TokyoTech-Waseda at TRECVID 2014

Nakamasa Inoue, Zhuolin Liang, Mengxi Lin,
Tran Hai Dang, Koichi Shinoda
Tokyo Institute of Technology.
{inoue, zhuolin, mengxi, dang}@ks.cs.titech.ac.jp
shinoda@cs.titech.ac.jp

Zhang Xuefeng, Kazuya Ueki
Waseda University.
yuki@pcl.cs.waseda.ac.jp,
ueki18@aoni.waseda.jp

1 Semantic Indexing

We aim at developing a high-performance semantic indexing system using Deep Convolutional Neural Networks (CNNs) and Gaussian mixture model (GMM) supervectors [7, 8, 9]. This year, we introduced Deep CNNs pre-trained on the ImageNET dataset to our system at TRECVID 2013, which uses GMM supervectors corresponding to six types of audio and visual features. Our best result was 28.1% in terms of Mean InfAP, which was ranked third among participating teams in the semantic indexing task. Note that the Deep CNNs improved Mean InfAP from 26.0% to 28.1%.

1.1 Deep Convolutional Neural Networks

We use deep Convolutional Neural Network (CNN) trained on the ImageNET LSVRC 2012 dataset to extract features from video shots. A 4096-dimensional feature vector is extracted from the key-frame of each video shot by using the CNN. As shown in Figure 1, the CNN has seven layers as proposed in [39]. The first to fifth layers are convolutional layers, in which the first, second, and fifth layers have max-pooling procedure. The sixth and seventh layers are fully connected. The parameters of the CNN is trained on the ImageNET LSVRC 2012 dataset with 1,000 object categories. Finally, from each key-frame, we extract a 4096-dimensional feature at the sixth layer to train an SVM for each concept in the Semantic Indexing task.

1.2 GMM Supervectors

1.2.1 Low-Level Feature Extraction

The following six types of visual and audio features are extracted from video data.

1. SIFT features with Harris-Affine detector (SIFT-Har)

   Scale Invariant Feature Transform (SIFT) proposed by Lowe [10] is a local feature extraction method that is widely used for object categorization since it is invariant to image scaling and changing illumination. The Harris-Affine detector [11], which is an extension of the Harris corner detector, improves the robustness against affine transform of local regions. SIFT features are extracted from every other frame, and principal component analysis (PCA) is applied to reduce their dimensions from 128 to 32.

2. SIFT features with Hessian-Affine detector (SIFT-Hes)

   SIFT features are extracted with the Hessian-Affine detector [11], which is complementary to the Harris-Affine detector. The combination of several different detectors can improve the robustness against noise. SIFT features are extracted from every other frame, and PCA is applied to reduce their dimensions from 128 to 32.

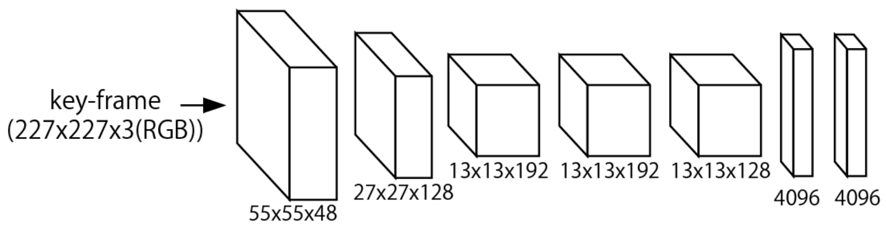

Figure 1: Deep Convolutional Neural Network

3. SIFT and hue histogram with dense sampling (SIFTH-Dense)

   SIFT features and 36-dimensional hue histograms [12] are combined to capture color information. SIFT+Hue features are extracted from key-frames by using dense sampling (100x100 grid with 3 scales). PCA is applied to reduce dimensions from 164 to 32.

4. HOG with dense sampling (HOG-Dense)

   32-dimensional histogram of oriented gradients (HOG) are extracted from up to 100 frames per shot by using dense sampling with 2x2 blocks. PCA is applied but dimensions of the HOG features are kept to 32.

5. LBP with dense sampling (LBP-Dense)

   Local Binary Patterns (LBPs) [13] are extracted from up to 100 frames per shot by using dense sampling with 2x2 blocks to capture texture information. We follow the procedure in [13] to extract LBP features. PCA is applied to reduce dimensions from 228 to 32.

6. MFCC audio features (MFCC)

   Mel-frequency cepstral coefficients (MFCCs), which describe the short-time spectral shape of audio frames, are extracted to capture audio information. MFCCs are widely used not only for speech recognition but also for generic audio classification. Δ MFCCs, $\Delta\Delta$ MFCCs, Δ log-power and $\Delta\Delta$ log-power are extracted in addition to the MFCCs. Here, "Δ" means the derivation of the feature. The dimension of the audio feature is 38, including 12-dimensional MFCCs.

1.2.2 Gaussian Mixture Models

Gaussian mixture models (GMMs), whose probability density function (pdf) is given by

$$p(x|\theta) = \sum_{k=1}^{K} w_k \mathcal{N}(x|\mu_k, \Sigma_k), \tag{1}$$

are used to model video shots. Here, x is a local feature, $\theta = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ is a set of GMM parameters, K is the number of Gaussian components (vocabulary size), w_k is a mixture coefficient, and $\mathcal{N}(x|\mu_k, \Sigma_k)$ is a Gaussian pdf with a mean vector μ_k and a covariance matrix Σ_k.

The GMM parameters are estimated for each shot under the maximum a posteriori (MAP) criterion. The MAP solution for GMM means, namely MAP adaptation, is given by

$$\hat{\mu}_k = \frac{\tau \hat{\mu}_k^{(\mathrm{U})} + \sum_{i=1}^{n} c_{ik} x_i}{\tau + C_k}, \quad c_{ik} = \frac{w_k g_k(x_i)}{\sum_{k=1}^{K} w_k g_k(x_i)}, \quad C_k = \sum_{i=1}^{n} c_{ik}, \quad g_k(x) = \mathcal{N}(x|\hat{\mu}_k^{(\mathrm{U})}, \hat{\Sigma}_k^{(\mathrm{U})}), \tag{2}$$

where $X_{\mathrm{F}} = \{x_i\}_{i=1}^{n}$ (F $\in$ {SIFT-Har, SIFT-Hes, SIFTH-Dense, HOG-Dense, LBP-Dense, MFCC}) is a set of feature vectors extracted from a shot, τ is a predefined hyper-parameter, and $\hat{\theta}^{(\mathrm{U})}$ is the parameter for a universal background model (UBM). The UBM presents how the features are distributed in the general case: therefore, the parameter $\hat{\theta}^{(\mathrm{U})}$ is estimated by using all features in the training set.

