National Institute of Informatics, Japan at TRECVID 2010

Duy-Dinh Le, Sebastien Poullot, Xiaomeng Wu, Bertrand Nouvel, Shin'ichi Satoh

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Japan 101-8430
{ledduy, poullot.sebastien, wxmeng, bertrand.nouvel, satoh}@nii.ac.jp

***Abstract*—This paper reports our experiments for three TRECVID 2010 tasks: instance search, semantic indexing, and content-based copy detection. For the instance search task, we present a simple approach that uses face-specific features for PERSON and CHARACTER queries and a combination of local and global features for the OBJECT and LOCATION queries. For the semantic indexing task, we report two approaches using frameworks KAORI-SECODE and PyCvF. For the content-based copy detection task, we use a baseline implementation of the audio fingerprinting and copy detection technique similar to that of last year. Our approaches can be considered as one of the baseline approaches for evaluation of these tasks.**

The hightlights: (1) We obtained the best result in the instance search task by using appropriate face-specific features. (2) Our best run using only global features (CM+LBP+EOH) achieved 0.0434 InfAP.

I. Instance Search

A. Face descriptors

We extracted frontal faces from queries and test keyframes (we extracted 50 keyframes/shot) using our frontal face detector [1]. For face descriptor, we evaluated 2 types: local-based descriptor and global-based descriptor.

The local descriptor [2] was formed by extracting pixel intensity around facial points. 9 facial feature points were detected, and 4 more facial feature points were inferred from these 9 points. In total, there were 13 feature points from which features are extracted. The features are intensity values lying within the circle with radius of 15 pixels. The output feature has 13x149 = 1,937 dimensions. The global descriptor was formed by extracting LBP feature in regions divided by a 5x5 grid of the face image.

We did internal experiments on the wild face dataset [3] and found that the local descriptor performed better than the global descriptor. Therefore we chose the local descriptor for face representation. As for similarity, L1 was used.

We computed the similarity between faces extracted from example images provided with the query and faces extracted from keyframes of the test set and then ranked shots by using the similarity scores.

We got the best performance with PERSON queries as shown in Figure 1. The results are shown in Table I.

B. Local descriptors

In this part an indexed system is used, inspired from [4].

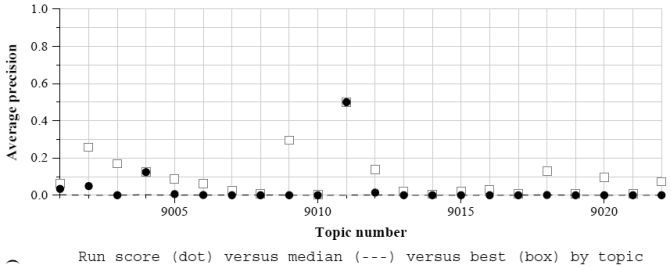

Fig. 1. Performance of our best run. We got high performance for PERSON queries.

Indexing the reference videos

First two databases were constructed, parameters are given right after the algorithm:

1) Extract keyframes (KF) from reference videos (about 1KF per second, depending on visual activity), then for each KF:
2) Extract the SIFT descriptors from keyframes (the vedaldi version[1] is used, on DoG positions),
3) Random project the SIFT descriptors in a 32 dimension space,
4) The projected SIFT descriptors are quantized on a N visual words vocabulary,
5) A KF is described in a BoF-like approach as a N bits vector, each i bit marking or not the presence of the $word_i$ or not in the KF,
6) Each SIFT descriptor position in the KF is associated to $M * l$ neighbors in the image plane to form M associations of $l + 1$ length,
7) An assocation define a hashing position depending on the quantizations of its projected SIFT descriptors,
8) An association also define a shape code: ratio between its longest side and shortest one,
9) The KF descriptor is inserted in a hash table at the positions described by all associations computed on it,
10) At each position the shape code of the corresponding association, the identifier of the video and the time code are also inserted.

Here, 2 vocabulary were used, N_1=2048 and N_2=16384

[1]http://www.vlfeat.org/ vedaldi/index.html

RunID	Description	MAP	Note
F_X_NO_NII.kaori_1	Face descriptors for PERSON and CHARACTER queries and CM+LBP+SIFT.HARHES for LOCATION and OBJECT queries	0.0330	Ranked 2nd
F_X_NO_NII.kaori_2	Face descriptors for PERSON and CHARACTER queries and CM+LBP for LOCATION and OBJECT queries	0.0331	Ranked 1st
F_X_NO_NII.per900d.seb_3	Auto 2 bases: 2 quantizations 2 associations Sift and RGB hist	0.0331	Buggy codes
F_X_NO_NII.per900d.seb_4	Auto 1 base: 1 quantization 1 associations Sift and RGB hist	0.0331	Buggy codes

TABLE I
THE PERFORMANCE OF NII'S RUNS FOR THE INS TASK.

