# IRIM at TRECVID 2016: Instance Search

Boris Mansencal, Jenny Benois-Pineau, Hervé Bredin, Alexandre Benoit,
Nicolas Voiron, Patrick Lambert, Georges Quénot

# IRIM at TRECVID 2016: Instance Search

Boris Mansencal[1], Jenny Benois-Pineau[1], Hervé Bredin[2], Alexandre Benoit[3], Nicolas Voiron[3], Patrick Lambert[3], and Georges Quénot[4]

[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]LISTIC, Domaine Universitaire, BP 80439, 74944 Annecy le vieux Cedex, France
[4]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 2016 instance search task.

## 1 Introduction

The TRECVID 2016 instance search task is described in the TRECVID 2016 overview paper [1, 2].

A new type of query was introduced in 2016, asking to retrieve specific persons in specific locations.

The dataset consists in videos from the BBC EastEnders soap opera. 10 locations (*Cafe1*, *Cafe2*, *Foyer*, *Kitchen1*, *Kitchen2*, *Laundrette*, *LivingRoom1*, *LivingRoom2*, *Market* and *Pub*) and 7 persons (*Brad*, *Dot*, *Fatboy*, *Jim*, *Pat*, *Patrick* and *Stacey*) are considered. 30 mixed queries or topics are built from these: *Jim in Pub* or *Pat in Kitchen1* for example.

Three French laboratories (LaBRI, LIMSI, LISTIC) as part of IRIM consortium (coordinated by Georges Quénot, LIG) collaborated to participate to the TRECVID 2016 instance search task.

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

## 2 Face recognition

The face recognition method developed by LIMSI is derived from the work described in [3].

### 2.1 LIMSI method

The *face recognition* module is actually built upon three submodules. First, shot boundaries are detected using optical flow and *displaced frame difference* [4]. Then, face tracking-by-detection is applied within each shot using a detector based on histogram of oriented gradients [5] and the correlation tracker proposed in [6]. 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 then described by its average *FaceNet* embedding and compared with that of the target person using the euclidean distance [7].

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 *FaceNet* 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 *FaceNet* 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 *faceA* and *faceE*.

The source code for this module is available in *pyannote-video* [8], that was initially introduced in [3]. Practically, we relied on *dlib* machine learning toolkit [9] for face detection [5] and tracking [6], and on *Openface* [10] for *FaceNet* embeddings [7].

## 3 Location recognition

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

### 3.1 LaBRI method

Similarly to INS 2014 LaBRI method[11], 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 was used. Detected interest regions are then described by the OpponentSIFT descriptor (of dimension 384). The RootSIFT [12] post-processing step is applied.

Approximate k-means algorithm [13] is then used to compute a vocabulary of k=1M visual words. Vocabulary on Opponent SIFT descriptors was computed on 24K randomly selected frames from the shots, with one image extracted per shot (that is 5% of the 471K shots). Hard assignment was used to compute the BoW vector. This vector was then weighted by the tf-idf scheme.

To compute shot signatures, a temporal aggregation was used. Several keyframes were uniformly extracted per shot, at a given framerate. A global histogram was computed for all the keyframes of the shot and averaged over the shot. This is the joint average scheme or average pooling used in [14]. This histogram was then L1-normalized. Keyframes were extracted at a rate of 1 fps (that represents ∼1.57M images for the 471K shots).

We used a dissimilarity, noted $L_1p$ that corresponds to the $L_1$ distance computed on the non-zero subspace of the query, i.e., $L_1$ distance is only computed for the words present in the query. Then a similarity $s = \frac{1}{L_1p+\epsilon}$ is computed from this dissimilarity. Our $L_1p$ dissimilarity can be computed efficiently with the help of an inverted file.

For queries, signatures of shots from which are extracted the example images are used. Each example image $e$, of each location $l$, is queried against each shot $s$, to obtain a similarity $Sim(e, l, s)$. A late fusion operator $f_1$ is applied to get a similarity $Sim(l, s)$ for each location $l$ with regard to each shot $s$. The MEAN operator was first chosen for $f_1$.

