IRIM at TRECVID 2018: Instance Search

Boris Mansencal, Jenny Benois-Pineau, Hervé Bredin, Georges Quénot

IRIM at TRECVID 2018: Instance Search

Boris Mansencal[1], Jenny Benois-Pineau[1], Hervé Bredin[2], and Georges Quénot[3]

[1]LaBRI UMR 5800, Université Bordeaux / CNRS / Bordeaux INP, Talence Cedex, France
[2]CNRS, LIMSI, Université Paris-Saclay, BP 133, 91403 Orsay Cedex, France
[3]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 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 2018 instance search task.

1 Introduction

The TRECVID 2018 instance search task is described in the TRECVID 2018 overview papers [1, 2, 3].

A new type of query was introduced in 2016 and continued since: 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 25 persons (*Archie*, *Billy*, *Brad*, *Chelsea*, *Darrin*, *Dot*, *Fatboy*, *Garry*, *Heather*, *Ian*, *Jack*, *Jane*, *Janine*, *Jim*, *Max*, *Minty*, *Mo*, *Pat*, *Patrick*, *Peggy*, *Phil*, *Ryan*, *Shirley*, *Stacey* and *Zainab*). This year topics include for example: *Jane at Cafe2* or *Max at Laundrette*. For persons, 4 example images are given. For locations, between 6 and 12 images are given. Shots from which these images were extracted are provided. A sample video 0 is also given.

Two conditions are considered:

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

Each run must also specify the source of training data:

- *A*: only sample video 0
- *B*: other external data
- *C*: only provided images/videos in the query
- *D*: sample video 0 AND provided images/videos in the query (A+C)
- *E*: external data AND NIST provided data (sample video 0 OR query images/videos)

Two French laboratories (LaBRI, LIMSI) as part of IRIM consortium (coordinated by Georges Quénot, LIG) collaborated to participate to the TRECVID 2018 instance search task with fully automatic runs.

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.

Due to reduced man power, IRIM 2018 participation kept part of our 2017 location recognition method[4] and the same late fusion scheme, and only focused on improving person recognition performance.

2 Person recognition

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

2.1 LIMSI method

For person recognition, the face recognition method developed by LIMSI is similar to LIMSI 2017 method[4]. It is derived from the work described in [5].

This *face recognition* module is actually built upon three submodules.

First, shot boundaries are detected using optical flow and *displaced frame difference* [6].

Then, face tracking-by-detection is applied within each shot using a detector based on histogram of oriented gradients [7] and the correlation tracker proposed in [8]. 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 [9] available in the *dlib* machine learning toolkit [10]. This network was trained on a dataset of about 3 million faces and 7485 identities, derived from FaceScrub [11] and VGG-Face [12] datasets. It projects each face into a 128-dimensional Euclidean space, in which faces from the

same person are expected to be close to each other (Euclidean distance d ¡= 0.6). 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* [13], that was initially introduced in [5].

2.2 LaBRI method

The previously described LIMSI method was our best performing person recognition method of the two methods used in our 2017 participation[4]. However, two issues were identified with this method.

- The face detector works mainly for frontal faces and thus misses lots of detections. Faces are missed on database movies but more tragically on example images. For instance, no face is detected on any of the 4 provided example images for "Mo" person. Besides, the rather low framerate at which detections are made on movies (i.e., every 500ms) is also a cause of detections misses.

- The averaging of face embeddings may not be optimal for correct face recognition. Indeed we may average faces of variable quality.

LaBRI method tries to address these issues.

First, the face detection is done with another detector, the pre-trained CNN based face detector, using a MMOD loss function[14], available in *dlib* [10]. This detector is applied at a framerate of 8fps on the movies. Then, as in LIMSI method, each face is projected into a 128-dimensional Euclidean space using the same ResNet network with 29 convolutional layers, also available in *dlib* [10]. However, here, no face tracking is done and no average of face embeddings is computed.

