

# IRIM at TRECVID 2017: Instance Search

Boris Mansencal<sup>1</sup>, Jenny Benois-Pineau<sup>1</sup>, Hervé Bredin<sup>2</sup>, Alexandre Benoit<sup>3</sup>, Nicolas Voiron<sup>3</sup>,  
Patrick Lambert<sup>3</sup>, Hervé Le Borgne<sup>4</sup>, Adrian Popescu<sup>4</sup>, Alexandru L. Ginsca<sup>4</sup>, and  
Georges Quénot<sup>5</sup>

<sup>1</sup>LaBRI UMR 5800, Université Bordeaux / CNRS / Bordeaux INP, Talence Cedex, France

<sup>2</sup>CNRS, LIMSI, Université Paris-Saclay, BP 133, 91403 Orsay Cedex, France

<sup>3</sup>LISTIC, Domaine Universitaire, BP 80439, 74944 Annecy le vieux Cedex, France

<sup>4</sup>CEA, LIST, Laboratory of Vision and Content Engineering, Gif-sur-Yvettes, France.

<sup>5</sup>Univ. Grenoble Alpes, CNRS, LIG, F-38000 Grenoble France

## Abstract

The IRIM group is a consortium of French teams working on Multimedia Indexing and Retrieval. This paper describes its participation to the TRECVID 2017 instance search task.

## 1 Introduction

The TRECVID 2017 instance search task is described in the TRECVID 2017 overview papers [1, 2, 3].

A new type of query was introduced in 2016 and continued in 2017: asking to retrieve specific persons in specific locations.

These queries are applied on a dataset consisting of videos from the BBC EastEnders soap opera. 30 mixed queries are built from 10 locations (*Cafe1*, *Cafe2*, *Foyer*, *Kitchen1*, *Kitchen2*, *Laundrette*, *LivingRoom1*, *LivingRoom2*, *Market* and *Pub*) and 15 persons (*Archie*, *Billy*, *Brad*, *Dot*, *Fatboy*, *Ian*, *Janine*, *Jim*, *Pat*, *Patrick*, *Peggy*, *Phil*, *Ryan*, *Shirley* and *Stacey*). This year topics include for example: *Peggy in Cafe1* or *Billy in Laundrette*.

Two conditions are considered:

- *A* : only provided images are used as examples
- *E* : video are used as examples (and optionally image examples)

Four French laboratories (CEA LIST, LaBRI, LIMSI, LISTIC) as part of IRIM consortium (coordinated by Georges Quénot, LIG) collaborated to participate to the TRECVID 2017 instance search task.

The IRIM approach to retrieve the shots containing a specific person in a specific location consists in three steps: first person recognition and location recognition are performed independently, then a late fusion is applied to produce the mixed query result.

## 2 Person recognition

For person recognition, two methods were developed by LIMSI and CEA LIST.

### 2.1 LIMSI method

For person recognition, the face recognition method developed by LIMSI is derived from the work described in [4].

The *face recognition* module is actually built upon three submodules. First, shot boundaries are detected using optical flow and *displaced frame difference* [5].

Then, face tracking-by-detection is applied within each shot using a detector based on histogram of oriented gradients [6] and the correlation tracker proposed in [7]. More precisely, face detection is applied every 500ms, and tracking is performed at 25fps in both forward and backward directions.

Finally, each face track is processed using the ResNet network with 29 convolutional layers [8] available in the *dlib* machine learning toolkit [9]. This network was trained on both FaceScrub [10] and VGG-Face [11] datasets to project each face into a 128-dimensional Euclidean space, in which faces from the same person are expected to be close to each other. Each face track is described by its average face embedding and compared with that of the target person using the Euclidean distance.

Two variants were tested, that differ only in the way the target embeddings were obtained. In the first case, we apply face detection on the (four) provided example images and use the average face embedding. In the second case, we search the test set for the face tracks corresponding to the provided example images and use face track average face embeddings – hopefully making the resulting embedding less sensitive to pose and illumination variability. The results obtained by these

two variants are hereinafter referred to respectively as *pers1A* and *pers1E*.

The source code for this module is available in *pyannote-video* [12], that was initially introduced in [4].

