National Institute of Informatics, Japan at TRECVID 2011

Duy-Dinh Le [1], Cai-Zhi Zhu [1], Sebastien Poullot [1], Vu Q. Lam [2]
Duc A. Duong [2], Shin'ichi Satoh [1]

[1] *National Institute of Informatics*
2-1-2 Hitotsubashi, Chiyoda-ku, Japan 101-8430

[2] *Faculty of Information Technology*
University of Science, VNU
227 Nguyen Van Cu, Dst 5, Ho Chi Minh City, Vietnam

ledduy,cai-zhizhu,poullot.sebastien,satoh@nii.ac.jp,
lqvu,daduc@fit.hcmus.edu.vn

***Abstract*—This paper reports our experiments for three TRECVID 2011 tasks: instance search, semantic indexing, and multimedia event detection. For the instance search task, we present three different approaches: *(i) Large vocabulary quantization by hierarchical k-means and weighted histogram intersection based ranking metric (ii) Combination of similarities based on Glocal quantization of two set of SIFTs and color histograms from the full frames, and (iii) Keypoint matching is used to compute the similarity between images of the query and images of all videos.* For the semantic indexing task and the multimedia event detection task, we report the experiments using NII-KAORI-SECODE framework. Our approaches can be considered as one of the baseline approaches for evaluation of these tasks.**

I. Instance Search

A. Method 1: Large Vocabulary Quantization by Hierarchical k-means and Weighted Histogram Intersection Based Ranking Metric

Here we present details about our algorithm for instance search task. Basically our algorithm searches for matching in a computationally cheaper high dimensional Bag-of-Word feature space. While being computationally cheaper and theoretically simple, it probably takes the lead of all submitted runs in this instance search task. The idea is inspired by four pioneer researches [1], [2], [3], [4], and thanks to the given object mask regions for all probe images, we can additionally consider to balance contributions originated from object regions and background context within a given probe image.

1) Offline Indexing: The framework of offline indexing is shown in Fig. 1. For the gallery dataset, we extract 3 frames per second from every video clips (100 frames per clip in average), and then SIFT descriptors are sparsely extracted. Throughout this experiment, the only feature we used is SIFT. More concretely, we use the SIFT library available in [5], and use Harris-Laplace and MSER two detectors for complementary coverage. Typically for each frame with size 352*288, we get 800 and 300 key points in average via above two detectors separately. Then for those detected key points, we exact 128-D SIFT or 192-D color SIFT descriptors [5]. In this way, we collect 2.3E9 SIFT and 2.3E9 color SIFT descriptors for 20982 video clips in total.

Next, we evenly sample 71M descriptors for clustering. For speed consideration, we build a vocabulary tree with 1M leaf nodes as done in [2], but try out different branching factor 10 and 100, with the tree level factor equal to 6 and 3 correspondingly. We do realize the shortcoming of hierarchical clustering in incurring larger quantization error, and the reason why we still chose hierarchical method is just to make the experiment doable within reasonable time. We believe our algorithm will benefit from a more accurate clustering algorithm suitable for such a large scale dataset problem.

Then for each video unit, we project all SIFT descriptors into the vocabulary tree and get only one Bag-of-Words histogram as its representation, thus we totally get 20982 video histograms. Since we consider all nodes in the tree besides the root, each histogram consists of 111110 or 10100 bins, while the branching factor is set to 10 or 100. Finally we try different bin-weighted schemes on this high-dimensional histogram: simply timing each bin by the number of nodes within that level in the vocabulary tree, or adopting an IDF weighting. Thereby, after a high-dimensional bin-weighted histogram is obtained for each video clip, we finish the whole offline indexing process.

2) Online Searching: The framework of online searching is shown in Fig. 2. As for online retrieval, given the probe image set of a topic, we first sparsely extract SIFT features as what we have done for the gallery dataset. Specially, to obtain better coverage for small object, we also densely extract SIFT features from object mask regions, and finally we project all these SIFT features to the vocabulary tree