The distance between a query person and a shot is determined by computing the Euclidean distance between each face descriptor found on the given example images and each movie face descriptor. For a given shot, we compute the Euclidean distance ($d_{q,s}$) between each face descriptor (f_q) found on the given example images and each face descriptor (f_s) found on the shot frames. In order to get one final distance for a given query example, we first combine the distances for the example images. For instance, for 4 detected faces on 4 example images for a given person, we combine the 4 respective distances with each face descriptor from the shot computing the mean of the K minimum distances (K in [0; 4[). It is noteworthy that K=1 is equivalent to compute the MIN of the 4 distances, and K=4 is the MEAN of the 4 distances. Finally, in order to get a final distance for the shot, we combine the M distances for the M face detections on the shot by computing the MIN.

This method is hereinafter referred to as *pers2*: *pers2A* for A condition, *pers2E* for E condition.

This method may be completed in three ways.

2.2.1 Data augmentation

In order to complete the 4 provided example image per person, we automatically collected images from Google Images.

We made automatic queries with "EastEnders XXX" where XXX is the name of the searched person. We kept the 200 first top results of the query. Then, in order to check that the returned images really contain the searched person, for each returned image, we detect faces on the image, compute the face descriptors, and compute the distances between each face descriptors and the four given example face descriptors. If this distance is inferior to a threshold th_f for one of the four provided example images then the face descriptor is kept as a new person example. As the face recognition network was trained to classify faces of a given identity with an Euclidean distance inferior to 0.6, we chose $th_f = 0.51$.

It is noteworthy that the query time depends in the number of images we want to get back. Both queries and checks can be parallelized. Finally, we do a two-by-two distance check to eliminate possible duplicates on kept descriptors. This way we obtained approximately a mean of 42 additional example face images per query person. The minimum is only 1 additional image (for *Darrin*) and the maximum is 115 additional images (for *Max*). Figure 1 show examples of faces of retrieved images for data augmentation for two persons: *Brad* and *Patrick*. We can see that retrieved images quality is quite varied.

This method is hereinafter referred to as *G*.

2.2.2 Face reranking

Similar to NII-Hitachi-UIT work for INS17[15], we do face reranking by training an SVM to classify faces corresponding to a specific person.

During a first search, the face descriptors of the top-N shots, with the smallest distance to the given example face descriptors, are kept as positive samples. For frames with several face detections, the descriptors with the second smallest distances are kept as negative samples. A SVM is then trained to classify faces corre-

Figure 1: Example of data augmentation faces for two characters: *Brad* on first row, and *Patrick* on second row

sponding to query person. We tested both a linear and RBF kernel.

This method is hereinafter referred to as *S*.

2.2.3 Transcripts

Using NIST provided transcripts, we do face post-filtering.

We extract shots where the name of the searched person appears. If the person name is present in a shot, we boost this shot and k nearby shots. Indeed, if her name appears in the transcript, it could give a hint that this person is present in the scene. However, the person is not necessarily present when her name is pronounced. In particular, if the "shot/reverse shot" filming technique is used, when person A is speaking to person B, only person A is visible on screen. But person B may be shown in the previous or next shot. As we are computing distances between faces descriptors, we actually reduce the found face descriptor distance for the shots: $d' = d * \alpha$ with $\alpha = 0.9$. We used $k = 3$ and took care of boosting a shot at most once.

This method is hereinafter referred to as *N*.

3 Location recognition

For location recognition, the method developed by LaBRI, applied in INS 2016 and 2017[16, 4], was used.

3.1 LaBRI method

The classical Bag-of-Words (BoW) approach with similarity search was applied. It consists in the following. First, sparse features are detected on regions of each example-frame 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 frame and those of database frames.

For features detection, the Harris-Laplace detector, described in [17], is used. Detections are filtered out if they belong to bounding boxes of characters (see Section 3.1.1). Kept detected interest regions are then described by the OpponentSIFT descriptor (of dimension 384). The RootSIFT [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 signature. BoW is then weighted by the tf-idf scheme[20].

