ON-THE-FLY SPECIFIC PERSON RETRIEVAL

Omkar M. Parkhi *Andrea Vedaldi* *Andrew Zisserman*

Department of Engineering Science, University of Oxford, United Kingdom.
{omkar,vedaldi,az}@robots.ox.ac.uk

ABSTRACT

We describe a method of visual search for finding people in large video datasets. The novelty is that the person of interest can be specified at run time by a text query, and a discriminative classifier for that person is then learnt on-the-fly using images downloaded from Google Image search. The performance of the method is evaluated on a ground truth dataset of episodes of Scrubs, and results are also shown for retrieval on the TRECVid 2011 IACC.1.B dataset of over 8k videos. The entire process from specifying the query to receiving the ranked results takes only a matter of seconds.

1 Introduction

Imagine that you have a large corpus of videos, such as a TV station archive or a large archive of internet videos like "The Moving Image Archive", and you need to find shots containing a particular person (and the corpus, of course, lacks complete meta-data describing who is in each video). We describe here a system to meet this need. Our goal is to be able to search for anyone in the corpus (based on their face [1, 2, 3, 4]) with the novelty that we wish to achieve this 'on-the-fly', in the manner of a Google search, where the person is specified by their name and the retrieval is almost immediate for any size of video corpus. We achieve this by pre-processing the video corpus so that it is searchable for *any* person and then, given a text query specifying the person, learning a discriminative classifier for that person from face images downloaded from Google image search. The classifier is used to rank the faces in the corpus, and thereby retrieve the person of interest. This paper extends the original method of [4] by using face tracks on the data set side, and no manual annotation on the downloaded faces for a query. The method is outlined in figure 1.

In the following section we describe the off-line and on-line steps of the web based search system in detail, explaining in particular how the design choices and parallel architecture enable the complete process from a text query to results to be carried out in matter of seconds for a corpus with millions of detected faces. Section 3 then gives a quantitative performance evaluation of the system on a dataset consisting of episodes of television program "*Scrubs*" for which there is ground truth person annotation.

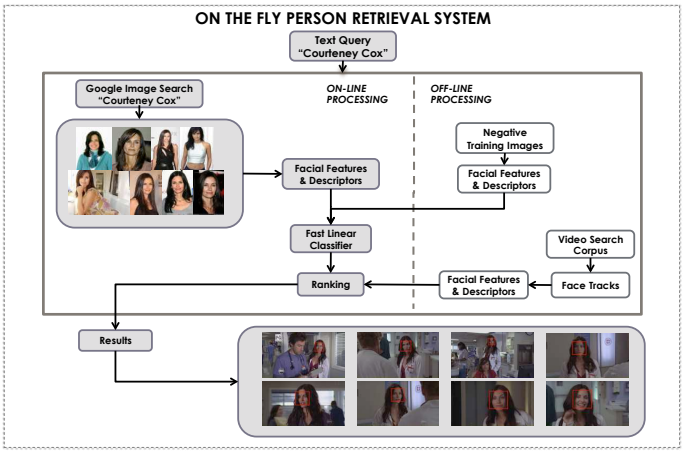

Fig. 1. Block diagram of the On-The-Fly Face Search System. The text query for "Courtney Cox" (CC) is used to obtain training images from Google Image search. A feature vector is computed from each of these faces (with the option of manual supervision to reject faces that are not CC), and a linear Support Vector Machine (SVM) classifier is trained using these as positives together with a reservoir of negative feature vectors. The video data has been pre-processing into face tracks, and the tracks are then ranked by the classifier to obtain the shots in which CC appears. In this example the corpus is of episodes of the TV comedy 'Scrubs' in which CC appeared for a time. The entire process from typing the query to obtaining the ranked shots takes a matter of seconds.

2 Off-line and On-line Processing

In order to achieve the on-the-fly person-specific training and retrieval it is necessary to carry out much of the processing in advance of a query. We describe the off-line steps next, followed by the processing that occurs at run time once a query is typed into the search window.

2.1 Off-line Processing

The off-line part of the system, pre-computes non query specific features in order for the run-time query specific operations to be fast. Non query specific processing includes computing feature vectors for the entire video corpus as well as feature vectors for a set of negative training samples.

The entire video collection is processed in three steps. In the first step, faces are detected in every frame of every video and linked together to form face tracks. In the second step,

false tracks are removed and an exemplar faces detection selected to represent each face track. In the final step feature vectors are computed for the exemplar face of each track to be used for ranking by the SVM classifier.