## 2.2 CEA LIST method

Faces are detected with the OpenCV implementation of the Viola-Jones algorithm [13] using two cascades (front and profile). On the videos, faces are detected every three frames, each frame being resized such that its largest size is less than 500 pixels. On the training dataset, faces are detected in every image with the same process.

Then, most of the study deals with the face description and more particularly the usage of external sources to define the recognition model. The following external sources of information have been considered:

- images automatically collected on the *Bing* and *Google* web search engines. In practice, we collect the  $n$  first answers to the query "eastender XXX" where XXX is the name of the character collected from the images provided with queries
- images coming from You Tube videos that are collected using the same queries as for *Bing*. The videos are split in individual frames before being re-ranked in order to retain only relevant faces.
- images automatically collected on Wikipedia. We use the same query as for Bing, and selected the first article returned that include the character's name. Then, we collected the image of the infobox of the article.

After testing on development data, we only consider the images coming from YouTube, Google and Bing. The images that are collected automatically are then re-ranked according to the method presented in [14]. More precisely, a kNN-based re-ranking is applied in order to select images that are likely to depict EastEnders characters. The selected images are then described by the FC7 of a VGG network [15] that has been adapted to faces according to the method given in [16]. The network is trained with over 5,000 identities and approximately 800 images per identity. Training images are collected from *Bing* and are automatically re-ranked using the same kNN-based procedure as for EastEnders characters. This model attains a 98.6% accuracy on the LFW dataset. This set of signature constitutes a *model of reference* for the topics considered.

The same signature is used to describe each face detected in the collection. For each such face, the system searches its closest neighbors in the *model of reference*. Several criteria were tested to determine whether the face has to be attributed to one of the EastEnders queries, leading to quite similar performances. We finally adopt a decision criterion based the coherency

of the four first neighbours: if they are the same, the tested face is attributed to this identity.

As this method uses auxiliary videos, it was only used for runs of *E* condition. This method is hereinafter referred to as *pers2E*.

## 3 Location recognition

For location recognition, two methods were developed by LaBRI and LISTIC.

### 3.1 LaBRI method

Similarly to INS 2016 LaBRI method [17], the classical Bag-of-Words (BoW) approach was followed. It consists in the following. First, features are detected on regions of each image and described by a feature descriptor. Feature descriptors are then quantized into visual words, creating a visual vocabulary. A similarity is then computed between histogram of quantized features of query image and those of database images.

For features detection, the Harris-Laplace detector is used. Detections are filtered out if they belong to characters bounding boxes (see Section 3.1.1). Kept detected interest regions are then described by the OpponentSIFT descriptor (of dimension 384). The Root-SIFT [18] post-processing step is applied.

Approximate k-means algorithm [19] is then used to compute a vocabulary of  $k=1M$  visual words. Vocabulary on Opponent SIFT descriptors is computed on 24K randomly selected frames from the shots, with one image extracted per shot (that is 5% of the 471K shots). Hard assignment is used to compute the BoW vector. This vector is then weighted by the tf-idf scheme.

To compute shot signatures, a temporal aggregation is used. Several keyframes are uniformly extracted per shot, at a given framerate. A global histogram is computed for all the keyframes of the shot and averaged over the shot. This is the joint average scheme or average pooling used in [20]. This histogram is then normalized. Keyframes are extracted at a rate of 1 fps (that represents  $\sim 1.57M$  images for the 471K shots).

For query, in the *A* condition (only images used as examples for topics), the normalized BoW vector of each example image is used as query signature. In the *E* condition (video examples used for topics), the signature of the shot to which belongs the example image is used as query signature. A similarity (or distance) is then computed between the query signature and all the shots of the dataset (accelerated with an inverted file index).

The L2-norm and the cosine similarity were used respectively for histogram normalization and similarity measure. Other combinations could be considered. Table 2 in Section 6 gives some results for the following

combinations: L1-norm/L1 distance, L2-norm/L2 distance and L2-norm/cosine similarity.

Some filtering (see Section 4) and a re-ranking step (see Section 3.1.2) are then applied.