1.2.3 Fast MAP Adaptation

The fast MAP adaptation technique [7, 8] reduces computational costs for calculating posterior probabilities c_{ik} in Eq. (2) by constructing a tree-structured GMM. The basic idea of the tree-structured GMM is to cluster Gaussian components and approximate them with a single Gaussian. Each leaf node corresponds to a Gaussian component of the UBM, and each non-leaf node has a single Gaussian that approximates its descendant Gaussian components. See [7, 8] for details.

1.2.4 GMM Supervector Representation

After video shots are represented by GMMs, GMM supervectors are extracted by combining normalized mean vectors as

$$\phi(X_{\mathrm{F}}) = \begin{pmatrix} \tilde{\mu}_1 \\ \tilde{\mu}_2 \\ \vdots \\ \tilde{\mu}_K \end{pmatrix}, \quad \tilde{\mu}_k = \sqrt{w_k^{(U)}} (\Sigma_k^{(U)})^{-\frac{1}{2}} \hat{\mu}_k. \tag{3}$$

1.3 Late Fusion

Support vector machines (SVMs) with the following RBF-kernel are used to train discriminative models for each semantic concepts.

$$k(X_{\text{F}}, X'_{\text{F}}) = \exp\left(-\gamma \|\phi(X_{\text{F}}) - \phi(X'_{\text{F}})\|_2^2\right), \quad \gamma = \frac{1}{\tilde{d}}, \tag{4}$$

where $\tilde{d}$ is the average distance between two GMM supervectors or two features from the CNN. Here, annotations are obtained from the collaborative annotations [4]. Finally, trained discriminative functions are linearly combined as

$$f(X) = \sum_{\text{F} \in \mathcal{F}} \alpha_{\text{F}} f_{\text{F}}(X_{\text{F}}), \quad 0 \le \alpha_{\text{F}} \le 1, \quad \sum_{\text{F}} \alpha_{\text{F}} = 1. \tag{5}$$

where $\mathcal{F} = \{\text{CNN}, \text{SIFT-Har}, \text{SIFT-Hes}, \text{SIFTH-Dense}, \text{HOG-Dense}, \text{LBP-Dense}, \text{MFCC}\}$. Combination coefficients α_{F} are optimized on a validation set.

1.4 Video-Clip Scores

The relationship between shots are useful for detecting semantic concepts. For example, Safadi et al. [14] proposes a re-ranking method to re-evaluate scores of video shots by using shot-score distributions. In our re-ranking method, we define a video-clip score as the maximum value of shot scores among all the shots in a video clip:

$$s_{\text{max}} = \max_i s_i \tag{6}$$

where $s_i (i = 1, 2, \cdots, n)$ are shot scores for a video-clip that consists of n shots. Our final score for ranking shots is given by

$$s'_i = (1 - p)s_i + p s_{\text{max}} \tag{7}$$

where p is a probability of appearance of a semantic concept in a video clip given by

$$p = r \left\langle \frac{\#(\text{positive shots in a video clip})}{\#(\text{shots in a video clip})} \right\rangle. \tag{8}$$

where r is a scaling parameter. The final score s'_i gets closer to s_{max} as the concept appear more often (e.g. an anchorperson in a news video). It gets closer to the original shot score s_i for concepts appear in few times (e.g. a bus in a street video).

1.5 Experimental Conditions

Mikolajczyk's implementation [11] was used to extract SIFT-Har and SIFT-Hes features. SIFT++ [15] was used to extract SIFTH-Dense features. HTK [16] (speech recognition toolkit) was used to extract MFCC. The sum of calculation time (for PCA projection, MAP adaptation, and SVM prediction) was reduced from 2.47 sec to 1.00 sec (59.5%) by using the fast MAP adaptation technique. The calculation time was measured by using a Intel Xeon 2.93 GHz CPU. The Caffe toolkit [40] is used to extract feature vectors from a CNN.

The following four runs are submitted to the TRECVID 2014 semantic indexing [1, 2, 3].

TokyoTech-Waseda_4

This run used GMM supervector SVMs with the six types of features described in Section 1.2.1. The number of Gaussian components was $K = 512$ for the visual features, and $K = 256$ for the audio feature. The UBMs were trained using 1,000,000 samples. Optimal tree structures for the UBMs were selected as to minimize computational costs for MAP adaptation (see [7, 8]). The hyper-parameter τ for MAP adaptation was set to 20.0. Weights α_{F} for fusion are optimized on IACC_1_B dataset. The scaling parameter r for video-clip scores is set to 0.9.

TokyoTech-Waseda_3

This run uses the same features as the baseline method. Training samples are randomly selected three times for each feature, and all detection scores are averaged.

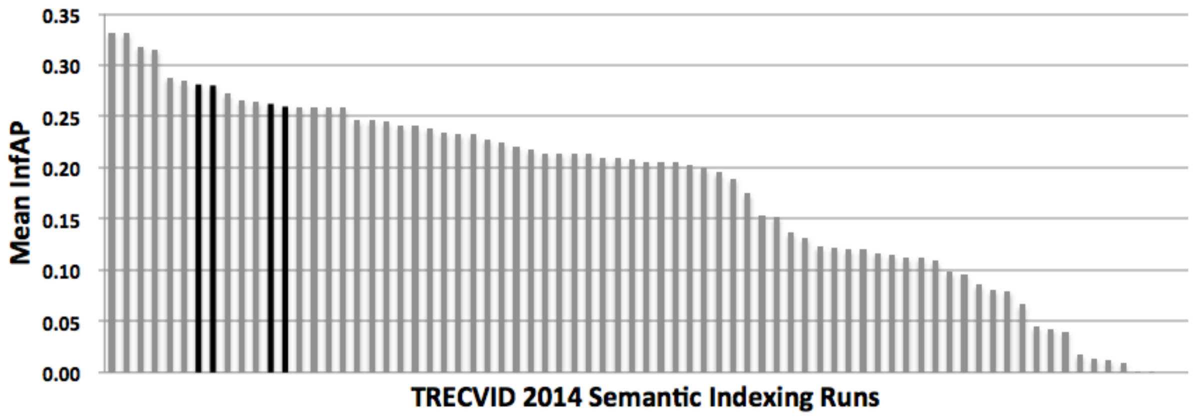

Figure 2: Overview of results of the semantic indexing task in TRECVID 2014. Our best result was Mean InfAP of 28.1%. Our four runs are colored in black.