RunID	Description	MAP	Note
F_A_nii.ksc.run_fuseall	Fusion of global features (CM+LBP+EOH) and local features (SIFT.HARHES+SIFT.HESAFF)	0.0485	Not submitted run - No bug
F_A_nii.ksc.run_global	Global features (CM+LBP+EOH) 5x5 grid and 3x3 grid, 3 classfiers/feature	0.0434	Not submitted run - No bug
F_C_nii.ksc.run1005_1	Fusion of local features (SIFT.HARHES and SIFT.HESAFF) and global features (CM, LBP, EOH) extracted from 3x3 and 5x5 grid - 11 classifiers	0.0151	Submitted run - Buggy codes
F_C_nii.ksc.run1002_2	Fusion of global features (CM, LBP, EOH) extracted from 3x3 grid - 3 classifiers	0.0144	Submitted run - Buggy codes
F_A_nii.ksc.run1001_3	Fusion of global features (CM, LBP, EOH) extracted from 3x3 and 5x5 grid - 6 classifiers	0.0142	Submitted run - Buggy codes
F_A_nii.PyCVF1_4	The run using PyCVF framework, features = CM +GIST, learner = 9-NN	0.0011	

TABLE II
THE PERFORMANCE OF NII'S RUNS FOR THE SIN TASK.

one. For the N_1 vocabulary, $l = 2$ so triplets are built, and $M = 5$. For the N_2 vocabulary, $l = 1$ so duets are built and $M = 5$.

Basically, about 300 SIFT descriptors are extracted in a KF, so a KF is indexed in 1500 buckets in one database.

The quantization is based on a K-means. We performed a K-means on the projected SIFT descriptors using $K = N_1$ or $K = N_2$. The SIFT descriptors are then quantized by the identifier of the nearest centroid using L_2 distance. 2 different reference databases are used, based on 2 different vocabularies.

Searching in the database

On search time, we performed the same operations on the images of a query QI_i for the two vocabularies:

1) Step 2 to 8 from the database construction are performed to obtain this image query descriptor ($QI_{i.qd}$), and its hashing positions along with their shape codes,
2) In each reference database, the hashing positions defined for a $QI_{i.qd}$ are checked as follows: for each element in these buckets, compare query association shape code and current element shape code, if similar enough then compute similarity S_{qd-c} between $QI_{i.qd}$ and current element descriptor.

Now for each image of each query QI_i 2 ranked sets of corresponding KF from the 2 databases are obtained, one for each vocabulary. For a given query, all lists of its images are simply fused all together.

Color histograms

In addition to the previous method, the colors are used for computing similarity. On each KF from the database, the global color histogram is computed. On each image of each query, the "IN mask" histogram and the "whole picture" histogram are computed. On search, after the steps previously presented , the similarity (L_1 normalized) Sh_{in-w} between "IN mask" histogram and "whole picture" histogram from KF, and Sh_{w-w} between "IN mask" from image query and "whole picture" histogram from KF are computed. They are merged to S_{qd-c} with the formula: $S = 0.6 \times S_{dq-c} + 0.35 \times Sh_{i-w} + 0.15 \times Sh_{w-w}$.

II. SEMANTIC INDEXING

A. Method Overview

1) The KAORI-SECODE Framework: In our framework, features are extracted from the input keyframes representing for shots. We extracted five keyframes per shot that are spaced out equally within the provided shot boundary. In the training stage, we use these features to learn SVM classifiers. These classifiers are then used to compute the raw output scores for the test image in the testing stage. These output scores can be further fused by taking the average for computing the final output score. In order to return K shots most relevant for one concept query that then are evaluated and compared in TRECVID benchmark, all normalized final output scores of shots are sorted in descending order and top K shots are returned. In the case of a shot consisting of several sub-shots, only the maximum score among subshots' scores is used for that shot.

As for feature extraction, we used both local and global features. The global features are color moments, edge direction histogram and local binary patterns. These features are extracted from a 5x5 grid and 3x3 grid of the input image, normalized to [0, 1]. The local features are SIFT descriptors extracted from keypoints detected by HARHES and HESAFF

keypoint detector. We used GreedyRSC+KMeans to find approximated 1,000 clusters for vector quantization. Then a standard bag-of-words was used to form the feature vector.