#### 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 present 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 tried to detect characters to filter out features detected on them.

To detect characters, we took advantage of the face detection already performed for the face recognition step (cf 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.

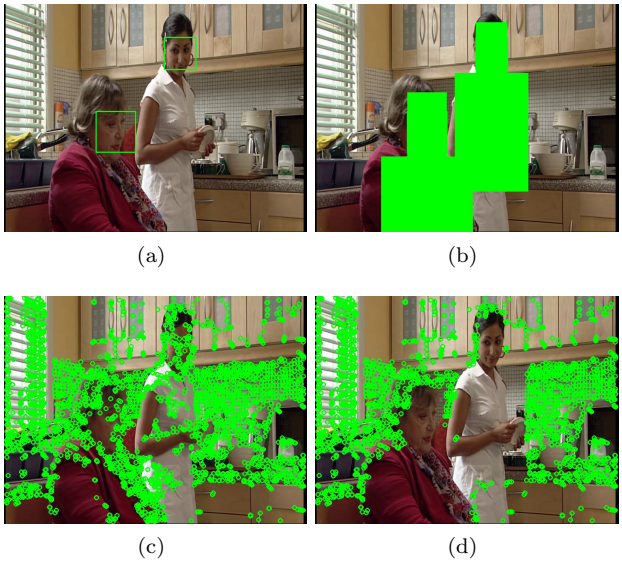



Figure 1: Example of a frame with characters bounding area computation and filtering of keypoints. (a) a keyframe with face detections as bounding boxes. (b) the bounding areas computed for characters. (c) the (3514) features detected on the whole frame. (d) the (2488) kept features after filtering thanks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

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

### 3.2 LISTIC method

Location detection at LISTIC has been experimented using a Deep Convolutionnal Neural Network (DCNN) pre-trained on a generic place recognition task and followed by a metric enabling adaptability to the specific INS task location targets. The considered DCNN is used as a feature extractor and is not retrained nor fine tuned. Only the 90 instance location images describing the 10 location targets have been considered for reference features extraction. Each test video shot is described from the features extracted across 10 images regularly sampled along time. Location matching is ensured by an Euclidean based metric.

#### 3.2.1 DCNN features extraction

The pre-trained *Places205 GoogLeNet* model proposed by [15] has been chosen. This model takes as input a

color image re-sized at resolution (224*224) and its top layer generates a normalized probability distribution of size 205 which corresponds to the 205 classes to be recognized on the MIT places database. Such network is a very generic places classifier trained on worldwide areas such as *airport terminal*, *bar*, *hospital*, *living room*. INS location instances partially match those, in addition provided examples exhibit strong intra instance variability and strong inter instance similarity. As an example, *Cafe1* and *Cafe2* both resemble to each other and have a kitchen connected to the main room. In this context, one will be able to study the discrimination power of such "generic" location detector on the very specific INS targets.

First, the layer generating the most discriminating features on the INS task has to be chosen. In a DCNN, any layer can be considered but the last ones are generally the most interesting since they generate highly generic signatures aggregating lower level features. However, the very last layer is generally specifically adapted to the initial training goal so that the previous ones can be preferable when considering new data and new tasks. As a preliminary step, the confusion matrix obtained over the 10 instance locations has been computed using euclidean distance between either the outputs of the last pooled feature layer $pool5/7\texttt{x}7\_s1$ or the final softmax layer *prob*. For that purpose, a 1-NN search is performed over the 90 instance location example feature vectors and matched location IDs are used to feed the confusion matrix. Figure 2 shows that the soft max probability layer is less discriminating than $pool5/7\texttt{x}7\_s1$ that already shows a good detection behavior. However, some confusion can be observed for resembling locations, in particular *Cafe1* and *Cafe2* and *LivingRoom1* and *LivingRoom2*.

This first evaluation shows the challenge for instance detection from few samples without retraining. One then has to design a metric and identify strategies for enhanced place detection. In the following, each location instance example and any video frame from the test collection is described by its feature vector obtained from $pool5/7\texttt{x}7\_s1$ of the chosen DCNN.