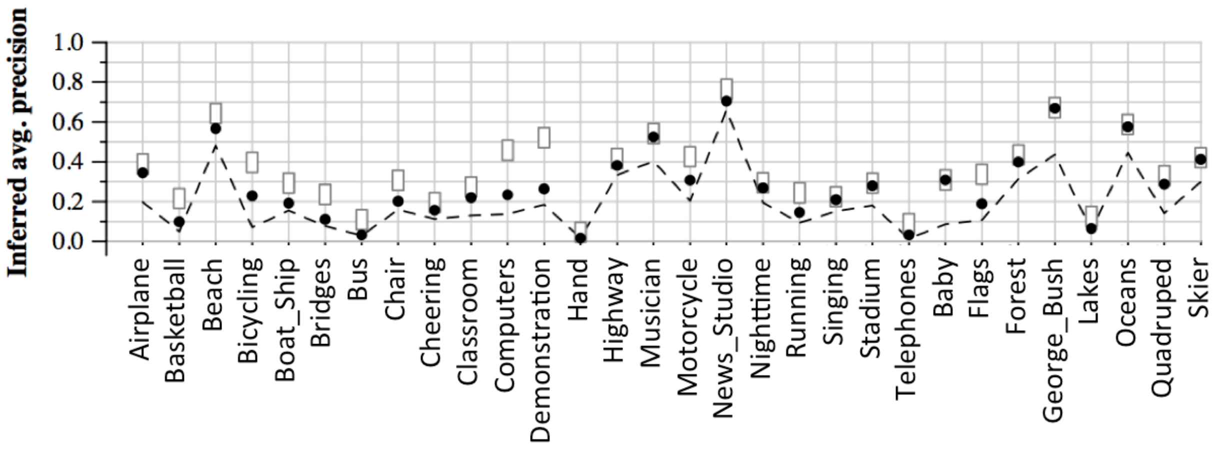

Figure 3: InfAP by semantic concept.

TokyoTech-Waseda_2

This run introduces Deep CNNs to the baseline method. Late fusion, in which detection scores from Deep CNNs and the baseline system are averaged, is used for this run.

TokyoTech-Waseda_1

This run optimizes a weight in late fusion on the TRECVID 2013 dataset.

1.6 Results

Figure 2 shows the overview of results of the semantic indexing task. Our best result by the run of TokyoTech-Waseda_1 was 28.1% in terms of Mean InfAP, which is ranked 7th among all runs and is ranked 3rd among participating teams. TokyoTech-Waseda_2 to 4 achieved Mean InfAP of 28.0%, 26.2%, and 26.0%, respectively. There was no significant difference between the run 1 and 2.

Figure 3 shows InfAP by semantic concepts. One of our runs achieved the best performance for "Baby", and "George_Bush". For these semantic concepts, audio information captured by our MFCC features improved the detection performance. Deep CNN improved AP by more than 3.0% for "Quadruped", "Baby", "Computers", "Airplane", "Bicycling", "Boat_Ship", "Flags", and "Classroom". This shows that increasing the number of samples helps improving detection accuracy since these concepts are included in the ImageNet dataset.

1.7 Conclusion

We proposed a high-performance semantic indexing system using a deep CNN and GMM supervectors with the six audio and visual features. Our best result was 28.1 % in terms of Mean InfAP, which was ranked third among participating teams in the semantic indexing task. Our future work will focus more on the spatio-temporal analysis based on neural networks.

2 Multimedia Event Detection

We apply GMM supervectors and SVMs to the MED task. This year, we added dense trajectory with MBH features to our system at TRECVID 2013. Five types of appearance features, color features, motion features, and audio features are used in this year. We submitted runs under the condition of EvalFull, noPRF with 10Ex and 100Ex for the Pre-Specified task and the Ad-Hoc task. Under the condition of 100Ex, our results ranked 9th among 12 teams for the Pre-Specified task, and 6th among 11 teams for the Ad-Hoc task. Under the condition of 10Ex, our results ranked 10th among 12 teams for both Pre-Specified and Ad-Hoc

2.1 Features

We use the following five types of features: appearance features (HOG-SP), color features (RGBSIFT-SP), motion features (HOG-VP and IDT), and audio features (MFCC).

Appearance

- Dense HOG features with spatial pyramids (HOG-SP)

  We use histogram of oriented gradients (HOG) features [22], which are sampled densely from 4×4 grids of image frames. Each HOG feature has 32 dimensions, and PCA is applied to it without dimensional reduction.

  We apply spatial pyramids [24] to HOG features. The video is divided into regions that are encoded separately and then concatenated. We use 3 levels of spatial pyramids: 1×1, 2×2, and 3×1 [26].

Color

- RGBSIFT features with spatial pyramids (RGBSIFT-SP)

  Color information is an important characteristic of video. We use RGBSIFT to capture it. We extract Scale-Invariant Feature Transform (SIFT) features [10] from 3 RGB channels, and concatenate them to a 384-dimensional feature. We apply PCA to reduce its dimension to 64.

  We also apply spatial pyramids to RGBSIFT features with 3 levels: 1×1, 2×2, 3×1.

Motion

- Dense HOG features with velocity pyramids (HOG-VP)

  We apply dense HOG features with velocity pyramids [6] to capture motion information. We extract HOG feature from 5×5 pixel grids, and a 1-second frame interval. Each HOG feature has 32 dimensions, and PCA is applied to it without dimensional reduction. We use pyramid structure of original, 2 directions, and 4 directions.