To handle the problem of unbalanced training data (in which most of training samples are negative), for each feature, we trained 3 classifiers using the same positive set and a subset of the negative set. For the negative subset, a maximum 20,000 sample was randomly selected. The prediction scores of these 3 classifiers were fused for the final score.

2) The PyCVF Software Framework: PyCVF is a young software framework that we are building in order to simplify video mining. By making things easier, we would like to see change what is achievable. PyCVF is young, and we do not had not enough workforce to dream about state-of-the-art performance, but we wanted to test it in TRECVID to evaluate the quality of the current specification of this framework.

PyCVF, in its 0.2 version, is available as a opensource software for download from Sourceforge, and from the Python Package Index. Instructions for installation are provided online [5]. In [6], we survey the essential commands of PyCVF 0.1.

On semantic indexing, our original plan was to compute some descriptors from local SIFTS descriptors. Since there was time limitations and since we faced difficulties to overpass some hardware bottleneck on our cluster infrastructure by the mean of reliable multicast, we could not compute the feature we wanted to have. So, we ended up implementing nearest-neighbors on a joint-descriptor based on color-moments and GIST. There was nothing really new, but it was the safest way to get one run, that will prove that the framework is getting ready to work. The color moments were computed up in RGB up to second order moments for an image subdivided in 4 blocks by 4 blocks. The GIST was computed with its default parameters. The weight of the color moments has not been optimized and the relative weight of GIST with respect to colormoments was not tuned, and therefore our run has low performance on this task. The approximative nearest neighbor-search structure used SASH [7].

Our participation to TRECVID this year has shown that the issues related to scalability and distributed computing were not yet refined enough in current PyCVF version. They are definitively to stand on the road-map for the next releases of the framework. PyCVF is open-source, we have not decided of the status of the TRECVID-related packages. It is likely that we would be interested in sharing some of these with the other teams that may be interested in PyCVF development.

B. Result

We submitted 4 runs (3 runs using KAORI-SECODE framework and 1 run using PyCVF framework) and the results are shown in Table II. Due to some buggy codes in keyframe extraction when handling video programs having highly different frame rates, the results of our 3 submitted runs using KAORI-SECODE were failed. After fixing the bugs, we re-run the experiments and the true performance was reported in Table II.

III. Audio + Video Copy Detection

We used a baseline implementation of the audio fingerprinting and copy detection technique as described in the paper [8], which has been demonstrated to be robust and reliable in the content-based copy detection task of the last two years. It focuses on the energies of 33 bark-scaled bands of each audio frame, and uses the sign of the energy band differences both in time and the frequency axis as a 32-bit fingerprint. The hashing technique is used for accelerating the copy detector. Figure 2 shows the results of audio-only copy detection based on this baseline.

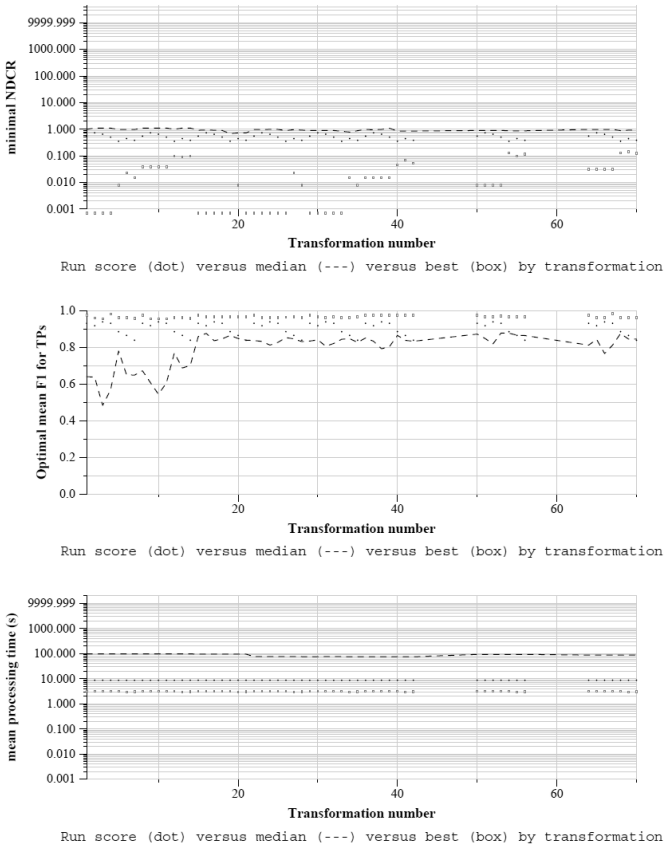

Fig. 2. Copy Detection Results (Balanced; Audio-Only).