Each example image (or shot in  $E$  condition)  $e$ , of each location  $l$ , is queried against each shot  $s$ , to obtain a similarity  $Sim(e, l, s)$ . A late fusion operator is applied to get a similarity  $Sim(l, s)$  for each location  $l$  with regard to each shot  $s$ . The MAX operator is used.

The results obtained by this method are hereinafter referred to as *loc1*.

### 3.1.1 Characters filtering

As EastEnders is a soap opera, scenes consist mainly in two or more characters interacting at a given location. Besides, numerous shots show the main characters, shot in close-up, talking to each other, and with not much motion. So a significant part of the features extracted for a frame and even a shot is detected on characters. To compute a shot signature that better represents the location, we want to remove all the descriptors detected on characters and keep only those corresponding to the actual location. Hence, we detect characters to filter out features located on them.

To detect characters, we took advantage of the face detection already performed for the face recognition step (see Section 2.1). From a face bounding box, we construct a bounding area that roughly encompasses the character bounding box. Figure 1 gives an example of such a construction. It is a very coarse approximation of the person bounding box, but it is very fast to compute. Detected features are then filtered keeping only those outside these bounding areas. This filtering process is applied to all the keyframes extracted for the shot.

### 3.1.2 Re-ranking

A re-ranking step is performed on the top ranked shots of the query results. The method is inspired from [21]. First, as queries have several images and shots contain multiple frames, it would be impractical to verify every image-frame pair. A representative pair of query image and video frame is thus selected. For each shot and each query topic, the pair of video frame and query image whose BoW histogram L1 distance is minimal is selected as representative. Then, for this representative pair, a VQ based feature matching is performed in which features quantized to the same words are considered as matches. Finally, a RANSAC method is applied to find the number of matches following the same transformation. This re-ranking method is practical for large datasets in particular because matching is rather fast to compute: there is no computation of distances between actual features and thus no need to load these

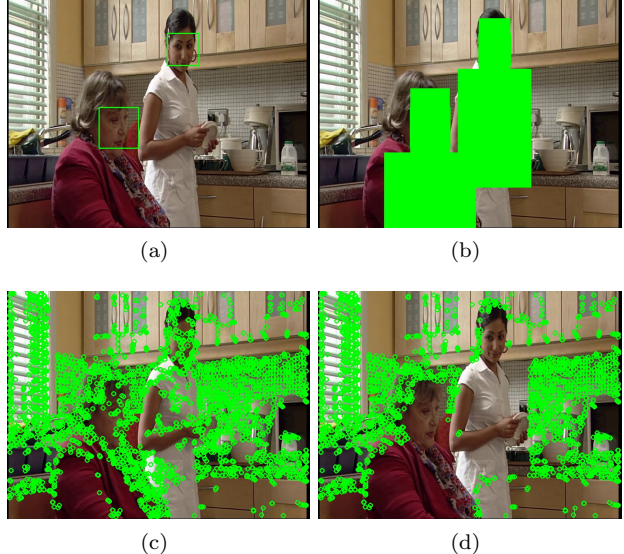


Figure 1: Example of a frame with characters bounding area computation and filtering of keypoints. (a) a keyframe with face detections as bounding boxes. (b) the bounding areas computed for characters. (c) the (3514) features detected on the whole frame. (d) the (2488) kept features after filtering thanks to characters bounding areas. *Programme material copyrighted by BBC*

features from disk. We applied this re-ranking step on the top 3300 results of each location query.

## 3.2 LISTIC method

LISTIC experimented with location recognition using a similarity metric comparing features extracted from a pretrained DCNN without any additional training. The aim was to propose a fast and light approach without explicit overfitting nor specialization to the target data.