- Dense trajectory with MBH feature (IDT)

  Another type of motion features is dense trajectory with MBH features [31]. We resize images to the width of 160, and skip every other frames of each interval. We use PCA to reduce the number of dimensions of MBH features to 64.

Audio

- MFCC features (MFCC)

  We use Mel Frequency Cepstral Coefficient (MFCC) to capture audio information. The feature has 38 dimensions including ΔMFCC, $\Delta\Delta$ MCFF, Δ power, and $\Delta\Delta$ power. PCA is applied to it while keeping the number of dimensions.

Task	10Ex	100Ex
Pre-Specified	8.0	21.7
Ad-Hoc	6.5	17.2

Table 1: The MAP of our runs

Type of features	MAP
HOG-original	21.7
HOG-SP	26.8
HOG-VP	26.1
CC-HOG-VP	28.1

Table 2: The effectiveness of camera motion cancelled velocity pyramid on MED11

2.2 Event detection using GMM supervectors and SVMs (GS-SVM)

As in the previous years, we use the combination of Gaussian Mixture Model (GMM) and Support Vector Machine (SVM), which has been proved effective [8, 25, 29, 30]. We use maximum a posteriori (MAP) adaptation to convert features to a GMM supervector. As the priori distribution for MAP adaptation, we use Universal Background Model(UBM), which was estimated from the training dataset. We set the number of Gaussian mixtures 512 for all types of features. We train event models from training dataset. We create distance matrix for the training and the test videos, and get SVM score of each video for each feature.

The fusion weights of features were decided by 2-fold cross validation. The threshold is determined by the averaged threshold from the 2-fold cross validation.

2.3 Results and analysis

We submitted one run for the Pre-Specified and Ad-Hoc tasks, under the condition of EvalFull, noPRF, with 10Ex and 100Ex. Our Mean Average Precisions (MAPs) are summarized in the table 1.

We see that MBH features are more effective than HOG features and HOF features in many events with actions (Table.3), for example, “Flash mob gathering”, “Getting a vehicle unstuck”, “Grooming an animal”, “Parade”, “Parkour”, or “Repairing an appliance” on MED11. We also see the effectiveness of camera motion canceled velocity in the table 2 for these events with camera motions.

Figures 4, 5, 6, and 7 compare MAP under the condition of 10Ex in 2014 and that under the condition of 100Ex in 2013 and 2014.

Comparing to last year, our system becomes efficient since we simplified detectors for extracting SIFT and STIP features. Note that we have no significant degradation in performance on the MED11 dataset. To improve the performance, our parameter setting strategies for fusion weights and video resizing should be improved. Since we used two folds for cross validation to save computational costs, the parameters may not be the optimal. An improved fusion scheme, which efficiently increases the number of folds, is needed to solve this problem.

2.4 Conclusion

This year we applied GMM supervectors and SVMs to MED task. We improved our method by adding MBH motion features and by removing redundant appearance features. Our future work includes adding some effective features, and customizing appropriate system settings.

Type of features	MAP
IDT-HOG	21.8
IDT-HOF	22.2
IDT-MBH	27.1

Table 3: The effectiveness of MBH features on MED11

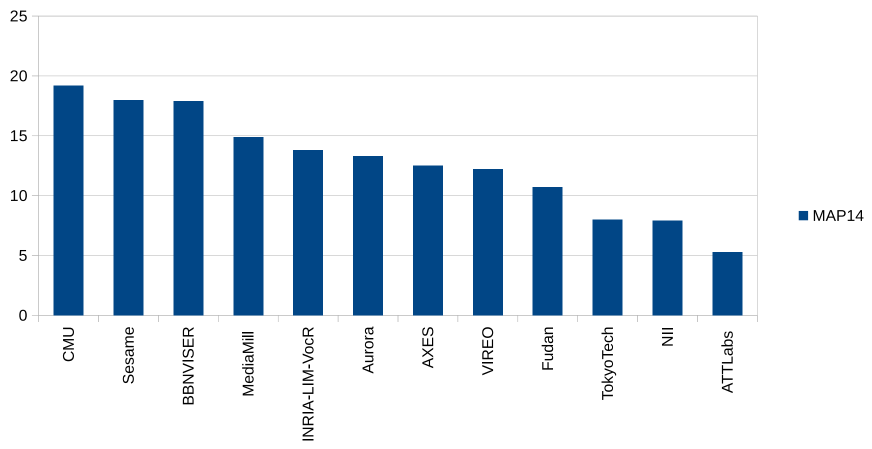

Figure 4: The comparison of MAP in 2014 for Pre-Specified task under 10Ex

3 Instance Search

Our goal is to construct an effective and yet efficient system for the instance search task on a large-scale video database. We exploit the most successful BoW(bag-of-words) framework[34] for this task. Compared with last year, this year we improve the feature detector and descriptor, apply the soft-assignment mechanism, and increase the size of the codebook for getting better performance. In this year the queries are provided not only with the sets of query images, but also with extra video shots from which the query images are extracted. We explore the possibility of making use of these video shots to improve performance, by propagating the object masks through the video shots and utilizing the new object features discovered in the frames to enrich the object models. The experiment results show that the new added/changed components are beneficial for the overall performance.

3.1 System Overview

The overall workflow of our system is depicted in Fig.8. Basically, the system can be viewed as two parts: the off-line indexing part and the on-line querying part.

For a video shot in the database, we sample frames from it and describe the frames with a BoW vector for each, following the standard pipeline[34][30]. To represent the shot, the BoW vectors of its sampled frames are accumulated to one vector and the vector is then normalized, i.e. the aggregated vector is the BoW vector of the shot. After that, an indexing structure called inverted-file[34] is made use of to index the shot BoW vector.

When a user submits a query topic that consists of several images and the corresponding video shots, we encode the information of the object of interest using the standard pipeline of BoW as well[34][30]. Then the constructed BoW vector will be taken to retrieve the relevant video shots that have been indexed beforehand. After getting the initial ranking list, we try to utilize the spatial layouts of the visual words to re-rank the list[32]. Finally, we adopt the query expansion technique to argument the query vector and use the augmented vector to query the database again[36].