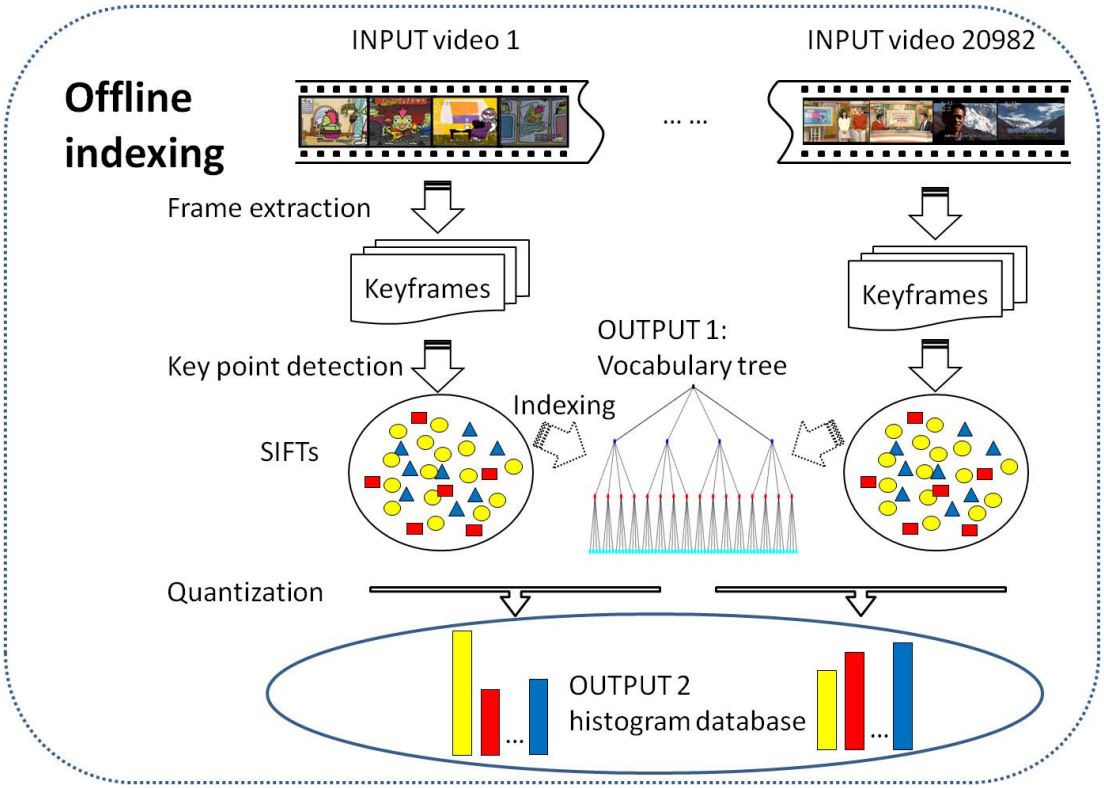

Fig. 1. Framework of offline indexing.

and get one histogram as the representation for current probe topic. Histogram intersection metric is then taken to rank the similarity between each probe topic with every candidate video clip.

3) Results: We choose MATLAB as our implementation platform. According to the framework of the algorithm, the computational burden mainly lies in building the big vocabulary tree during the offline phase. For the online retrieval, both quantizing features with the built tree, and ranking by computing histogram intersection over sole very sparse high dimensional histogram representation for each video/topic unit, can be very fast. The most computationally heavy part usually involved in other retrieval algorithms, spatial geometry verification, can be avoided in this framework thanks to the considerably fine quantization [2]. In our experiments, the average online retrieval time needed for a probe topic is around 15 minutes with our unoptimized MATLAB implementation.

We submitted two runs with this method: the first run uses IDF weighted histogram accumulated on color SIFT with tree branching factor equals to 100, and the other is a simple fusion over all considered configurations, such as: SIFT or color SIFT, branching factor 10 or 100, and level or IDF weighting. Overall the former one shown in Fig.1 performs better, which acquired optimal performance on 11 topics and nearly optimal performance on other 8 topics (where solid circles intersect with corresponding hollow squares in Fig. I-A3) among all 25 topics, with MAP equal to 0.531 and precision of top-10 shots returned around 90%, which is rather high in term of retrieval accuracy. The fusion one also ranks top on 7 topics with MAP 0.491. In total this method ranks top on 17 topics. We didn't carefully explore impact of different configurations due to lack of time and no ground truth available in the meantime, and we believe there is still margin to get the performance improved.

Figure I-A3 and I-A3 show the performance of these runs.

B. Method 2: Using Set of Features and Color Histograms at Frame Level

For this run we used full frame descriptors: Glocal descriptors (set of features, built on SIFT) and color histograms (HSV and RGB). The frames were densely extracted from the reference video set. On search time the same features are extracted from the sub queries full images, then sequentially compared to the database ones. The purpose of the approach is to obtain reasonable results in a very short time.

1) Build up the Video Reference Database: !

Local Descriptors

First, from the 20,982 reference videos, 1 frame out of 10 (about 2.5 frames per second) is extracted, 1,452,803 frames overall. Each frame F_i has a unique metadata set $\{ID, Tc\}$, where ID is the video ID and Tc is the frame number in this video. Then, for each frame 2 sets of SIFT descriptors are computed.