To compute shot signatures, a temporal aggregation is used. Several keyframes are uniformly extracted per shot, at a given frame rate. 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 [21]. This histogram is then normalized. Keyframes are extracted at a rate of 1 fps (that represents ∼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).

We used L2-norm and the cosine similarity respectively for histogram normalization and similarity measure.

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.

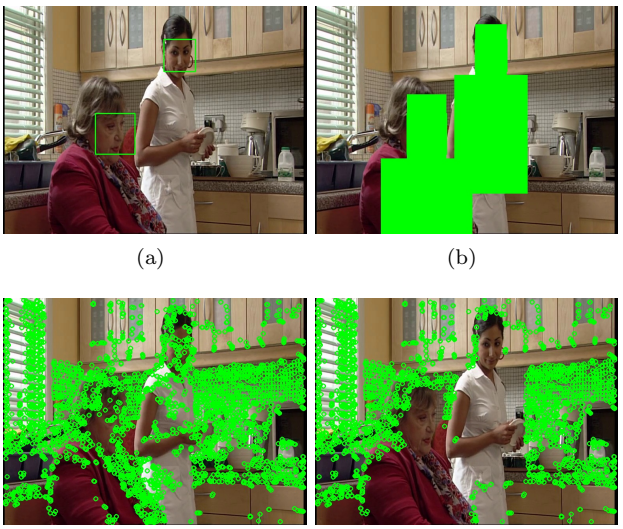
The results obtained by this method are hereinafter referred to as *loc1*.

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 2 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.



(a) (b)

(c) (d)

Figure 2: 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*

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 [22]. 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 affine transformation of image plane. 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 features from disk. We applied this re-ranking step on the top 3300 results of each location query.

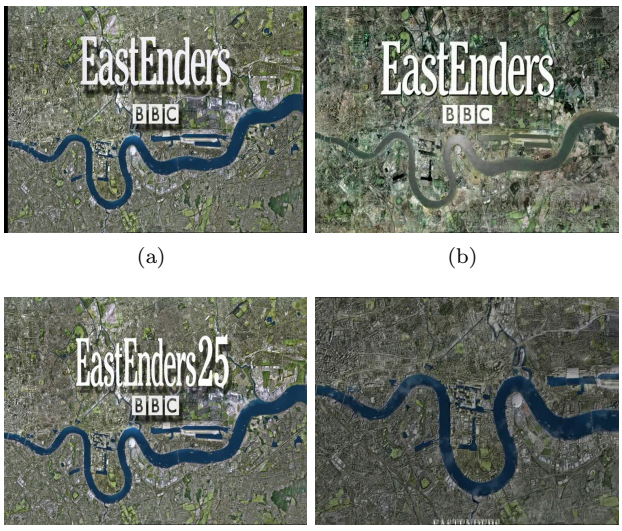
4 Results filtering

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 3 shows examples of frames used for credits.



(a) (b)

(c) (d)

Figure 3: 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*

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 3). The distance is computed as one minus the correlation of the histograms (of 32 bins) computed on the luminosity channel of the 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 α_c respectively set to 0.1 and 0.2 for opening and end credits.

This filtering using opening and end credits is hereinafter referred as C.

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 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.

We used a fusion operator derived from MAX operator. This operator computes the new similarity $sim'(s)$ of shot s from its initial similarity $sim(s)$ and the maximum similarity of the shot thread t this way:

$$sim'(s) = \beta * sim + (1 - \beta) * \underset{s_i \in t}{MAX}(s_i) \quad (1)$$

We used low values of β (typically 0.2).

This filtering using shots threads is hereinafter referred as T.

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*), the fusion operator is applied 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 person results. Then it is finally used to fuse person and location results.

We chose $\alpha = 0.28$ for A condition, and $\alpha = 0.41$ for E condition.