The GoogLeNet architecture *GoogLeNet-places365* [22] pretrained on the Places365 location classification dataset was considered. This differs from the 2016 INS submission where the involved network was trained on the more restricted Place205 dataset that limit adaptation to new contexts. The *pool5/7x7\_s1* layer output was considered to generate the feature vectors. Location recognition is performed by comparing the features extracted from the few 90 reference images with the ones of a set of  $n = 10$  frames regularly sampled in time all along the candidate video shots. Last year, a similarity metric considering the candidate video shot location feature variability and inter location feature variability measured from the reference image set was considered but did not prove to be efficient. This year a more selective approach is chosen by looking for the average of the shortest distances between each video shot image sample and the reference images of each

target location. Formally, for each of the  $n$  sampled images of a given candidate video shot, the minimum distance to each target location is computed and averaged over  $n$ . Considering that a video shot presents only a reduced camera rotation, then, the main visual scene features are maintained and averaging cleans the distance measure between shot  $s$  and location  $l$  that is further denoted  $minDistLocation(s, l)$ .

A similarity score to each target location is then computed using equation 1 where  $topicsDistStd$  is the standard deviation of the distances between locations computed from the 90 reference images.

$$Sim(s, l) = \exp\left(\frac{minDistLocation(s, l)}{topicsDistStd}\right) \quad (1)$$

This method actually performed better than the previous attempt while reducing computational cost. In the end, the computational resources are consumed by the convolutional network inference that requires 800ms to process each video shot on average using a NVIDIA K80 GPU.

The results obtained by this method are hereinafter referred to as *loc2*.

## 4 Results filtering

Three filtering steps may be applied to the results of queries.

### 4.1 Credits filtering

The videos from the dataset may contain extra shots unrelated to EastEnders soap opera. In particular, they often contain advertising at the end. As these videos often have opening and end credits, we can detect those in order to remove unrelated shots from results. More precisely, we need to filter out all the shots before the last frame of the opening credits and after the first frame of the end credits.

One difficulty is that the credits are not exactly the same in all the videos. Figure 2 shows examples of frames used for credits.

To detect opening and end credits respectively last and first frame, we use a near duplicate frame detection method. The last frame of opening credits is searched from the start till the  $N_1$ -th frame of the movie. The first frame of the end credits is searched from the  $N_2$ -th frame of the movie till the end of the video.  $N_1$  is arbitrarily set to 3500.  $N_2$  is computed to be 97% of the movie length. On these segments, we compute the minimal distance between the current frame and a set of example frames (see Figure 2). The distance is computed as one minus the correlation of the histograms (of 32 bins) computed on the luminosity channel of the

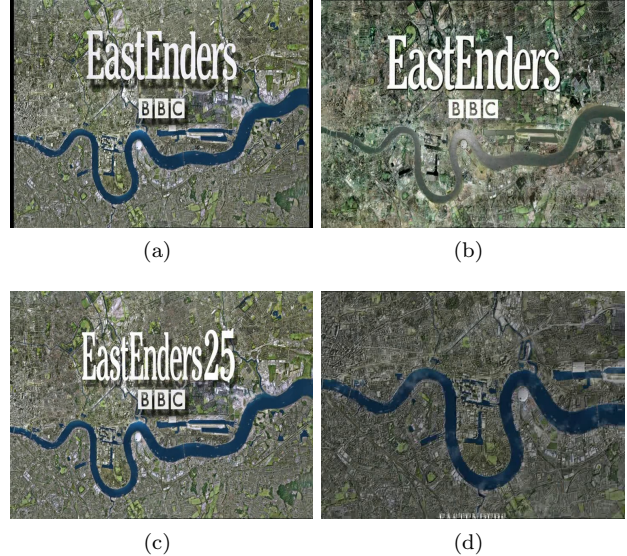


Figure 2: Examples of opening and end credits frames. (a), (b) and (c) show different opening credits last frame examples. (d) shows an example of first frame of end credits, with the start of the rolling credits at the bottom. *Programme material copyrighted by BBC*

two frames. If the minimal distance is below a fixed threshold, frames are considered to be duplicate.

If the end (resp. start) of the opening (resp. end) credits is found, the similarities of shots corresponding to frames before (resp. after) this frame are substantially lowered. This filtering operation is hereinafter referred to as  $p_c$ . The new similarity  $p_c(sim)$  is computed as a fraction of the current similarity  $sim$ :  $p_c(sim) = \alpha_c * sim$ , with  $\alpha_c$  respectively set to 0.1 and 0.2 for opening and end credits.

This filtering using opening and end credits is hereinafter referred to as  $F$ .