One set S_1 was computed at positions given by the fast multi-scale Hessian descriptor. The other set S_2 was computed at dense positions. The dense positions are extracted as follows. The frame is divided in a grid by a regular

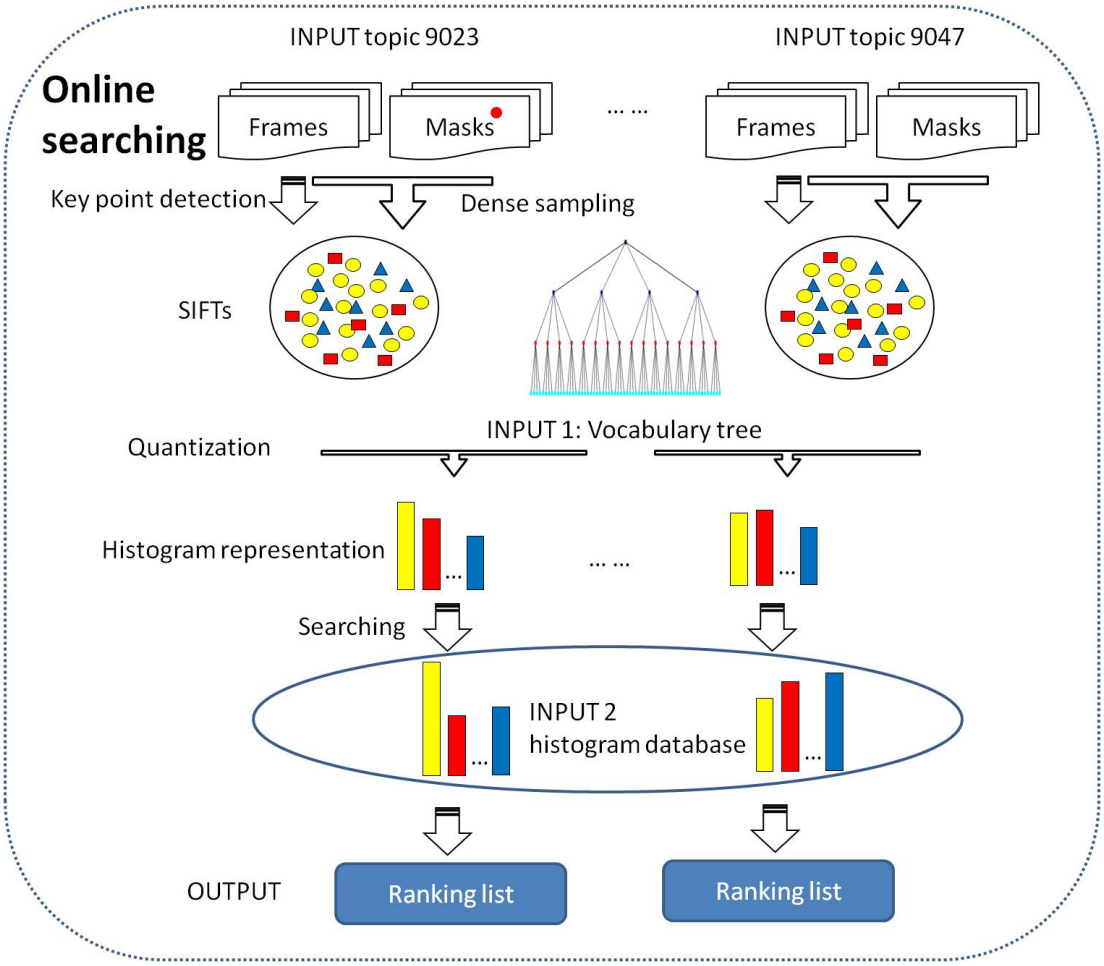

Fig. 2. Framework of online searching.

cut, in this case every $p = 7$ pixels. Then, on every patch of $7 * 7$ pixel, one position is selected, this position corresponds to the pixel with the higher luminance $L = \sqrt{R^2 * 0.241 + G^2 * 0.691 + B^2 * 0.068}$, where R, G and B are the RGB components of the pixel.

These 2 sets of SIFT are separately treated in the following.

Visual Vocabulary

For each set S_1 and S_2, a visual vocabulary is build up:

- 2 random matrices ($128*32$ dimensions) are drawn (Rm_1 and Rm_2) in order to project the SIFT on 32 dimensions features,
- 2 subset of 500,000 SIFT are randomly drawn,
- 2 visual vocabularies (V_1 and V_2) are build up using the K-means algorithm on each subset,
- the lengths of the 2 vocabularies are set to 2048 words, $|V_1| = |V_2| = 2048$,
- in order to go fast, the K-means processes are stopped after 5 iterations.