#### 3.2.2 Location detection

The example collection is composed of $m = 90$ examples of $L = 10$ locations ($m = \sum_{i=0}^{L} \#(examples, i)$). We first compute the average distance between locations examples $meanDistInterLocations$. This reference will be used to normalize test shot distances to these examples.

On the test collection, a video shot can show various point of view of its location. Then, in order to facilitate location recognition, each video is described by a set of $n = 10$ frames regularly sampled in time all along the shot length. In the case of video sequences shorter than 10 frames, all the available frames are considered (in that case, $n < 10$).

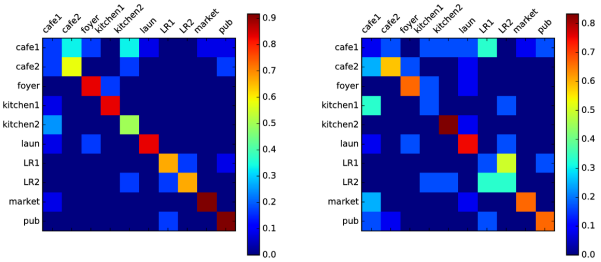



Figure 2: Places205-GoogLeNet location detection confusion matrices for layers $pool5/7\texttt{x}7\_s1$ (left) and the final softmax layer *prob* (right).

Then, for each video shot $s$, a set of $n * (n-1)/2$ euclidean distances between each feature descriptors is computed and the minimum value is used as the reference denoted $minDistIntra(s)$. This reference highlights a baseline related to intra shot distances. Next, for each location $l$, the set of $n * \#(examples, l)$ distances to location is computed and the minimum distance is kept as $minDist(s, l)$. Various similarity metrics have been experimented and minimum intra shot distance was chosen in order to enhance similarity values.

Finally, a similarity measure $Sim(s, l)$ is computed following eq. 1.

$$Sim(s,l) = 1 - max\Big(0, min\Big(1, \frac{minDist(s,l) - minDistIntra(s)}{meanDistInterLocations - minDistIntra(s)}\Big)\Big). \quad (1)$$

Such similarity metric is normalized with respect to the inter location distances (from the few provided examples) and the intra shot distance.

From a computational point of view, this approach is mostly impacted by neural network inference processing time and a shot can be ran in 900ms on a computer based on a NVIDIA K80 GPU.

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

## 4 Results filtering

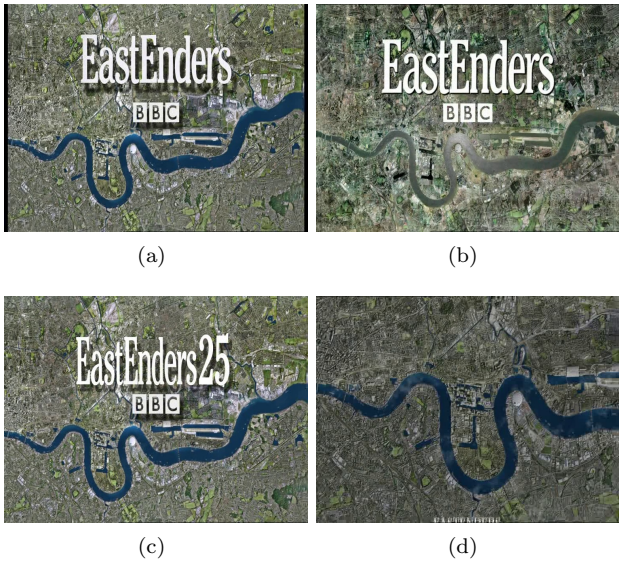
Two filtering steps may be applied to the results of queries.

### 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 re-

sults. More precisely, we need to detect the last frame of the opening credits and 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 $\alpha_c$ respectively set to 0.1 and 0.2 for opening and end credits.