Face tracks. A face track is a temporal connection of detected faces of a single person. Face tracks are automatically generated using the method described in [5]: near-frontal faces are detected in every frame using the OpenCV Viola-Jones detector [6]. The detector is configured to return high confidence face detections of size greater than 40×40 pixels. These faces are then associated temporally through the video using a RBF kernel-based regressor tracker.

The important issue here is that the granularity of representing people in the video is reduced from a face per frame to a track. Since a face track can link tens or hundreds of face detections this is a substantial data reduction. Table 1 gives statistics for the datasets used in the experiments. Note that the number of tracks is a hundred times less than that of the detections. Also, since a track is a temporal connection of faces of the same person, it provides a quick way of transferring a classification label to detections in a video. Additionally, tracking provides robustness against false face detections.

Facial feature localization. Nine facial features are then detected and localized on every face detection using the method of [7]. These features are left and right corners of each eye, two nostrils and tip of nose, and left and right corners of mouth. Additional features corresponding to centres of the eyes, a point between eyes, and the centre of mouth, are computed from the detected features giving in total 13 different facial feature locations. These facial features are detected using a pictorial structure model [8] consisting of a mixture of Gaussian trees for the layout with discriminative feature appearances. Facial features can be located with high reliability in the faces detected by the face detector despite variation in pose, lighting, and facial expression.

The output of these modules is a consistent linking of faces detections and corresponding feature locations.

Cleaning and representing tracks. Due to erroneous face detections there will be a number of false face-tracks generated. A significant proportion of these can be removed simply based on their length. Face tracks are also filtered out based on the score of the facial feature detector. The remaining tracks are then each represented by combining temporally sampled detections from the track with selecting the 'best' detection based on the facial feature detector score. This score is a measure of confidence of facial feature locations and proves to be useful for selecting suitable candidates.

Representing face appearance. A representation of the face appearance is extracted by computing descriptors for each of the located facial features. Extracting descriptors based on the location of the facial features [2, 9] gives robustness to pose variation, lighting, and partial occlusion. Before extracting the descriptors, the image is converted to grayscale and faces are normalized and affine transformed to a canonical size (80×80) and layout of the feature locations. This is done to reduce the scale uncertainty in detector output and to reduce the effects of pose variations. A facial feature descriptor is computed from the gradients of the pixels in a circular region of radius 7 pixels around each facial feature point, and normalizing them to have zero mean and unit variance. A $3,849$ dimensional descriptor for every face is then formed by concatenating the descriptors for each facial feature.

Dataset	Videos	Hours	Faces	Shots	Tracks
Scrubs	12	5	303,251	4,955	5,743
TRECVid 2011(IACC.1.B)	8,216	226	2,911,805	137,152	25,535

Table 1. Dataset statistics.

Processing negative training data. Negative training face images are pre downloaded from the Internet and are kept the same for all queries. We use a publicly available face dataset as the negative training set (the details are given in section 3). The assumption is that the negative data will not contain a significant number of faces from the particular person we wish to search for. The face detector, facial feature detector and appearance representation pipeline described above is applied to each of the negative images to produce a feature vector.

2.2 On-line Processing

This part performs all query specific tasks which include: querying and downloading images from the Internet; computing feature vectors from the downloaded images to provide positive training samples; training the classifier; and ranking and displaying the results.

Selecting positive training images. When a user enters a text query, the query is transferred to Google Image Search, and the top results are downloaded using a parallel downloading module. This module is a python web service and queries the user string to Google Image Search service. Google Image Search provides an advance search facility to restrict the search to only face images. We use this feature to inhibit any non face images from being downloaded. Figure 2 shows examples of images downloaded for the query "Courteney Cox".

The face detector, facial feature detector and appearance representation pipeline is then applied to each of the downloaded images to obtain a feature vector for every detected face. As expected, some non-faces are detected and some faces will not be of the query person (e.g. if there is more than one person in the downloaded image). There are then two possibilities: either these problems can be ignored and all feature vectors are used to provide the positive training samples for the classifier; or a user can manually select which faces to use. We evaluate both possibilities over a number of examples in section 3. In the case of manual supervision, a simple web-interface is provided using a web service implemented with a combination of Python, HTML and JavaScript.