For each frame F_i, 2 Glocal descriptors G_{1i} and G_{2i} are computed, using S_{1i}, V_1 and S_{2i}, V_2. One descriptor is 2048 bits long (256 bytes). The bit at rank j of G_{1i} is set to 1 if the frame i contains at least one occurrence of a projected SIFT quantified on j-th word by vocabulary V_1. To quantify a projected SIFT we use the $1 - NN$ among the $|V_1|$ words. This description is a set of features (similar to BoW, but without counting of the occurrences).

Histogram Extraction

Beside the 2 glocal descriptors, the HSV and RGB color histograms, HSV_i, RGB_i are computed as well. The HSV histogram is quantized on $3 * q_{HSV}$ values, the RGB histogram is quantized on $3 * q_{RGB}$ values. The values were set to $q_{RGB} = 64$, $q_{HSV} = 16$.

Build Database Finally a frame F_i is described by a quadruplet $B_i = \{G_{1i}, G_{2i}, HSV_i, RGB_i\}$. The database is composed of 4 files. One file contains all G_1 descriptors, one file contains all G_2 descriptors, one file contains all HSV histograms, the last one contains all RGB histograms. No indexing method was used. Overall the database sizes 1.4Gb (366Mb for each Glocal file, 134Mb for the HSV one, 533Mb for the RGB one).

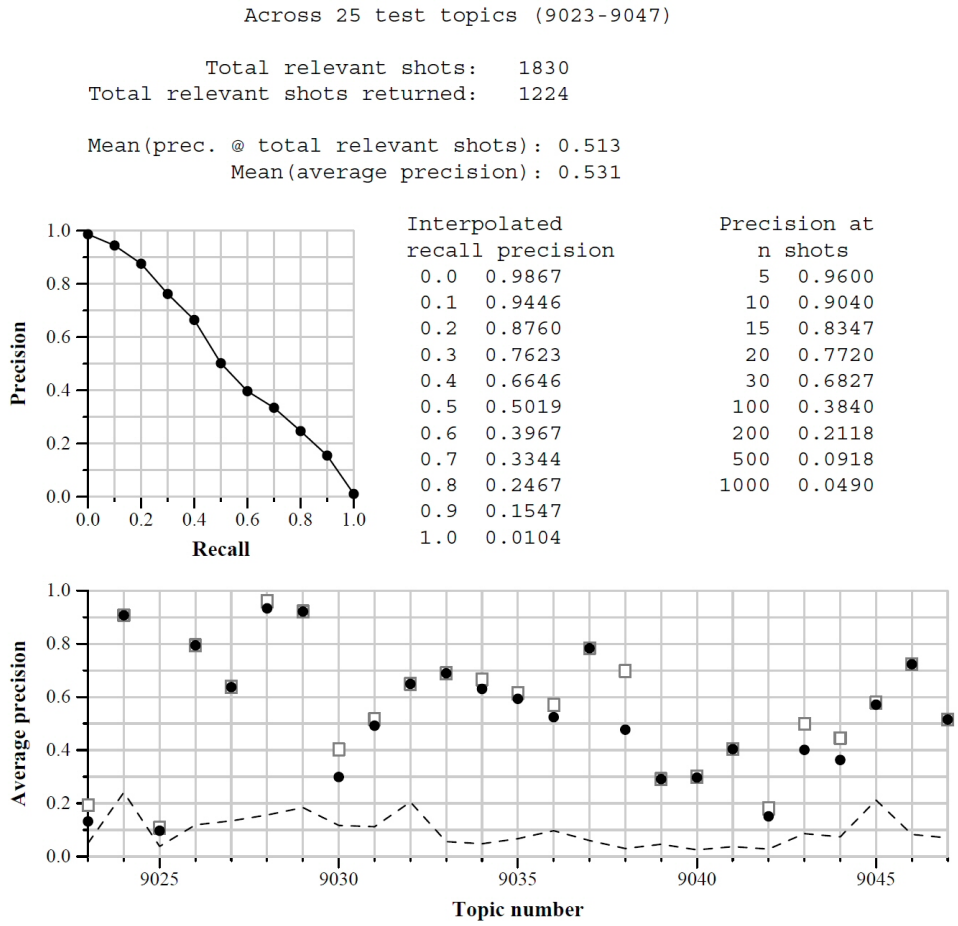

Fig. 3. Performance of the run R1 that is our best run.