### 4.2 Indoor/Outdoor shots filtering

For query regarding an indoor (respectively outdoor) location, results should also contain only indoor (respectively outdoor) locations. To this end, an indoor/outdoor classifier is applied to the query images and shots, and only shots of the same category than the query image (or shot) are kept in the results.

This classifier is built on the *Places365* database and models, derived from the work by [23]. The 365 categories of the database have been manually classified: 190 categories as indoor, 175 categories as outdoor. The pre-trained *Places365 VGG16* model is applied to each image. An image is classified as indoor (respectively outdoor), if the majority of the 365 categories are in the indoor (respectively outdoor) category. Time permitting, this rudimentary classifier should be replaced by one model fine-tuned to detect these two

categories.

This filtering using indoor/outdoor categorization is hereinafter referred as *I*.

### 4.3 Shot threads filtering

Inspired from [24], we compute shots threads, that is temporally constrained clustering of shots that appear similar. A shot belongs to a cluster if the the intersection of the BoW signatures between this shot and the other shots of the cluster is inferior to a threshold.

From these shots threads, a filtering step of results is derived where similarities of shots belonging to the same shot thread (or cluster) are combined with a fusion operator.

TODO: détailler opérateur de fusion !!!

This filtering using shots threads is hereinafter referred as *C*.

## 5 Late Fusion

Once the scores for the face recognition and location recognition steps are computed, we apply a late fusion operation, denoted  $\oplus$ . As scores are of different nature (distances for *pers1* and *pers2*, similarities for *loc1* and *loc2*), we apply the fusion operator on the ranks. For two ranks *rank1* and *rank2*, the chosen operator  $\oplus$  is a simple linear combination of the ranks:

$$\oplus(rank1, rank2) = \alpha * rank1 + (1 - \alpha) * rank2 \quad (2)$$

This operator may be used to fuse the two location results, and the two person results independently. Then it is finally used to fuse person and location results.

TODO: decire alternative fusion !!!

## 6 Evaluation of the submitted runs

Seven runs were submitted for both conditions A and E by IRIM in 2017. Table 1 presents the results obtained by these runs as well as the best and median runs for comparison.

The six fully automatic runs of PKU-ICST were ranked first, thus IRIM with its best run finished second in terms of participants.

The seven submitted runs by IRIM may be described by the following equations (where  $\oplus$  and  $\diamond$  are ranked

rank	System/run	MAP
1	Best run: F_E_PKU_ICST_1	0.5491
7	F_E_IRIM_1	0.4466
8	F_E_IRIM_2	0.4173
9	F_E_IRIM_3	0.4100
12	F_A_IRIM_2	0.3889
13	F_A_IRIM_3	0.3880
17	F_E_IRIM_4	0.3783
18	F_A_IRIM_4	0.3769
16	Median run	0.3800

Table 1: IRIM, best and median runs results among the 31 fully automatic INS submitted runs.

based fusion methods):

$$F\_E\_IRIM\_1 = ((p1E \oplus p2E) \oplus (l1E \oplus l2E))$$

$$F\_E\_IRIM\_2 = (p1E \oplus (l1E \oplus l2E))$$

$$F\_E\_IRIM\_3 = (p1E \oplus l1E)$$

$$F\_E\_IRIM\_4 = (p1E \diamond l1E)$$

$$F\_A\_IRIM\_2 = (p1A \oplus (l1A \oplus l2A))$$

$$F\_A\_IRIM\_3 = (p1A \oplus l1A)$$

$$F\_A\_IRIM\_4 = (p1A \diamond l1A)$$

where:

$$p1E = pers1E + C$$

$$p2E = pers2E + C$$

$$l1E = loc1E + F + I + R + C$$

$$l2E = loc2E + F + I + C$$

and likewise for A condition.

As a remainder:

- F indicates the begin and end credits filtering
- I indicates the indoor/outdoor filtering
- C indicates the filtering by shots threads
- R indicates the application of the re-ranking step

In F\_E\_IRIM\_1, all four individual methods are used. As *pers2* is only available for the *E* condition, F\_A\_IRIM\_1 was not be submitted. In F\_{A,E}\_IRIM\_2, only three individual methods are used. In F\_{A,E}\_IRIM\_3, only two individual methods are used. F\_{A,E}\_IRIM\_4 is similar to F\_{A,E}\_IRIM\_3 except that an alternative fusion function is used.