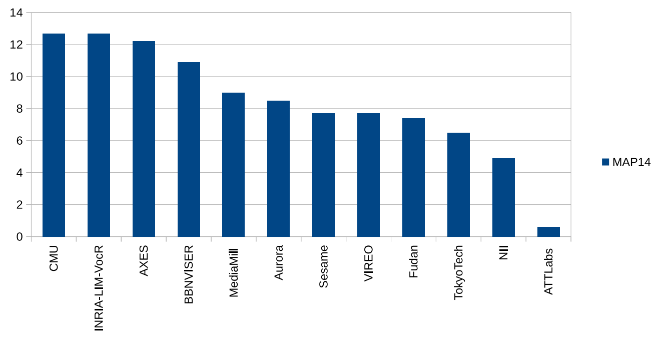

Figure 5: The comparison of MAP in 2014 for Ad-Hoc task under 10Ex

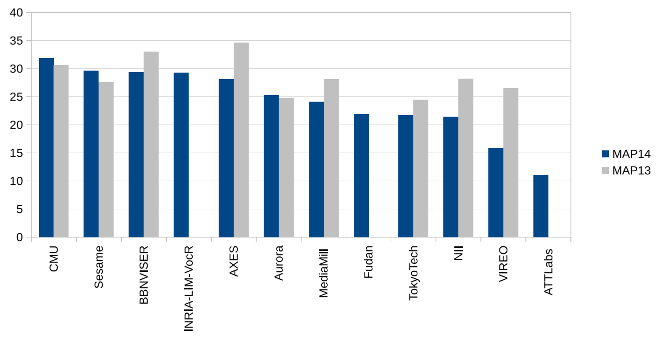

Figure 6: The comparison of MAP in 2013 and 2014 for Pre-Specified task under 100Ex

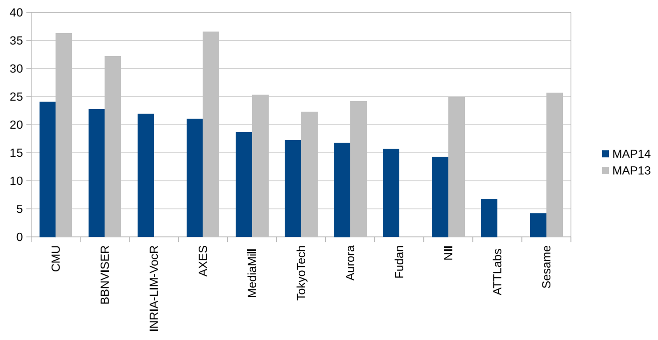

Figure 7: The comparison of MAP in 2013 and 2014 for Ad-Hoc task under 100Ex

3.2 Video Shot Representation and Indexing

In this section, we demonstrate how we represent the database video shots to effectively encode the visual information. Also, we briefly describe the method we use to index the video shots.

To efficiently represent a video shot and avoid a large amount of redundancy, we firstly sample key-frames among the shot. Making trade-off between the information we may lose and the efficiency of the system, we sample key-frames uniformly from the shot, with a rate of 1fps. Uniform sampling helps us to cope with the failures of the shot boundary detection, in which stage a video clip is tried to be segmented into shots. Moreover, according to our observation, one object may appear in some frames while disappearing in the others, even though these frames belong to the same shot. This is mainly due to occlusion or view point changes of camera. Uniform sampling is likely to avoid missing any objects in the scene, as long as the sampling rate is high enough. Notice that last year we only took the middle frame of a shot as its key-frame, leading to too much information loss.

Following the framework of BoW, we first extract local visual features from the frames. Compared with last year, we change the feature detector from MSER[33] to Hessian Affine Detector and normalize

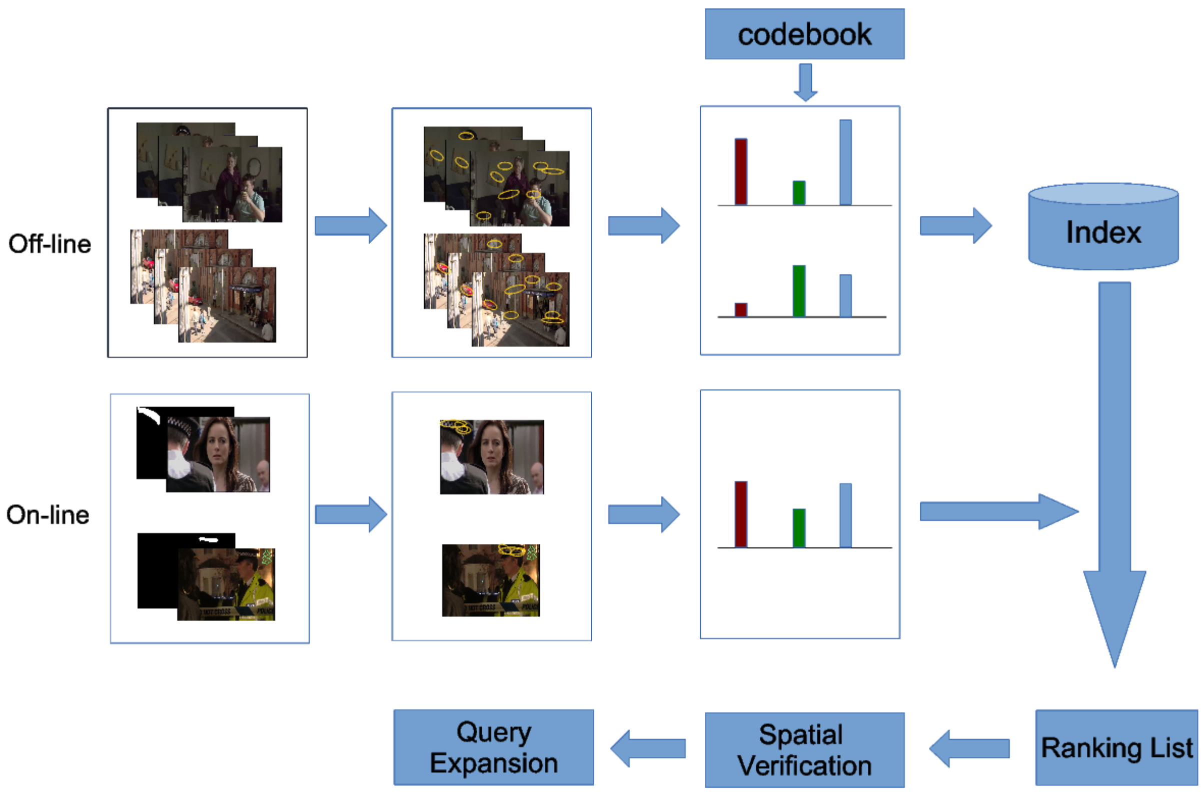

Figure 8: System Overview of Instance Search (Programme material copyrighted by BBC)

the detected image patches before extracting their descriptors. Hessian Affine Detector is able to output much more feature points than MSER, making the retrieval more reliable. And the patch normalization can help to improve the robustness against affine transformations to some extent. And also, we replace the SIFT descriptor with rootSIFT[37] descriptor, which applies non-linear square root transform to each element in the SIFT descriptor, for highlighting the small values.