	Dataset/Actor	Brendan Fraser	Courteney Cox	Michel Fox
1	Training – Scrubs Episode (Pos/Neg)	59/409	99/464	101/445
2	Testing – Scrubs Data (Pos/Neg)	80/3718	45/3753	38/3760
3	Training – Google (Pos)	45	54	34
4	Training – Caltech Faces (Neg)	4559		

Table 2. **Training and testing data statistics**.1: Number of positive and negative tracks in the episode used for training the model for a particular actor. 2: number of positive and negative tracks present for the actor in the test episodes. 3: Number of training samples obtained from Google Image search for the actor. 4: Number of negative training samples from the Caltech Face dataset [11].

This interface allows the user to quickly indicate negative examples simply by clicking on them (the default is that everything is positive).

Training the classifier and ranking. We use a linear SVM as the classification method. A linear SVM is used because it is both fast to train and fast to test (since testing only involves a scalar product between the learnt weight vector and the feature vector, i.e. no sum over many support vectors as in the non-linear SVM case). The training data consists of the feature vectors computed on the downloaded images for the query as positives, together with the pre-computed negative data (as described in the off-line processing of section 2.1 above). The classifier is trained using the software package LibLinear [10] which is optimized for linear classifiers and has complexity linear in the number of training samples. Parameters of the classifier (such as the C value) are optimized using cross-validation off-line and are kept fixed for all searches.

The trained classifier is then used to rank the representative image of each track in the video corpus. An example of the top ranked results trained for Courteney Cox can be seen in figure 2.

3 Datasets and Performance Evaluation

Datasets. Performance is quantitatively evaluated by creating a ground truth dataset for the television comedy series "Scrubs". The dataset consists of 12 episodes from different seasons of the series. Each episode is about 20-25 minutes in length. We have annotated tracks (as positive or negative) for a number of guest actors: Brendan Fraser, Courteney Cox, and Michel J. Fox.; and use these actors to assess performance. It is important to note that by choosing to use guest actors, we increase the difficulty level of the problem as these actors do not appear frequently in the collection compared to a principal cast member. The dataset is split into three episodes for training and nine episodes for testing. Each of the guest actors appears two different videos in the dataset, one them is used for the training and the other one is used for testing purposes. Statistics of the training and test sets are given in table 2.

The negative training data is obtained from images taken from the publicly available Caltech 10,000 Web Faces Dataset [11]. Face detection on this data results in 4559 faces (there is a minimum size for detected faces).

	Training Data		Average Precision					
			Brendan Fraser		Courteney Cox		Michel J. Fox	
	Positives	Negatives	E1	E2	E1	E2	E1	E2
1	Scrubs	Scrubs	0.56	**0.60**	0.88	**0.88**	0.49	**0.53**
2	Scrubs	Caltech	**0.25**	0.24	0.62	**0.72**	0.52	**0.56**
3	Scrubs	Scrubs+Caltech	0.44	**0.55**	0.83	**0.93**	0.58	**0.70**
4	Google	Scrubs	0.40	**0.42**	0.48	**0.47**	**0.38**	0.34
5	Google	Caltech	0.41	0.41	0.56	0.56	0.57	0.57
6	Google	Scrubs+Caltech	0.41	**0.42**	0.57	**0.59**	**0.56**	0.54
7	Chance		0.02		0.01		0.01	

Table 3. **Classifier Performance**. Average Precision values for various combinations of the training data. **E1**: uses a single sample per track in training and testing. **E2**: uses additional samples per track (see text). See figure 2 for examples of ranked results.

Retrieval performance evaluation. Performance is assessed by computing the Average Precision (AP) for the retrieval of each of the three actors in the test set. Classifiers are trained with different combinations of positives and negative training data.

In the experiments the positive data is either from Google Image search (for that query) or from the Scrubs training set; and the negatives either from the Scrubs training set or from the Caltech negative images, or from both. As can be seen in table 3 in almost all cases there is a drop in performance by using positives from Google (with the same set of negatives) compared to positives from the Scrubs training set. However, the performance with Google images is still quite reasonable. Increasing the number of negative examples (using both Scrubs and Caltech) always improves performance, even though the negative images are from a different distribution to the positives. Note, the Google image results are after manual rejection of false faces. If instead *all* the Google images are used, there is a drop in performance of only 3.0%. Given this small drop there is little need for manual or automated selection (e.g. by clustering of the faces as in [9, 12]).

