In Communications of the ACM, 2008.

[3] G Csurka, C Dance, L Fan, J Willamowski, C Bray. "Visual categorization with bags of keypoints". In ECCV'2004 Workshop on Statistical Learning in Computer Vision.

[4] H Bay, A Ess, T Tuytelaars, L Van Gool. "SURF: Speeded up robust features". In ECCV'2006.

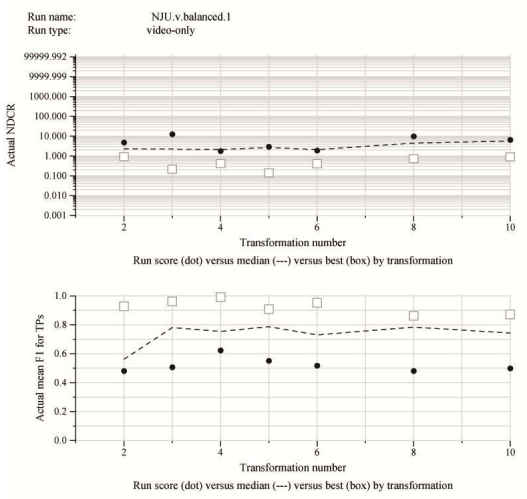
[5] Developed by OpenCV group. "http://opencv.willowgarage.com/wiki/".

[6] Developed by A Andoni et al, MIT. "http://www.mit.edu/~andoni/LSH/".

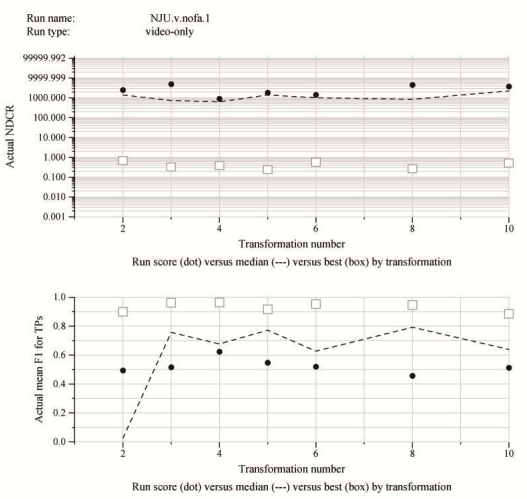
[7] M Muja, DG Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration". In VISAPP'2009.

[8] Developed by M Muja et al, UBC. "http://people.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN".

[9] Developed by INRIA-LEAR IMEDIA group, "http://lear.inrialpes.fr/people/douze/trecvid_generator/".

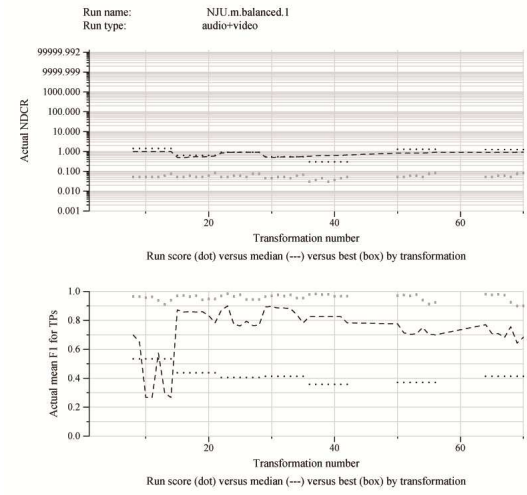


a) balanced

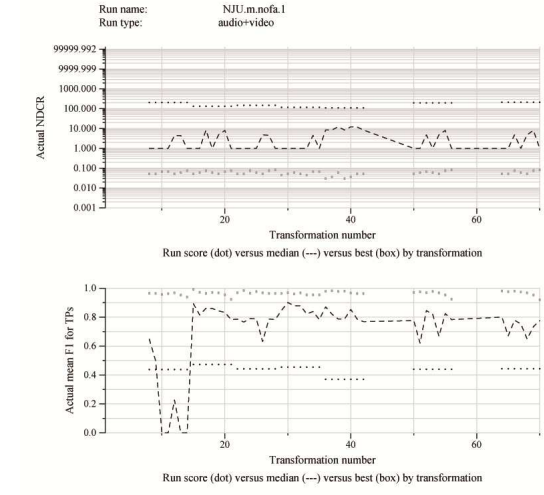


b) no false alarm

Figure 3.
Performance of our method in
video-only task



c) balanced



d) no false alarm

Figure 4.
Performance of our method in
video + audio task