6 Evaluation of the submitted runs

Eight runs were submitted by IRIM in 2018: four runs for A condition and four for E condition. Table 1 presents the results obtained by these runs as well as the best and median runs for comparison.

rank	System/run	MAP
1	Best run: F_E_E_PKU_ICST_1	0.4629
3	F_A_C_IRIM_2	0.4426
4	F_A_E_IRIM_1	0.4424
5	F_E_C_IRIM_2	0.4365
6	F_E_E_IRIM_1	0.4325
9	F_A_E_IRIM_3	0.3980
10	F_E_E_IRIM_3	0.3952
12	F_A_C_IRIM_4	0.3835
15	F_E_C_IRIM_4	0.3772
16	Median run	0.3696

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

Two fully automatic runs of PKU-ICST were ranked first, the first IRIM run was ranked third. IRIM, with its best run, thus finished second in terms of participants.

The eight submitted runs by IRIM may be described by the following equations (where $\oplus$ is the rank based fusion method):

$$F\_E\_E\_IRIM\_1 = (p1E \oplus l1E)$$
$$F\_E\_C\_IRIM\_2 = (p2E \oplus l1E)$$
$$F\_E\_E\_IRIM\_3 = (p3E \oplus l1E)$$
$$F\_E\_C\_IRIM\_4 = (p4E \oplus l1E)$$
$$F\_A\_E\_IRIM\_1 = (p1A \oplus l1A)$$
$$F\_A\_C\_IRIM\_2 = (p2A \oplus l1A)$$
$$F\_A\_E\_IRIM\_3 = (p3A \oplus l1A)$$
$$F\_A\_C\_IRIM\_4 = (p4A \oplus l1A)$$

where:

$p1E = (pers2E + G + T + N) \oplus (pers2E + S + T + N)$
$p2E = pers2E + S + T + N$
$p3E = pers2E + G + T + N$
$p4E = (pers2E + T + N) \oplus (pers1E + T + N)$
$l1E = loc1E + C + I + R + T$

and likewise for A condition.

As a remainder:

- C indicates the begin and end credits filtering
- I indicates the indoor/ourdoor filtering
- R indicates the application of the re-ranking step for locations
- T indicates the filtering by shots threads
- G indicates data augmentation for persons
- N indicates post-filtering using transcripts and person name for persons
- S indicates reranking using an SVM for persons

Some remarks on the submitted runs:

- Runs differ only on the used person location method : they all use the same location recognition method $l1$. This method was studied in details in [4]. The largest contributions to location recognition performance were R and T steps. Even if C and I contributions were negligible, these steps were kept for comparison sake.
- Regarding source of training data, only cases C and E are present. Runs 1 and 3 use data augmentation G, thus are in case E. Runs 2 and 4 only use provided query images/videos and thus are in case C.

From Table 1, we can observe that the best results were obtained for runs 1 and 2, thus with $p1$ and $p2$ person recognition methods. Besides, methods for the A condition seem to produce slightly better results than methods for E condition.

In order to better understand the individual contributions of our methods, we present in Table 2 the mAP computed with different individual face recognition methods and variants, for 2016, 2017 and 2018 queries. In all cases, the same location recognition method $l1$ is used and late fused with face recognition results.

Some observations can be drawn from these results:

- The row A1 corresponds to the *pers1* method used for person recognition. INS18 results for the A condition and INS16 results for the E condition are not present: as two persons were not detected, respectively Patrick and Mo, the final result was non representative.
- The B rows compare the *pers2* method, varying the number K of example faces used to compute the euclidean distance with detections on shot. K varies from 1 to 4. We can see that K=1, i.e., MIN, is the worst of the 4 tested methods. For K=2, 3 or 4, the difference is meaningful on 2017 queries. The best results are obtained for K=2 or 3. K=3 was used for all the submitted runs. Compared to *pers1* method (row A1), we can see that the results are slightly inferior on 2017, but better on 2016 and 2018 queries (when available). It seems to give credit to our hypothesis that computing distances against averaged face embeddings may in certain cases be detrimental to face recognition performance.
- The C1 row allows to evaluate the benefits of using step N, post-filtering with transcripts and person name. Compared to row B2, we can see that it does not really change the performance, except for 2017 queries, where the improvement is noticeable.
- The D rows display the effect of step S, reranking using an SVM. Overall, the results are improved, compared to row B2. Rows D1 and D2, compared to D3 and D4 allow to see the effect of an RBF or linear kernel. The RBF kernel bring better results. Adding step N (D1 vs D2, D3 vs D4) only seems to slightly change the results. D2 is exactly our 2018 submitted run2, that is our better ranked run.
- The E rows allow to see the effects of step G, data augmentation for persons. The results are also improved (E1 vs B2). The step N (E2 vs C1) seems to also have a limited effect. E2 is our 2018 submitted run3.
- The F rows combine step G, data augmentation for persons, followed by step S, reranking using an SVM with an RBF kernel. The results are worst than G or S step alone.
- The G1 row shows the results for another way to combine steps G and S, via late fusion. Here, the

Person recognition method	MAP (condition A)			MAP (condition E)		
	2016	2017	2018	2016	2017	2018
A1) pers1 + T	0.2860	0.3719	X	X	0.4083	0.2849
B1) pers2 (K=4) + T	0.3318	0.3260	0.3529	0.3454	0.3621	0.3536
B2) pers2 (K=3) + T	0.3324	0.3518	0.3603	0.3429	0.3927	0.3590
B3) pers2 (K=2) + T	0.3284	0.3540	0.3508	0.3401	0.3950	0.3503
B4) pers2 (K=1) + T	0.2919	0.2765	0.2772	0.3014	0.3079	0.2818
C1) pers2 (K=3) + + T N	0.3309	0.3762	0.3626	0.3412	0.4163	0.3646
D1) pers2 (K=3) + S (RBF) + T	0.3793	0.4569	0.4409	0.3935	0.5163	0.4347
D2) pers2 (K=3) + S (RBF) + T + N == Run2	0.3788	0.4575	**0.4426**	0.3927	0.5202	**0.4365**
D3) pers2 (K=3) + S (LIN) + T	0.3795	0.4355	0.4258	0.3930	0.4978	0.4156
D4) pers2 (K=3) + S (LIN) + T + N	0.3786	0.4355	0.4279	0.3917	0.5010	0.4180
E1) (pers2 + G) (K=3) + T	**0.3959**	0.4529	0.3974	**0.4033**	0.5021	0.3936
E2) (pers2 + G) (K=3) + T + N == Run3	0.3901	0.4574	0.3980	0.3983	0.5110	0.3952
F1) (pers2 + G) (K=3) + S (RBF) + T	0.3798	0.4417	0.4292	0.3895	0.4905	0.4222
F2) (pers2 + G) (K=3) + S (RBF) + T + N	0.3797	0.4435	0.4305	0.3888	0.4953	0.4241
G1) (pers2 + G) (K=3) + T + N ⊕ pers2 (K=3) + S (RBF) + T + N == Run1	0.3860	**0.4772**	0.4424	0.3965	**0.5315**	0.4325
H1) pers2 (K=3) + T + N ⊕ pers1 + T + N == Run4	0.3673	0.4452	0.3835	0.3765	0.4895	0.3772

Table 2: Various person recognition methods evaluations on 2016, 2017 and 2018 queries, against NIST groundtruth. In all cases, late fusion is done with location recognition method $l1$

results are better than individual methods on 2017 queries and quite similar to D2 results alone for 2018 queries. G1 is our 2018 submitted run1.

- The H1 row displays the results of late fusion of $pers1$ and $pers2$, with step N. This fusion results is better than individual methods alone (A1 and C1). H1 is our 2018 submitted run4.