We evaluate retrieval performance for two sampling strategies. The first, *Experiment 1*, uses a single sample per track for training and testing, as described in section 2.1. The second, *Experiment 2*, adds more samples per track using the first, middle and last faces of the track (in addition to the one having highest facial features detection score). For training data, samples are only added if their facial feature confidence is above a threshold (of 5 here). All samples selected are used in training (up to four per track), so there is more training data, and during testing a track is ranked by the maximum classifier score over the selected samples. Note, in almost all cases adding samples improves performance.

The top results for a search on "Courteney Cox" for a classifier trained on a combination of Google Images and Caltech images for positives and negatives respectively are shown in figure 2. We also show qualitative results for queries on the TRECVid 2011 IACC.1.B dataset.

Timings for on-the-fly retrieval. Near real time performance is achieved by exploiting parallel computing architectures.

Fig. 2. **Retrieved Results. Even Rows**: images downloaded from Google for Courteney Cox (CC), Michel Fox (MF), George Bush (GB) and Tony Blair (TB). **Odd Rows**: Top results retrieved by our system on Scrubs (CC,MF) and IACC.1.B data (GB,TB). For Tony Blair, the displayed results are positives in top 10 results. Notice the difference in the lighting, pose etc. between the downloaded Google images and the ones in the video corpus.

Typically 100–150 images are downloaded, which results in 80–90 extracted frontal faces. This process of downloading and extracting faces with feature computation takes less than 10 seconds on 20 parallel threads. The optional labelling of the 80–90 faces takes about a minute, and is the major bottleneck in the system. Linear classifier training and ranking is also done within couple of seconds. So the entire end-to-end process typically takes less than 20 seconds for the fully automated method, and about a minute more if a user annotates the positive face instances.

4 Conclusion and Future Work

We have described an on-the-fly search method applicable to any person known to Google Image Search. There are many variations on this theme. A similar system could be built for searching one's own personal video collection (clips from mobile phones and cameras).

Once the video corpus has been processed, any face (track) within it can provide positive examples for the classifier training, and so any person in the video corpus found initially by browsing can then be searched for. To this end, we also provide the user with an option to find similar faces to one selected in the retrieved results. This is achieved by using the selected face as a positive example and training a SVM classifier using negative training examples from the Caltech dataset. This functionality can be extended in the future by applying various schemes for relevance feedback.

Since a vector represents each track, standard methods such as bag of visual words [13], product quantization [14], etc can be applied to improve retrieval speed and scalability.

Acknowledgements. We are grateful to Ken Chatfield for providing the Google Image Downloader service. Financial support was provided by ERC grant VisRec no. 228180 and EU Project AXES ICT-269980.

References

[1] L. Jin, S. Satoh, F. Yamagishi, D. Le, and M. Sakauchii, "Person X detector," in *TRECVID Workshop*, 2004.

[2] J. Sivic, M. Everingham, and A. Zisserman, "Person spotting: Video shot retrieval for face sets," in *CIVR*, 2005.

[3] J. Yang, M. Chen, and A. Hauptmann, "Finding person x: Correlating names with visual appearances.," in *CIVR*, 2004.

[4] J. Philbin, A. Bosch, O. Chum, J. Geusebroek, J. Sivic, and A. Zisserman, "Oxford TRECVID 2006 – notebook paper," in *TRECVID Workshop*, 2006.

[5] N. E. Apostoloff and A. Zisserman, "Who are you? – real-time person identification," in *BMVC*, 2007.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.

[7] M. Everingham, J. Sivic, and A. Zisserman, "Taking the bite out of automatic naming of characters in TV video," *Image and Vision Computing*, 2009.

[8] P. Felzenszwalb and D. Huttenlocher, "Pictorial structures for object recognition," *IJCV*, 2005.

[9] T. Berg, A. Berg, J. Edwards, and D. Forsyth, "Who's in the Picture," in *NIPS*, 2004.

[10] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Machine Learning Research*, 2008.

[11] "Caltech 10,000 web faces," `http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/`.

[12] A. D. Holub, P. Moreels, and P. Perona, "Unsupervised clustering for google searches of celebrity images.," in *Proc. Int. Conf. Autom. Face and Gesture Recog.*, 2008.

[13] J. Sivic and A. Zisserman, "Efficient visual search of videos cast as text retrieval," *IEEE PAMI*, 2009.

[14] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE PAMI*, 2011.