2) Sequential Similarity Search: For this process, 23 queries Q_i were given. Each sub query Q_{ij} is also described by a quadruplet $S_{ij} = \{G_{1ij}, G_{2ij}, HSV_{ij}, RGB_{ij}\}$. Each quadruplet S_{ij} is then sequentially compared to each quadruplet B_i from the database. Two different type of similarity are used for this purpose. For the Glocal descriptors, the Dice coefficient is used. For instance between two Glocal descriptors G_a and G_b:

$$S_{\text{Dice}}(\mathbf{G}_a, \mathbf{G}_b) = \frac{2\,|\mathcal{G}_a \cap \mathcal{G}_b|}{|\mathcal{G}_a| + |\mathcal{G}_b|} \tag{1}$$

For the HSV and RGB histograms, a simple L_1 distance is computed. For instance between two histograms HSV_a and HSV_b:

$$S_{\text{L1}}(\mathbf{HSV}_a, \mathbf{HSV}_b)) = \sum_{i=0}^{3}\sum_{j=0}^{q} HSV_a[i,j] - HSV_b[i,j] \tag{2}$$

where q is the quantization of the histograms.

The global similarity Sim between a sub-query Q_{ij} and a frame from the database F_k was designed as follows:

$$\begin{aligned} Sim(\mathbf{S_{ij}}, \mathbf{B_k}) &= \frac{1}{2} * S_{\text{Dice}}(G_{1ij}, G_{1k}) \\ &+ \frac{1}{2} * S_{\text{Dice}}(G_{2ij}, G_{2k}) \\ &+ \frac{1}{4} * \frac{1}{S_{\text{L1}}(HSV_{ij}, HSV_k)} \\ &+ \frac{1}{4} * \frac{1}{S_{\text{L1}}(RGB_{ij}, RGB_k)} \end{aligned} \tag{3}$$

While performing search, the $Dice$ coefficients are first computed, if one of these is below a threshold $Th = 0.1$ the frame F_k is discarded. Otherwise the color histogram similarities are computed. In other words, a minimum of matching SIFT is required.

Given that each single similarity belongs to a $[0; 1]$ interval, and given this threshold, $Sim(S_{ij}, B_k) \in [0, 2; 1, 375]$. For each sub-query the results are descending sorted.

3) Merge Sub-queries Results: In order to merge the results from the N sub-queries, we normalize the similarities. The higher similarity of each query is set to 1.0, and the following ones modified according to this. If a query has N sub-queries, at least N similarities are $Sim = 1.0$ (if the threshold test was successfully passed at least once for each sub-query).