### 4.2 Shot threads clustering

Inspired from [16], we compute shots threads, that is temporally constrained clustering of shots that appear similar (via SIFT matches).

From these, 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.

This filtering operation is hereinafter referred to as $p_t$.

We denote the combination of functions $p_t$ and $p_c$ as $p$: $p = p_t \circ p_c$.

## 5 Late Fusion

Once the scores for the face recognition and location recognition steps are computed, we apply a late fusion operation, denoted $g$. We keep only shots present in both results. As scores are of different nature (distances for *faceA* and *faceE*, similarities for *loc1* and *loc2*), we apply the fusion operator on the ranks. For two ranks $rank1$ and $rank2$, the chosen operator $g$ is a simple linear combination of the ranks:

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

This operator is used to fuse face and location results, denoted $g_{fl}$, or two locations results, denoted $g_{ll}$.

## 6 Evaluation of the submitted runs

Four runs were submitted:

- F_E_IRIM_1 = $g_{fl}(faceE, g_{ll}(p(loc1), p(loc2)))$
- F_E_IRIM_2 = $g_{fl}(faceE, p(loc1))$
- F_A_IRIM_3 = $g_{fl}(faceA, p(loc2))$
- F_A_IRIM_4 = $g_{fl}(faceA, loc2)$

F_A_IRIM_4 and F_A_IRIM_3 differ only in the use of the filtering steps (with credits and shot threads) in the latter. F_E_IRIM_2 differs from F_A_IRIM_3 both in the face and location results used. F_E_IRIM_1 is similar to F_E_IRIM_2, but uses a combination of both location results.

In these runs, both the fusion operator $f_1$ used for $loc1$ (cf section 3.1) and $p_t$ used in $p$ operator (cf section 4.2) were the MEAN operator.

Table 1 presents the result obtained by the four runs submitted as well as the best and median runs for comparison.

In order to understand why our results are so low, we need some groundtruth. With the results, NIST also provided the groundtruth for mixed queries (person $P$

| rank | System/run | MAP |
|---|---|---|
| 1 | Best run: F_A_WHU_NERCMS_1 | 0.7584 |
| 25 | F_A_IRIM_3 | 0.0676 |
| 26 | F_E_IRIM_1 | 0.0645 |
| 27 | F_A_IRIM_4 | 0.0618 |
| 29 | F_E_IRIM_2 | 0.0395 |
| 21 | Median run | 0.1324 |

Table 1: IRIM, best and median runs results among the 41 fully automatic INS submitted runs.

in location $L$). But to assess the individual results of our location or face recognition methods, we need individual groundtruth for locations and persons. To this end, we have first derived individual groundtruth from the NIST provided groundtruth. Indeed as we have the groundtruth for a person $P$ in locations $L1, ..., L_n$, we can extract the individual groundtruth for $P$ as the union of all groundtruth relative to $P$. Likewise, we have the groundtruth for persons $P1, ..., P_n$ in location $L$, we can extract the individual groundtruth of $L$ as the union of all groundtruth relative to $L$. The groundtruth extracted this way is hereinafter referred to as $GT_{NIST}$.

However this extracted groundtruth $GT_{NIST}$ is rather limited: the number of shots annotated for each person or location is low. It does not allow to assess correctly our individual methods. For example, our $faceE$ results for person $P$ may contain correct results, but as person $P$ is not in one of the location queried for INS2016 they are not in $GT_{NIST}$. So we undertook to complete the available groundtruth to have a better assessment of our methods. As it is a very time consuming task, we used a very simplified process. For a given concept (person or location), we annotated the relevance of a shot, only by looking at one of its keyframe. If the concept was present (resp. missing) in the keyframe without ambiguity, we annotated the shot as relevant (resp. non relevant). If there was a doubt, the shot was skipped. We tried to annotate at least (most of) the 4000 first ranked shots returned by each of our methods. Applying this process, we have completed the $GT_{NIST}$ groundtruth for three locations (*Laundrette*, *LivingRoom1*, *Pub*) and four persons (*Brad*, *Dot*, *Fatboy*, *Jim*). This new groundtruth is denoted $GT_{IRIM}$. Table 2 presents the number of relevant shots in the two groundtruths, as well as the number of annotated shots for these concepts. It is noteworthy that the relevant shots for location $L_i$ can be used as non-relevant shots in groundtruth for location $L_j$, for $j \neq i$. But we can not do the same for persons (presence of person $P_i$ does not say anything about presence of person $P_j$). Nevertheless, even with this simplified process, annotation stays a tedious task and our groundtruth $GT_{IRIM}$ is still very incomplete. For *Laundrette*, only 30444 on 471K shots were annotated, that is 6.45% of the total number of shots. For *Jim*, it is only 1.27%.