Then we quantize the rootSIFT features into visual words with a codebook. This year, we increase the size of the codebook from last year's 50K to 1M words. In order to construct the 1M-size codebook efficiently, we follow the idea presented in [32]. We randomly sample 76M RootSIFT descriptors from the database and cluster them into 1M groups using Approximate K-means Algorithm. This algorithm speeds up the general k-means algorithm by replacing the brute-force searching with a more efficient kd-tree-based approximate searching, in the stage of looking for nearest neighbours. As for the quantization, we apply the same approximate nearest neighbour searching algorithm to find the closest descriptor in the codebook for a query feature vector.

To alleviate the quantization error, we adopt soft-assignment as suggested in [35] this year. Instead of assigning a feature to only one visual word, we assign it to several and give different weights to the assignments, according to the distances from the feature to the assigned words. Specifically, we find out three nearest words to the feature and give them the weights based on the Eq.9, where t is the target feature, k is a visual word in the codebook, *d(i,j)* represents the distance between descriptors of i and j, *N(t)* is the set of nearest neighbours of t.

$$Weight_t(k) = \begin{cases} \dfrac{\exp(-d^2(t,k)/2\sigma^2)}{\sum_{i \in N(t)} \exp(-d^2(t,i))/2\sigma^2)} & (k \in N(t)) \\ 0 & (otherwise) \end{cases} \tag{9}$$

Finally, a histogram is constructed based on the occurrences of visual words in the frame. One feature will vote for its three assigned words according to the weights. The histograms of frames that belong to the same shot are accumulated to one vector and that vector is L2-normalized. The normalized histogram is taken as the representation of the shot.

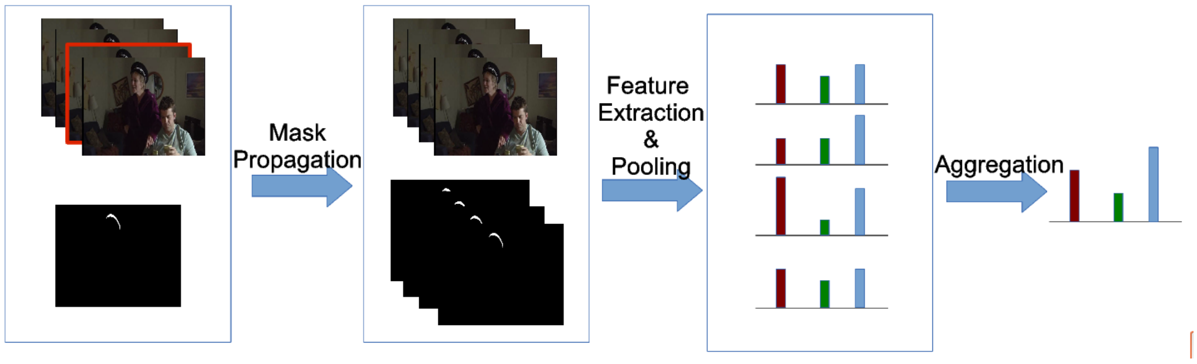

Figure 9: Procedures of Query Vector Construction with a Query Shot Provided (Programme material copyrighted by BBC)

We index the generated shot BoW vectors of the database using a structure called inverted-file[34], as same as last year.

3.3 Query Vector Construction

In the instance search task, we are provided with a query topic, along with several query images accompanied with masks specifying the locations of the object of interest. Being different with last year, the video shots from which the query images are extracted are also offered in this year. In this section, we will briefly describe how we make use of these query materials to issue a BoW vector that catches the key information of the object of interest.

In the case with merely several query images and their corresponding masks provided, we simply extract the features from the regions that are masked as interest, and pool the features in the same way as for the database video shots. After pooling, we can get a BoW vector for each query image that belongs to the same query topic. Then all the BoW vectors for one query topic will be summed up. The accumulated vector is L2-normalized to represent the object of interest.

When the corresponding query video shots are provided in addition to the query images, we try to track the object of interest through the video shots, in order to catch more visual information of the object from different viewpoints when the object or the camera is moving. The basic idea is to propagate the object mask through the query shot based on the optical flows, and extract the features from the new regions specified by the propagated masks. The procedure is shown in Fig.9. In our experiments, we use Gunnar Farneback's method[38] to calculate the optical flows for neighbouring frames and warp the mask accordingly. In order to avoid bad tracking results due to accumulated propagation errors, as well as reducing the computational burden as much as possible, we set up a criterion to decide at which point the propagation should stop. We calculate the Canny Edges of the frame and compare the edge map with the warped mask. If the boundary of the mask can match the edge map well, i.e. the edges of the mask are also the edges of the frame and their gradient directions are almost the same, then we assume the propagation is precise enough and accept the result. When the edge consistence criterion is broken, we stop the propagation.

With the propagated masks, we extract and encode the features from the regions being specified. The encoded BoW vectors for the frames are averaged to issue a query vector.

3.4 Spatial Verification for Re-ranking

Though using the above techniques can effectively retrieve video shots that probably contain the instance topic, the order of them is not precise enough. One reason is that the spatial information has been abandoned in the BoW representation, which is critical in image and video applications. Checking the spatial layout of the words is undoubtedly beneficial for discovering and eliminating the mismatched word pairs.