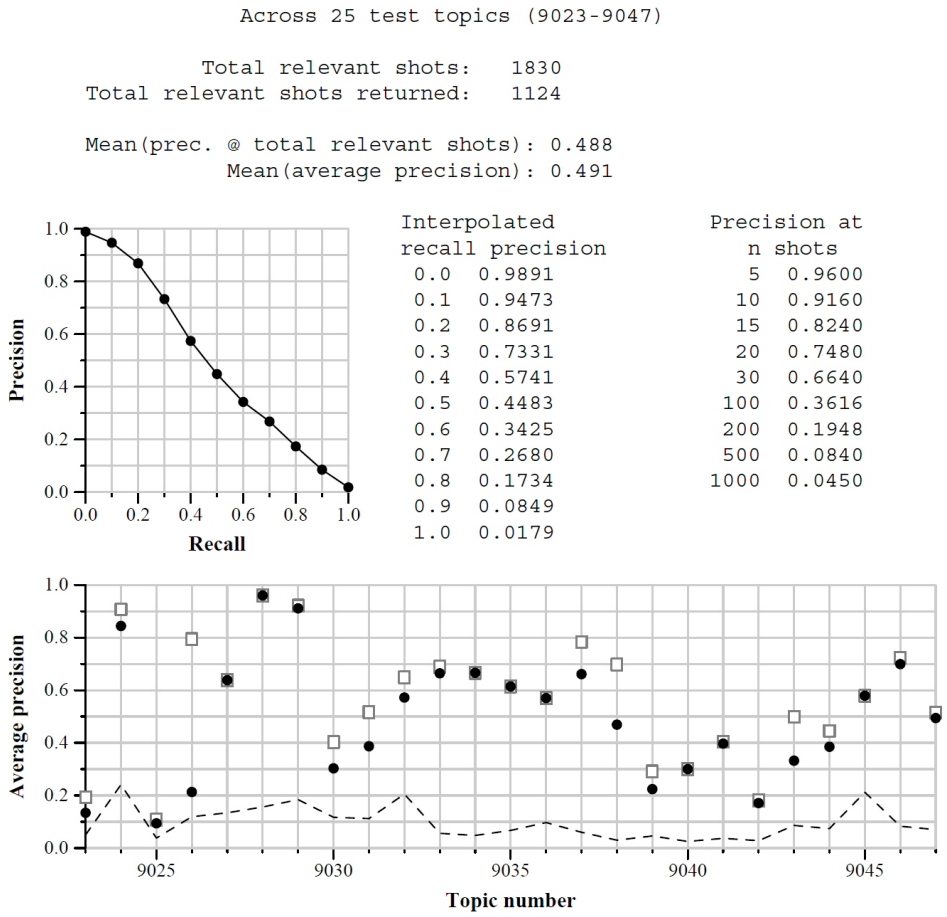

Fig. 4. Performance of the run R2

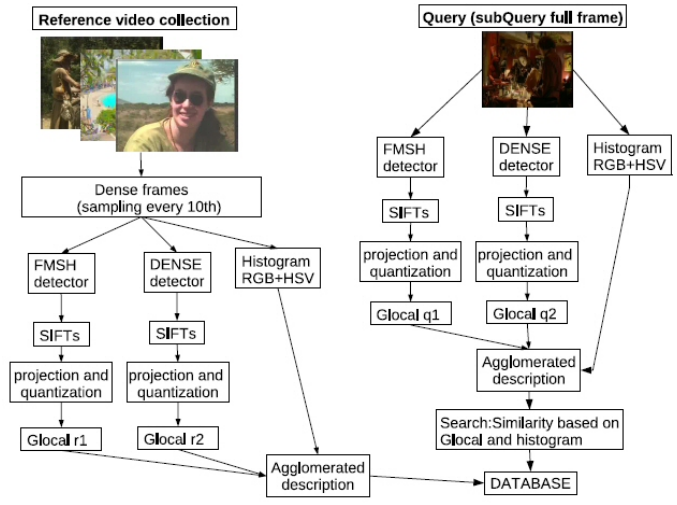

Fig. 5. Sum up of the database construction and search process from a unique sub query

Once normalization are done, the results of all sub-queries are concatenated and descending resorted according to the similarities. If 2 similarities are equal, we make a sub-comparison between the *Dice* coefficients sum of the 2 results. The one with the higher sum will be ranked first.

Figure I-B3 shows the performance of this run.

C. Method 3: Using KeyPoint Matching Between Query Images and KeyFrames Extracted from Test Videos

We extracted 5 keyframes per second in test videos. The total number of keyframes after extracting 20,982 test video clips is 2,657,073. For each keyframe of query images and test images, denseSIFT with sampling step of 6 pixels is used to extract 5,244 SIFT descriptors. The total number of descriptors is (2,657,073 + 95)$\times$ 5,244 = 13,934,188,992. For each keypoint in query images, we find 4 nearest keypoints in all keypoints of the test database. The similarity between the input query and one test video is computed by counting the number of matches between query images and keyframes of the test video. Specifically,