| concept | #relevant shots $GT_{NIST}$ | #relevant shots $GT_{IRIM}$ | #annotated shots $GT_{IRIM}$ |
|---|---|---|---|
| *Laundrette* | 696 | 4769 | 30344 |
| *LivingRoom1* | 2786 | 2852 | 25178 |
| *Pub* | 5218 | 10444 | 25716 |
| *Brad* | 2988 | 6420 | 11380 |
| *Dot* | 4443 | 8248 | 18839 |
| *Fatboy* | 824 | 3115 | 7723 |
| *Jim* | 503 | 802 | 5979 |

Table 2: Number of relevant and annotated shots in groundtruths $GT_{NIST}$ and $GT_{IRIM}$.

Table 3 presents the MAP obtained for our individual methods on each completed concept in groundtruth $GT_{IRIM}$. We can see that both our location recognition results, *loc1* and *loc2*, are quite low. For face recognition, *faceE* is often better than *faceA*, but not always (for *Dot* for example). In particular, for *Jim*, the results are also quite low.

| method / location | $loc1$ | $loc2$ |
|---|---|---|
| *Laundrette* | 0.1262 | 0.4371 |
| *LivingRoom1* | 0.2671 | 0.1517 |
| *Pub* | 0.1587 | 0.2096 |
| method / person | $faceA$ | $faceE$ |
| *Brad* | 0.4714 | 0.6479 |
| *Dot* | 0.5340 | 0.3667 |
| *Fatboy* | 0.5880 | 0.6531 |
| *Jim* | 0.0263 | 0.2139 |

Table 3: MAP for individual methods on individual concepts (location or person) against $GT_{IRIM}$.

Although incomplete, this groundtruth helped us to start refining our methods. We denote $loc1'$ the results of our $loc1$ method were the fusion operator $f_1$ is changed from MEAN to MAX. Identically, we change the fusion operator $p_t$ used to fuse results for shots belonging to the same shot thread from MEAN to MAX, denoted as $p'_t$. We denote the combination of functions $p'_t$ and $p_c$ as $p'$, such as $p' = p'_t \circ p_c$.

Table 4 presents the results of these modifications on the three locations augmented in $GT_{IRIM}$. The results of method $loc1'$ (first column) are much improved than the results of $loc1$ (cf table 3, first column for location). We can see that the filtering of the opening and end credits alone does not bring much improvement (second

column vs first column). The filtering using the shot threads seems far more beneficial (third column vs first column). Combining the two filtering steps (by credits and shot threads) is marginally better (fourth row vs third row).

| location \ method | $loc1'$ | $p_c(loc1')$ | $p'_t(loc1')$ | $p'(loc1')$ |
|---|---|---|---|---|
| *Laundrette* | 0.5251 | 0.5266 | 0.6783 | 0.6793 |
| *LivingRoom1* | 0.6281 | 0.6281 | 0.7229 | 0.7242 |
| *Pub* | 0.3285 | 0.3285 | 0.4024 | 0.4176 |

Table 4: MAP for modified methods on augmented locations in $GT_{IRIM}$.