From Table 1, we can make several observations:

- The alternative fusion function  $\diamond$  (used in runs F\_E\_IRIM\_4 and F\_A\_IRIM\_4) is worse than the original fusion  $\oplus$ .
- The runs in the *E* condition are better than the runs in the *A* condition.

- The differences between F\_E\_IRIM\_2 and F\_E\_IRIM\_3, and between F\_A\_IRIM\_2 and F\_A\_IRIM\_3, seem to indicate that *loc2* does not bring much improvements to *loc1* alone.
- The difference between F\_E\_IRIM\_1 and F\_E\_IRIM\_2 seems to indicate that that *pers2E* fused with *pers1E* brings some improvements to *pers1E* alone.

In order to better understand the individual contributions of our methods, some groundtruth is required. With the results, NIST also provides the groundtruth for mixed queries (person  $P$  in location  $L$ ). But to assess the individual results of location or person recognition methods, individual groundtruth for locations and persons is needed. To this end, we have extracted individual groundtruth from the NIST provided groundtruth. Indeed as we have the groundtruth for a person  $P$  in locations  $L_1, \dots, L_n$ , we can extract the individual groundtruth for  $P$  as the union of all groundtruth relative to  $P$ . Likewise, we have the groundtruth for persons  $P_1, \dots, P_n$  in location  $L$ , we can extract the individual groundtruth of  $L$  as the union of all groundtruth relative to  $L$ . It is noteworthy that the groundtruth extracted this way is very incomplete. In particular, if a person  $P$  is present in numerous locations different that the ones in the mixed queries, these shots will not be in the groundtruth of  $P$ .

Table 2 presents the mAP computed on our different individual methods and some variants, for both the 2016 and 2017 queries. This mAP is computed on all the returned shots and not only on the first 1000 shots as for the mixed query results. Indeed, as the last fusion step between person and location results is some kind of intersection, it is possible that a shot with person  $P$  in location  $L$  appears at an elevated rank in each individual results.

As the groundtruth is only partial, the mAP computed with this groundtruth on our individual methods, reported in Table 2, is not exact and probably underestimated. However, it gives a broad idea of the performance of our methods.

Some observations can be drawn from these results:

- A1, A2 and A3 underline that the choice of the histogram normalization and the distance/similarity methods impact the BoW performance. Here, L2/Cosine produces better results.
- A3 and A4, A7 and A9, and B2 and B3, seem to indicate that the two filter options F and I have very little influence on the results. Indeed, they filter very few shots from the results (respectively less than 0.03% and 0.08% of the shots) and moreover these filtered shots should be ranked rather low if the method already works correctly.

- A3 and A5, and A7 and A6, emphasize the improvements brought by the filtering by shots threads.
- A6 and A3, and A5 and A7, highlight the improvements brought by the re-ranking step.
- A6 and C1, A7 and C2, or A9 and C3 seem to indicate that the gain of fusion of *loc2* with *loc1* is limited.
- D1 and D3, D2 and D4, or E1 and E2, seem to indicate the gain of filtering by shots threads is limited for person recognition methods.
- For D3 and F3, and D4 and F4, it is not clear if there is an improvement on both 2016 and 2017 queries. This lack of difference may be due to the incomplete groundtruth. Indeed *pers2E* method has tendency to bring a wider variety of poses in the recognized persons than *pers1E*.