$$d(Q,V) = \sum_{i,j} w_k * MatchCount(q_i, v_j, k)$$

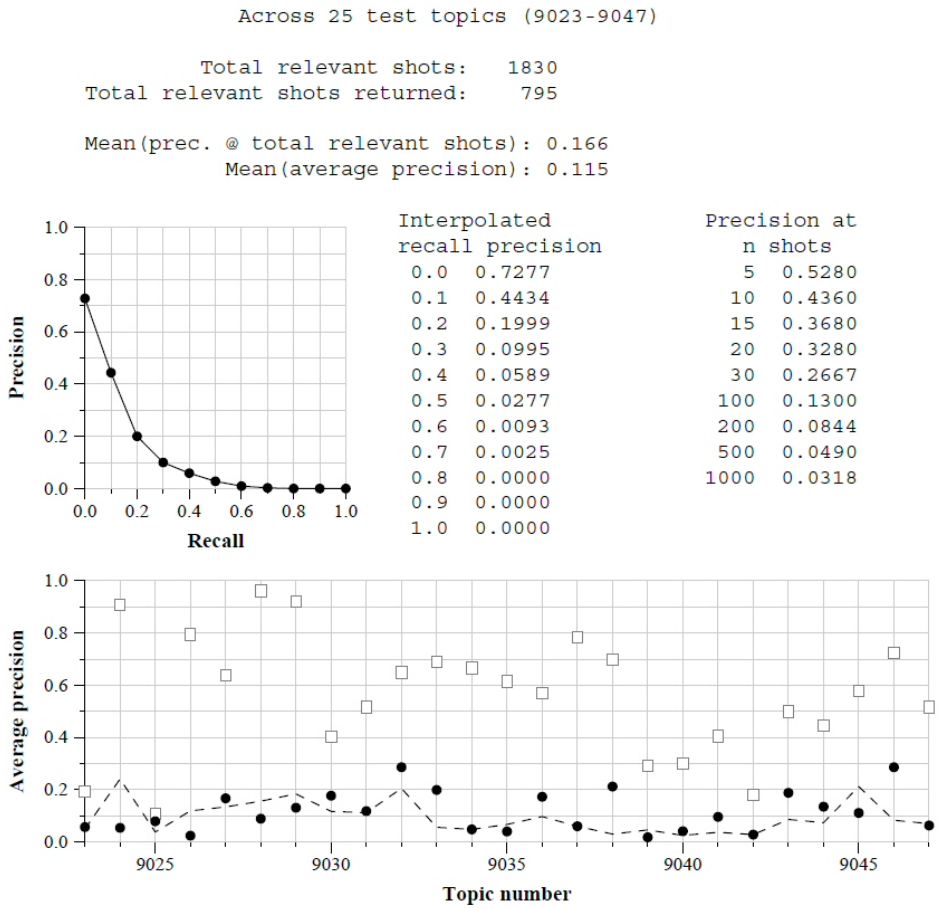

Fig. 6. Performance of the run using set of features and color histograms at frame level

where q_i is a query image of the query Q, v_i is a keyframe image of the test video V, $k \in [1..4]$ is the order of the nearest neighbor. $MatchCount(q_i, v_j, k)$ returns the number of matches of k_th nearest neighbor between q_i and v_j. $w_k = \frac{1}{2^{k-1}}$ is the weight for the match of k_th nearest neighbor. The highest weight is given for the 1-NN.

Since the number of keypoints in the test database is huge, the matching speed is very slow.

Figure I-C shows the performance of this run.

II. The NII-KAORI-SECODE Framework

A. Method Overview

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, dense sampling is used for finding keypoints from which SIFT and COLORSIFT descriptors are extracted. We used GreedyRSC+KMeans to find 500 clusters for vector quantization. Then a standard bag-of-words with soft-weighting was used to form the feature vector.

B. Semantic Indexing

The performance of runs with different configurations is available online at: http://satoh-lab.ex.nii.ac.jp/users/ledduy/Demo-KAORI-SECODE/. Due to time limitation, we could only submitted 3 runs and their performances are shown in Table I:

C. Multimedia Event Detection

We extracted one keyframe for every 4 seconds. The total number of keyframes for MED11TEST is 877,310. The total number of keyframes for EVENTS is 90,449. As for features, BOW of SIFT descriptors extracted by dense sampling at 2 scales with sampling step of 6 pixels (code of COLORSIFT [6] is used). 1.5 millions of keypoints from the training set (keyframes extracted from all 15 EVENTS) were used to

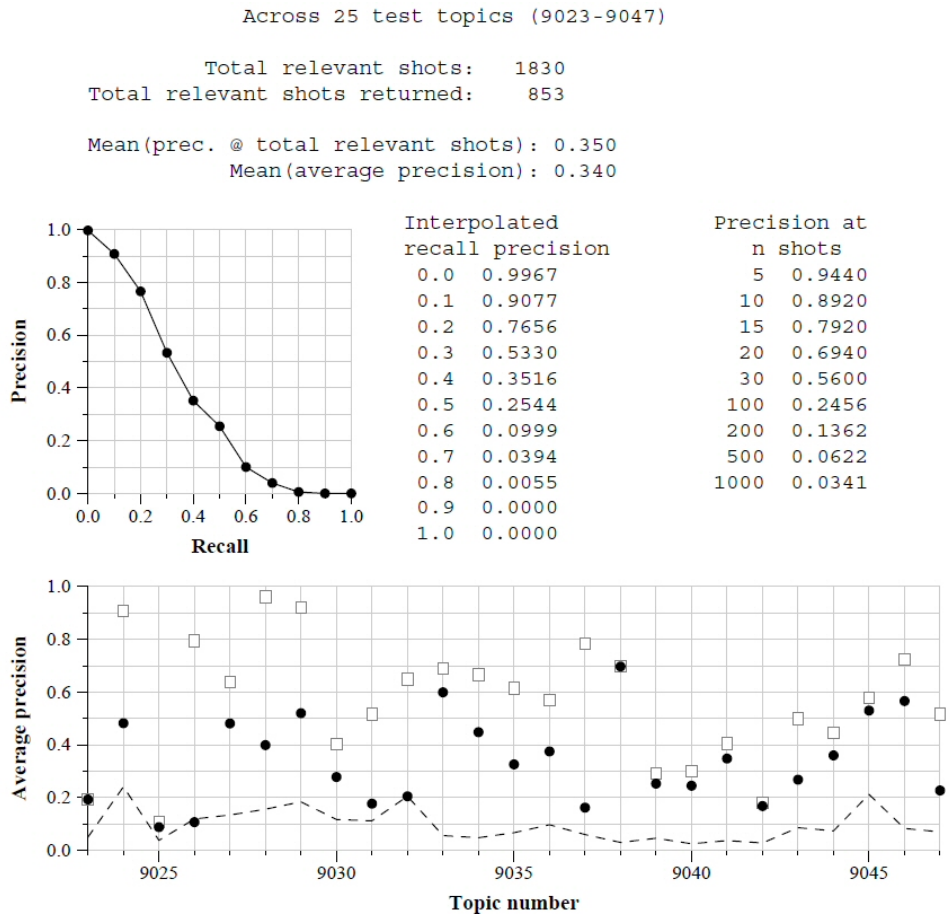

Fig. 7. Performance of the run using keypoint matching

TABLE I
PERFORMANCE OF OUR SUBMITTED RUNS FOR SIN TASK.