For each shot candidate in the list, we have represented it in form of several sampled key-frames, and the Hessian Affine feature points have also been detected on the frames. We match the feature points in the query image to the ones in each key-frame if the feature points are sharing common visual words in the codebook. For each matching pair, we estimate an affine transformation that transforms the image plane from query image to the key-frame, by utilizing the geometric information provided by the Hessian Affine Detector. The other matching pairs will cast votes to the estimated affine transformation, if they are consistent with it, i.e. the positions of the projected points by the transformation are sufficiently close to that of the corresponding feature matching points. The affine transformation that gains the most of votes will be regarded as the *true* transformation between the query image and the frame. The procedure is similar as described in [32].

Then we re-score the shots according to the votes of their optimal transformations, which can be described in Eq.10, where k represents the kth shot in the initial retrieval list, *Frames(k)* represents the set of key-frames of the shot, *Inliers(i)* is the set of matching pairs between query image and the frame i that are subject to the optimal affine transformation computed above, $W(M_1)$ and $W(M_2)$ represent the soft-assignment vectors of the two feature points that define the match pair$M$, D_{idf} is a diagonal matrix with the diagonal being the *idf* vector. Here, we take into account the discrimination capacity of the visual words and the distances between the feature point and its assigned visual word vectors, by incorporating the *idf* values and soft assignment weights in the calculation.

$$Re\_score(k) = \max_{i \in Frames(k)} \sum_{M \in Inliers(i)} (D_{idf} W(M_1)).(D_{idf} W(M_2)) \tag{10}$$

Last year, we estimated the affine transformation with coordinates of three matching pairs at each iteration. This year, for each iteration, we only make use of one matching pair, along with the scale and orientation information obtained from the Hessian Affine Detector. Since the number of combinations of three matching pairs is much larger than the number of matching pairs itself, the search for the best affine transformation between a query image and a key-frame is much faster.

3.5 Query Expansion

Query expansion is proven effective in many retrieval applications[36]. In order to improve the retrieval recall, we also apply query expansion in our system.

To perform query expansion, we choose the shots that rank in the front of the retrieval list and use the key-frames, which have more than 10 feature point matching pairs conforming the optimal estimated affine transformation, to expand the original query vector. We construct the BoW vectors of the selected key-frames by pooling the feature points whose projected positions from the key-frames to the query image are in the region of interest. Then we take average of all these BoW vectors with the original query vector and the averaged vector will be issued to query the database again.

The query expansion helps to incorporate new relevant visual words into the query, thus being able to effectively improve the retrieval recall. Since we only choose the key-frames that pass the strict geometry test, it's unlikely that the selected key-frames are not relevant to the query topic. This will guarantee not losing the precision when gaining the improvement in recall.

3.6 Experiments and Results

We totally submitted four runs, being different in the number of query images and whether the corresponding shots of query images are exploited. The results of our four runs are shown in Table 4. We get the best mAP when using three query images, contradicting with our expectation that the best run should be the *4 query images + video shots*. We found that the degradation of average performance while using 4 query images was mainly caused by the degradation of topic 9107 and the topic 9108, from 0.211 to 0.129 and from 0.283 to 0.052 respectively. The degradation of these two topics may be due to the presence of some visual words in the 4th query image that are not discriminant. Comparing the run of *4 query images* and the run of *4 query images + video shots*, we can see a little improvement, which suggests the effectiveness of our query vector construction method mentioned in section 3.2.

In Fig.10, we show the overview of all the participating teams' best results. Our team rank 11th among 22 teams. Comparing with the result of last year, which was 0.0057 in mAP, we achieve better performance.

3.7 Conclusion

We focus on improving our retrieval system this year and make some effective changes in the BoW framework. Specifically, we change the feature detector, feature descriptor and increase the size of the codebook. And also, we change the way of performing spatial verification while applying query expansion

Runs	**mAP**
1 query Image	0.1362
3 query Images	0.1654
4 query Images	0.1552
4 query Images + video shots	0.1575

Table 4: mAPs of Our Different Runs

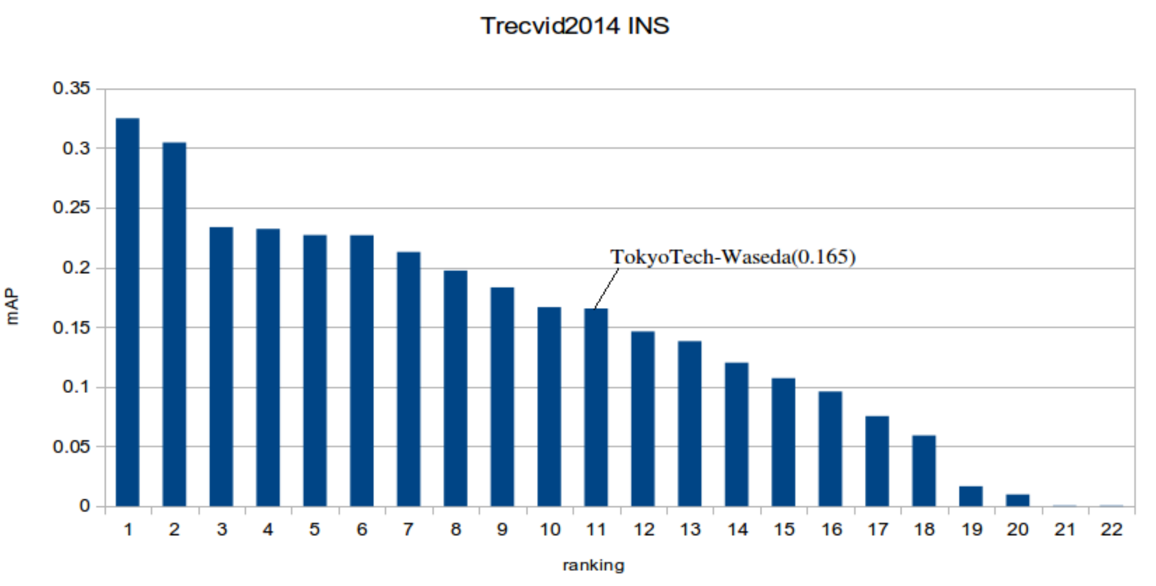

Figure 10: All Participating Teams' Best Results in INS14

technique as well. We also explore the possibility of exploiting the query video shots that are attached to the query images. And the results show that the way in which we exploit can improve the performance a little.