In order to better understand which combinations of methods bring the best results, we consider another (unsubmitted) run F\_E\_IRIM\_5, such as

$$F\_E\_IRIM\_5 = ((p1E \oplus p2E) \oplus l1E)$$

Besides, retrospectively, we build F\_E\_IRIM\_X', which is the run F\_E\_IRIM\_X where fusion weights have been optimized for the 2017 NIST groundtruth. We also build F\_E\_IRIM\_X'' which is the same than F\_E\_IRIM\_X' except that credits and indoor/outdoor filters ( $F+I$ ) have not been used. Table 3 shows the results of these optimal (unsubmitted) runs:

Some observations can be drawn from these results:

- All F\_E\_IRIM\_X' and F\_E\_IRIM\_X'' are very close. It confirms that the two filtering steps, on credits and indoor/outdoor classification ( $F+I$ ), do not bring significant improvements.
- On 2017 topics, F\_E\_IRIM\_5', F\_E\_IRIM\_3' and F\_E\_IRIM\_2' are worse than F\_E\_IRIM\_1'. It seems to indicate that the fusion of all four methods is the combination bringing the best improvement. However, on 2016 topics, the results are much closer to each other. It is noteworthy that these results on 2016 topics are better than the runs we submitted in 2016.
- It also seem that F\_E\_IRIM\_5' is consistently worse than other combinations.

We will have to further investigate how to improve the fusion of our individual methods, and in particular of our person recognition methods.

## 7 Conclusion

Our system proposes a simple scheme that combines two person recognition methods and two location recog-



Method	MAP	
	2016	2017
A1) loc1E (L1/nL1)	0.1836	0.1050
A2) loc1E (L2/nL2)	0.1777	0.1334
A3) loc1E (L2/Cosine)	0.2551	0.2075
A4) loc1E (L2/Cosine) + F + I	0.2575	0.2093
A5) loc1E (L2/Cosine) + C	0.2766	0.2256
A6) loc1E (L2/Cosine) + R	0.2965	0.2449
A7) loc1E (L2/Cosine) + R + C	0.3292	0.2838
A8) loc1E (L2/Cosine) + F + I + C	0.2783	0.2267
A9) loc1E (L2/Cosine) + F + I + R + C == <i>l1E</i>	0.3302	0.2851
B1) loc2E	0.0663	0.0623
B2) loc2E + C	0.0999	0.0865
B3) loc2E + F + I + C == <i>l2E</i>	0.1000	0.0863
C1) A6 $\oplus$ B2	0.3016	0.2490
C2) A7 $\oplus$ B2	0.3341	0.2853
C3) A9 $\oplus$ B3	0.3351	0.2862
D1) pers1A	0.1305	0.0613
D2) pers1E	0.1425	0.0656
D3) pers1A + C	0.1489	0.0708
D4) pers1E + C == <i>p1E</i>	0.1686	0.0769
E1) pers2E	0.1230	0.0448
E2) pers2E + C == <i>p2E</i>	0.1317	0.0484
F1) D1 $\oplus$ E1	0.1420	0.0662
F2) D2 $\oplus$ E1	0.1335	0.0703
F3) D3 $\oplus$ E2	0.1620	0.0768
F4) D4 $\oplus$ E2	0.1573	0.0827

Table 2: Individual methods evaluations on 2016 and 2017 queries, with individual (incomplete) groundtruth extracted from NIST groundtruth.

Method	MAP	
	2016	2017
F_E_IRIM_1'	0.2984	0.4493
F_E_IRIM_1"	0.2984	0.4516
F_E_IRIM_2'	0.3031	0.4193
F_E_IRIM_2"	0.3025	0.4210
F_E_IRIM_3'	0.2922	0.4130
F_E_IRIM_3"	0.2910	0.4119
F_E_IRIM_5'	0.2762	0.3964
F_E_IRIM_5"	0.2736	0.3990

Table 3: Optimal runs (with and without credits and indoor/outdoor filters) on 2016 and 2017 queries with NIST groundtruth.

nition methods, do a late fusion on each type individually, and apply a final late fusion to get the mixed query results.

Our system effectiveness has been quite improved compared to our previous year participation.

This partial evaluation show that some steps are particularly useful (shot threading for all methods, re-ranking for *loc1*) and other do not bring significant im-

provements (credits and indoor/outdoor classification, *F+I*) and may be removed.

This evaluation also emphasizes that our face and location recognition methods have to be improved individually, and the fusion steps have to be further investigated to better understand which combinations are the most effective.

In particular, several aspects of our methods seem to

be worth investigating.