Table 5 presents the results of the late fusion between $loc1'$ and $loc2$, with $g_{ll}$ operator used with an optimal $\alpha = 0.95$, and with the two filtering steps applied. Compared to the results of $loc1'$ alone (table 4, last column), results are improved.

| location \ method | $g_{ll}(p'(loc1'), p'(loc2))$ |
|---|---|
| *Laundrette* | 0.7245 |
| *LivingRoom1* | 0.7542 |
| *Pub* | 0.4678 |

Table 5: MAP for fusion of $loc1'$ and $loc2$ results on augmented locations in $GT_{IRIM}$.

This improved individual method $loc1'$ was used to update two of our runs, denoted F_E_IRIM_RUN1′ and F_E_IRIM_2′. Table 6 presents the results that would have been obtained by these two runs (so assessed with NIST provided groundtruth, not $GT_{NIST}$ nor $GT_{IRIM}$). We see that our results are improved, with F_E_IRIM_RUN1′ above the median (cf table 1).

| rank | System/run | MAP |
|---|---|---|
| (17) | F_E_IRIM_1′ | 0.1455 |
| (22) | F_E_IRIM_2′ | 0.1302 |

Table 6: Corrected IRIM runs results among the 41 fully automatic INS submitted runs.

However, despite correct results on individual methods, these results on mixed queries are still quite low.

It seems related to the fact that a person P or a location L may be present a high number of time in the 471K shots, but that the mixed query *person P in location L* is far more uncommon. To hope to have correct results for mixed query, as we use a late fusion on rank, we need to have a very high precision on individual person *and* individual location results. For example, for *Brad*, our face recognition method *faceE*, has a precision greater than 0.92 at rank 2000, 0.82 at rank 4000, 0.71 at rank 6000, on $GT_{IRIM}$. So this character is rather well recognized. However, on $GT_{NIST}$, that is *Brad* in *Foyer*, *Kitchen1*, *Laundrette*, *LivingRoom1* or *Pub*, that corresponds to around 3000 shots, our recall is just 0.20 at rank 2000, 0.36 at rank 4000, 0.45 at rank 6000 and 0.55 at rank 10000. Thus, even at an elevated rank, our method has still not returned shots with *Brad* in the desired locations. This means that *Brad* is present in a very elevated number of shots, probably more than 10000. This entails that we should have a high precision on more than the first 10000 returned shots to hope to return the desired shots with *Brad* in the desired locations.

# 7 Conclusion

Our system proposes a simple scheme that combines face recognition and location recognition with late fusion.

Without any groundtruth, it was difficult to produce an effective system. The partial groundtruth we made helped us to understand where our system was failing.

Both our face and location recognition steps should be improved. In particular, several aspects of our methods seem to be worth investigating.

- The location recognition method based on BoW ($loc1'$) when parametrized correctly gave encouraging results. In particular the character detection filtering, despite quite basic, could be explored.
- The location recognition method based on DCNN ($loc2$) did not give the expected results. Fine tuning on this data collection should be considered.
- Filtering by thread shots improved our results and should be further examined.
- Face recognition should also be improved. In particular with improved face detection, and a higher framerate, we could possibly improve our results.

Results of our system are still quite low, in particular compared to the best run on INS2016. We hope to improve them in the coming year.

# 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 was also partly supported by the CHIST-ERA CAMOMILE project, which was funded by the ANR (Agence Nationale de la Recherche, France).

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, J. Fiscus, M. Michel, D. Joy, W. Kraaij, A. F. Smeaton, G. Quenot, M. Eskevich, R. Aly, and R. Ordelman, "TRECVID 2016: Evaluating Video Search, Video Event Detection, Localization, and Hyperlinking," in *Proceedings of TRECVID 2016*, NIST, USA, 2016.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: a Unified Embedding for Face Recognition and Clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

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

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

[10] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "OpenFace: A general-purpose face recognition library with mobile applications," tech. rep., CMU-CS-16-118, CMU School of Computer Science, 2016.

[11] N. Ballas *et al.*, "IRIM at TRECVID 2014: Semantic Indexing and Instance Search," in *Proceedings of TRECVID 2014*, NIST, USA, 2014.

[12] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[13] 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.

[14] 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.

[15] 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.

[16] 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.