The audio and video copy detector can be integrated at feature, fingerprint, or decision level. Because of the different feature representation strategies that we used for the two modalities, we chose an integration at decision level. Because the integration is based on the individual audio and video copy detector outputs, three choices determine the effectivity of the integration: 1) the operator used for integrating the sequence, 2) the operator used for integrating the frame, and 3) how to compute the joint confidence score. It has been demonstrated, by our comparison experimentation of last year, that a union operator is a better choice for the sequence integration. Meanwhile, the video-only copy detector normally achieves much lower copy location recall than the audio-only one, while the precision in both cases is similar. The

reason comes from the representative-frame-based strategy that the video-only copy detector used. A union operator is thus chosen for the frame integration to moderate the number of misses and keep the precision constant. For the fusion of the confidence score, we used the weighted average strategy. We assume that the confidence of the audio-only copy detector is higher than that of the video-only one considering the much more complicated transformations used for the video streams. A higher weight (0.65) is thus associated to the audio-only outputs. Figure 3 shows the results. Though slightly, the integration still improved the minimal NDCR of the copy detection accuracy.

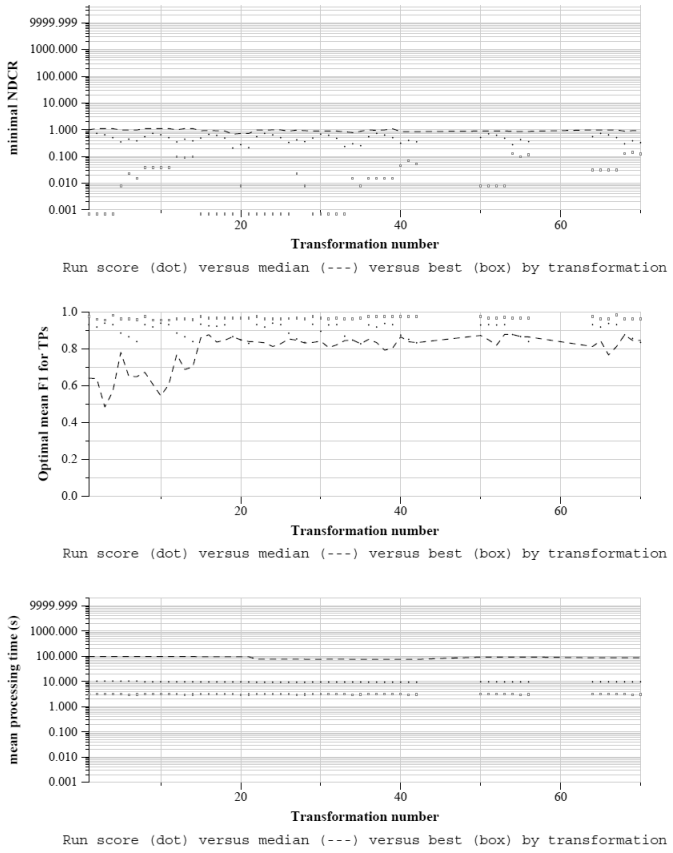

Fig. 3. Copy Detection Results (Balanced; Audio + Video).

REFERENCES

[1] D.-D. Le and S. Satoh, "Multi-stage approach to fast face detection," vol. 89, no. 7, Jul 2006, pp. 2275–2285.

[2] M. Everingham, J. Sivic, and A. Zisserman, "'Hello, My name is... Buffy' – automatic naming of charecters in tv video," in *Proc. British Machine Vison Conf.*, 2006.

[3] T. L. Berg, A. C. Berg, J. Edwards, M. Maire, R. White, Y. W. Teh, E. G. Learned-Miller, and D. A. Forsyth, "Names and faces in the news," in *Proc. Intl. Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 848–854.

[4] S. Poullot, M. Crucianu, and S. Satoh, "Indexing local configurations of features for scalable content-based video copy detection," in *LS-MMRM '09: Proceedings of the First ACM workshop on Large-scale multimedia retrieval and mining*, 2009.

[5] "PyCVF website," http://pycvf.sourceforge.net/. [Online]. Available: http://pycvf.sourceforge.net/

[6] B. Nouvel and S. Satoh, "The python computer vision framework," in *Proceedings of ACM MultiMedia 2010*, 2010.

[7] M. Houle and J. Sakuma, "Fast approximate similarity search in extremely high-dimensional data sets," 2005.

[8] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *3rd International Conference on Music Information Retrieval*, 2002.