- The location recognition method based on BoW (*loc1*) gave encouraging results. However, the character filtering is quite rough and it should be explored if it does not filter out too many features.
- The location recognition method based on DCNN (*loc2*) did not give the expected results. Applying a deep-learning approach effectively to location search is still a challenge.
- The two face recognition methods seem to bring faces in very different poses. It should be investigated if a better fusion could take advantage of this diversity.
- Filtering by thread shots improved our results and should be further examined.

## 8 Acknowledgments

This work has been carried out in the context of the IRIM (Indexation et Recherche d'Information Multimédia) of the GDR-ISIS research network from CNRS.

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see <https://www.grid5000.fr>). This work has been partly done thanks to the facilities offered by the Université Savoie Mont Blanc MUST computing center.

## References

- [1] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and TRECVID," in *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, (New York, NY, USA), pp. 321–330, ACM Press, 2006.
- [2] G. Awad, A. Butt, J. Fiscus, D. Joy, A. Delgado, M. Michel, A. F. Smeaton, Y. Graham, W. Kraaij, G. Qunot, M. Eskevich, R. Ordelman, G. J. F. Jones, and B. Huet, "Trecvid 2017: Evaluating ad-hoc and instance video search, events detection, video captioning and hyperlinking," in *Proceedings of TRECVID 2017*, NIST, USA, 2017.
- [3] G. Awad, W. Kraaij, P. Over, and S. Satoh, "Instance search retrospective with focus on trecvid," *International Journal of Multimedia Information Retrieval*, vol. 6, no. 1, pp. 1–29, 2017.
- [4] H. Bredin and G. Gelly, "Improving speaker diarization of TV series using talking-face detection and clustering," in *ACM MM 2016, 24th ACM International Conference on Multimedia*, (Amsterdam, The Netherlands), October 2016.
- [5] Y. Yusoff, W. Christmas, and J. Kittler, "A Study on Automatic Shot Change Detection," in *Multimedia Applications, Services and Techniques*, pp. 177–189, Springer, 1998.
- [6] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893 vol. 1, June 2005.
- [7] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, "Accurate Scale Estimation for Robust Visual Tracking," in *Proceedings of the British Machine Vision Conference*, BMVA Press, 2014.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [9] D. E. King, "Dlib-ml: A Machine Learning Toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.
- [10] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *Image Processing (ICIP), 2014 IEEE International Conference on*, pp. 343–347, IEEE, 2014.
- [11] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015.
- [12] H. Bredin, "pyannote-video: Face Detection, Tracking and Clustering in Videos." <http://github.com/pyannote/pyannote-video>. Accessed: 2016-07-04.
- [13] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I-511–I-518 vol.1, 2001.
- [14] P. D. Vo, A. Ginsca, H. L. Borgne, and A. Popescu, "Harnessing noisy web images for deep representation," *Computer Vision and Image Understanding*, pp. –, 2017.
- [15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.



- [16] Y. Tamaazousti, H. Le Borgne, A. Popescu, E. Gadeski, A. L. Ginsca, and C. Hudelot, "Vision-language integration using constrained local semantic features," *Computer Vision and Image Understanding (CVIU)*, 2017.
- [17] B. Mansencal *et al.*, "IRIM at TRECVID 2016: Instance Search," in *Proceedings of TRECVID 2016*, NIST, USA, 2016.
- [18] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [19] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object Retrieval with Large Vocabularies and Fast Spatial Matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [20] C.-Z. Zhu, H. Jegou, and S. Ichi Satoh, "Query-Adaptive Asymmetrical Dissimilarities for Visual Object Retrieval," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.
- [21] X. Zhou, C.-Z. Zhu, Q. Zhu, S. Satoh, and Y.-T. Guo, "A practical spatial re-ranking method for instance search from videos," in *Image Processing (ICIP), 2014 IEEE International Conference on*, pp. 3008–3012, IEEE, 2014.
- [22] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [23] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning Deep Features for Scene Recognition using Places Database," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 487–495, Curran Associates, Inc., 2014.
- [24] M. Tapaswi, M. Bauml, and R. Stiefelhagen, "StoryGraphs: Visualizing Character Interactions as a Timeline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 827–834, 2014.