Overall, we can see that D2 combination gave the better results for 2018 queries, G1 combination gave better results for 2017 queries, and E1 gave better for 2016 queries. So, data augmentation and face reranking seem to improve person recognition results the most. However, even if some results are really close, there is not a single combination that gives the better results on all queries. Besides, it can also be observed that best results on 2016 and 2017 queries are obtained in E condition, but best results on 2018 queries are obtained in A condition. It is noteworthy that this evaluation is done on mixed queries ('person P at location L'). To better evaluate individual methods, individual groundtruth for person or location alone is necessary. As seen in [4], the individual groundtruth extracted from this complete groundtruth is very incomplete and thus is not very meaningful for thorough evaluation.

7 Conclusion

Our system proposes a simple scheme that combines two person recognition methods and one location recognition methods, first do a late fusion on face recognition results, and apply a final late fusion to get the mixed query results.

Our system effectiveness continues to improve compared to our previous year participation.

This year, we reused a part of our 2017 location recognition method, and focused on improving person recognition results.

For the location recognition method, there are still some points that should be examined. For instance, the character filtering is quite rough and it should be explored if it does not filter out too many features. Besides, applying a deep-learning approach effectively to location search is still a challenge.

For the person recognition method, we have combined two methods, that differs in particular on how distances between query and database face descriptors are computed. We also started to investigate how to use data augmentation, faces reranking and transcripts to further improve these results. This partial evaluation shows that some steps, like data augmentation and faces reranking, are particularly useful and bring real improvements to face recognition results and thus final results. But other steps, like post-filtering using transcript and person name, need to be improved to bring significant benefits. PKU-ICST got a high increase of mAP using text-based search and transcripts in their INS17 participation[25]. We should further investigate how these transcripts could be better exploited. Besides, we should also research how to improve the fusion of our individual methods. For example, when doing late fusion of individual person recognition methods,

we do not always get better results. Moreover, no specific combination is better on all the three years topics. We should check in details, query by query, where the differences in performance come from. However, as we only have a ground truth for mixed queries ('person P at location L'), it is not easy to know exactly whether it is the location recognition part or the person recognition part that must be improved first.

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. This work has also been partially carried out in the context of the Guimuteic project funded by Fonds Européen de Développement Régional (FEDER) of région Auvergne Rhône-Alpes. Finally, it was also supported by ANR through the PLUMCOT (ANR-16-CE92-0025) project.

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, K. Curtis, J. Fiscus, A. Godil, A. F. Smeaton, Y. Graham, W. Kraaij, G. Qunot, J. Magalhaes, D. Semedo, and S. Blasi, "Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search," in *Proceedings of TRECVID 2018*, NIST, USA, 2018.

[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] B. Mansencal *et al.*, "IRIM at TRECVID 2017: Instance Search," in *Proceedings of TRECVID 2017*, NIST, USA, 2017.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] D. E. King, "Dlib-ml: A Machine Learning Toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[11] 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.

[12] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015.

[13] H. Bredin, "pyannote-video: Face Detection, Tracking and Clustering in Videos." `http://github.com/pyannote/pyannote-video`. Accessed: 2016-07-04.

[14] D. E. King, "Max-margin object detection," *CoRR*, vol. abs/1502.00046, 2015.

[15] P. Sang *et al.*, "NII_Hitachi_UIT at TRECVID 2017," in *Proceedings of TRECVID 2017*, NIST, USA, 2017.

[16] B. Mansencal *et al.*, "IRIM at TRECVID 2016: Instance Search," in *Proceedings of TRECVID 2016*, NIST, USA, 2016.

[17] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[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] M. J. Salton, G; McGill, *Introduction to modern information retrieval*. McGraw-Hill, 1986.

[21] 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.

[22] 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.

[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.

[25] Y. Peng *et al.*, "PKU_ICST at TRECVID 2017: Instance Search task," in *Proceedings of TRECVID 2017*, NIST, USA, 2017.