RunID	Description	MAP (%)
F-A-nii.SuperCat-dense6-1	DENSE6 - grid 3x1	11.3
F-A-nii.SuperCat-dense6mul.rgb-2	DENSEMUL6.rgbSIFT - grid 3x1	10.9
F-A-nii.SuperCat-dense6-3	PHOW8 - grid 3x1	11.9

construct a codebook of 500 codewords. Soft assignment is used for constructing feature vectors. 3 spatial layouts are used including 1x1, 2x2 and 3x1 (colxrow)

Figure II-C shows the performance of our primary system using dense color.

REFERENCES

[1] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *CVPR*, 2008.
[2] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR*, 2006.
[3] K. Grauman and T. Darrell, "Approximate correspondences in high dimensions," in *NIPS*, 2006.
[4] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *ICCV*, 2003.
[5] "Featurespace," http://www.featurespace.org/.
[6] "Color descriptors," http://koen.me/research/colordescriptors/.

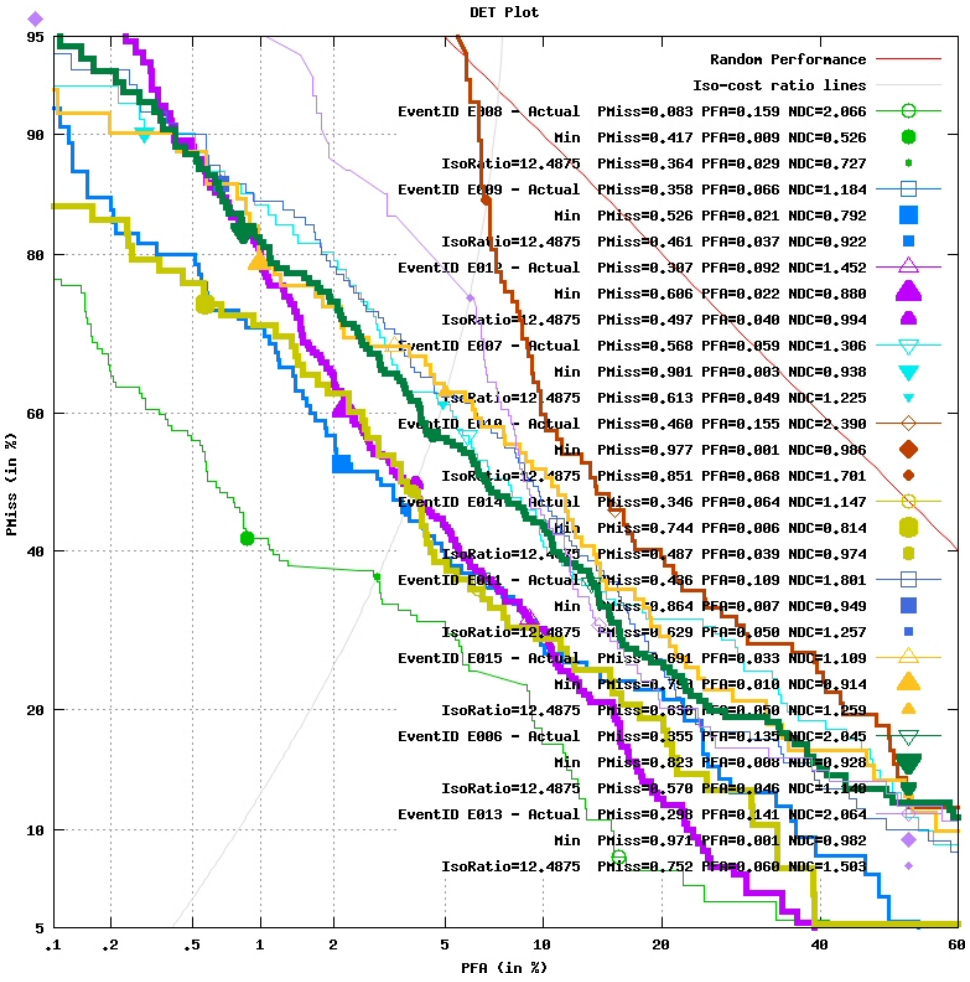

Fig. 8. Performance of our primary system for MED task