References

[1] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, W. Kraaij, A. F. Smeaton, and G. Quéenot. TRECVID 2014 – An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics. In Proc. of *TRECVID workshop*, 2014.

[2] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVid. In Proc. of *ACM MIR workshop*, pp.321–330, 2006.

[3] A. F. Smeaton, P. Over, and W. Kraaij. High-Level Feature Detection from Video in TRECVid: a 5-Year Retrospective of Achievements. In Multimedia Content Analysis, Theory and Applications, Springer Verlag, pp.151–174, 2009.

[4] S. Ayache, and G. Quéenot. Video Corpus Annotation using Active Learning. In Proc. of *ECIR*, pp.187–198, 2008.

[5] S. Strassel, A. Morris, J. Fiscus, C. Caruso, H. Lee, P. Over, J. Fiumara, B. Shaw, B. Antonishek, and M. Michel. Creating HAVIC: Heterogeneous Audio Visual Internet Collection. In Proc. of *LREC*, 2012.

[6] Z. Liang, N. Inoue and K. Shinoda, Event Detection by Velocity Paramid. In Proc. of 20th Anniversary International Conference Multimedia Modeling (MMM), pp. 353-364, 2014-01.

[7] N. Inoue, and K. Shinoda. A Fast and Accurate Video Semantic-Indexing System Using Fast MAP Adaptation and GMM Supervectors. In IEEE trans. on Multimedia, vol.14, no.4, pages 1196–1205, 2012.

[8] N. Inoue, and K. Shinoda. A Fast MAP Adaptation Technique for GMM-supervector-based Video Semantic Indexing Systems. In Proc. of *ACM Multimedia* (short paper), 2011.

[9] N. Inoue, and et al. High-Level Feature Extraction using SIFT GMMs and Audio Models. In Proc. of *ICPR*, 2010.

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *In IJCV*, 2004.

[11] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. In *IJCV*, 60(1):63–86, 2004.

[12] J. van de Weijer and C. Schmid. Coloring local feature extraction. In Proc. of *ECCV*, vol.2, pages 334–348, 2006.

[13] X. Wang, T. X. Han, and S.Yan. An HOG-LBP Human Detector with Partial Occlusion Handling. In Proc. of *ICCV*, pages 32–39, 2009.

[14] B. Safadi and G. Qunot. Re-ranking by Local Re-scoring for Video Indexing and Retrieval. In Proc. of *CIKM*, 2011.

[15] A. Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms, http://www.vlfeat.org/, 2008,

[16] S. J. Young, G. Evermann, M. J. F. Gales, D. Kershaw, G. Moore, J. J. Odell, D. G. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. The htk book, version 3.4, 2006.

[17] M. Hradis, M. Kolar, J. Kral, A. Lanik, P. Zemcik, and P. Smrz. Annotating Images with Suggestions - User Study of a Tagging System. In Proc. of *ACIVS*, 2012.

[18] T.Ojala, M.Pietikainen and D.Harwood. Acomparative study of texture measures with classification based on feature distribution. *Pattern Recognition*, vol. 29, no. 1, pp. 51-59 1996.

[19] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. CVPR*, vol. 1, 2001.

[20] M. Isard and A. Blake. Condensation: Unifying low-level and high-level tracking in stochastic framework. *Proc. ECCV*, 1998

[21] B. K. P. Horn and B. G. Schunck, Determining optical flow. *Artifical Intelligence*, vol. 17, no. 1, pp. 185-203, 1981.

[22] N. Dalal and B. Triggs.B. Histograms of oriented gradients for human detection. In Proc. of *CVPR*, 2005.

[23] I. Laptev. On space-time interest points. In *IJCV*, vol.64(2), pp.107-127, 2005.

[24] S. Lazebnik, C. Schmid and J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In Proc. of *CVPR*, 2006.

[25] N. Inoue, Y. Kamishima, T. Wada, K. Shinoda and S. Sato, TokyoTech+Canon at TRECVID 2011. TREC Video Retrieval Evaluation (TRECVID) workshop, Dec.5, 2011.

[26] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, Learning Realistic Human Actions from Movies. In Proc. of *CVPR*, pp. 1–8, 2008.

[27] H. Wang, M. M. Ullah, A. Klšer, I. Laptev, and C. Schmid, An Evaluation of local spatio-temporal features for action recognition. In Proc. of *BMVC*, 2009

[28] Y. Kamishima, N. Inoue, K. Shinoda, Event detection in consumer videos using GMM supervectors and SVMs. EURASIP Journal on Image and Video Processing. Vol. 2013, pp. 1-13. Sep 2013

[29] N. Inoue, Y. Kamishima, K. Mori and K. Shinoda, TokyoTechCanon at TRECVID 2012. TRECVID 2012, Nov, 2012

[30] N. Inoue, K. Mori, Z. Liang, M. Lin, K. Shinoda and S. Sato, TokyoTechCanon at TRECVID 2013. TRECVID 2013, Nov, 2013

[31] H. Wang and C. Schmid, Action recognition with improved trajectories. In Proc. ICCV, 2013.

[32] J. Philbin, O. Chum, M. Isard, J. Sivic and A. Zisserman, Object retrieval with large vocabularies and fast spatial matching. In Proc. of *CVPR*, 2007.

[33] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. Ban Gool, A Comparison of Affine Region Detectors. International Journal of Computer Vision, 2005.

[34] J. Sivic, A. Zisserman, Video Google: a text retrieval approach to object matching in videos. In Proc. of *CVPR*, 2003.

[35] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. In Proc. of *CVPR*, 2008.

[36] O. Chum, J. Philbin, J. Sivic, M. Isard, A.Zisserman, Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval. In Proc. of *ICCV*, 2007.

[37] R. Arandjelovic, A. Zisserman, Three things everyone should know to improve object retrieval. In Proc. of *CVPR*, 2012.

[38] G. Farneback, Two-Frame Motion Estimation Based on Polynomial Expansion. Lecture Notes in Computer Science, 2003, (2749), 363-370.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks. Proc. NIPS, 2012.

[40] Y. Jia, et al., Caffe: Convolutional Architecture for Fast Feature Embedding. Proc. ACM Multimedia Open Source Competition, 2014.