Detecting and Naming Actors in Movies using Generative Appearance Models

Vineet Gandhi
INRIA / LJK, Grenoble, France
vineet.gandhi@inria.fr

Remi Ronfard
INRIA / LJK, Grenoble, France
remi.ronfard@inria.fr

Abstract

We introduce a generative model for learning person and costume specific detectors from labeled examples. We demonstrate the model on the task of localizing and naming actors in long video sequences. More specifically, the actor's head and shoulders are each represented as a constellation of optional color regions. Detection can proceed despite changes in view-point and partial occlusions. We explain how to learn the models from a small number of labeled keyframes or video tracks, and how to detect novel appearances of the actors in a maximum likelihood framework. We present results on a challenging movie example, with 81% *recall in actor detection (coverage) and* 89% *precision in actor identification (naming).*

1. Introduction

Detecting and naming actors in movies is important for content-based indexing and retrieval of movie scenes and can also be used to support statistical analysis of film style. Additionally, detecting and naming actors in unedited footage can be useful for post-production. Recent advances in face detection, upper-body detection and full-body detection have already been applied to this problem but such detectors are designed to be generic and they discard person specific and costume-specific features. Methods for learning such features are desired to improve the recall and precision of actor detection in long movie scenes where the appearance of actors is consistent over time.

Contributions We propose a complete framework to learn view-independent actor models using maximally stable color regions (MSCR) [10] with a novel clustering algorithm. The actor's head and shoulders are represented as constellations of color blobs where the appearance of each blob is represented in a 9 dimensional space combining color, size, shape and position relative to the actor's coordinate system, together with a frequency term. Based on such a model we propose a detection framework with two stages. The first stage is a search space reduction using the k-nearest neighbours corresponding to the actor by just using the appearance of the blobs in the model. The second stage is a sliding window search for the best localization of the actor in position and scale. By repeating those two steps for all actors at all sizes, we obtain detection windows and actor names that maximize the posterior likelihood of each video frame.

Organization The remainder of the paper is structured as follows. First, we briefly review related work in generic and specific actor detection in movies. Then Section 3 describes the proposed statistical model and how it can be learned from examples. Section 4 describes the corresponding detection algorithms. Experimental results and evaluation of the method are presented in section 5. Finally, Section 6 draws some conclusions and discusses directions for future work.

2. Related work

Much previous work on actor detection and recognition has been based on face detection. Everingham et al. [5] use scripts and subtitles to learn the association between character names and faces in television drama using a frontal face detector. Sivic et al. [18] demonstrate a much improved coverage (recall) by using profile views. In one of their examples, they report that 42% of actor appearances are frontal, 21% profile and 37% are actors facing away from the camera. Drawing on that observation, they suggest that future work on increasing coverage must go beyond the use of face detection as a first step. Indeed, generic methods for detecting upper-body or full-body actors have been proposed by Dalal et al. [3], Eichner and Ferrari [4] and Felzenswalb et al. [7], among others. While such methods can potentially increase the coverage of actor detectors by detecting actors in profile and back views, they also suffer from the higher variability of actor appearances in such views.

Extending the work in [18], Ramanan et al.[15] demonstrate that color histogram of body appearance can be used as a strong cue to group detected faces into tracks. They show that given the unconstrained nature of the video, peo-

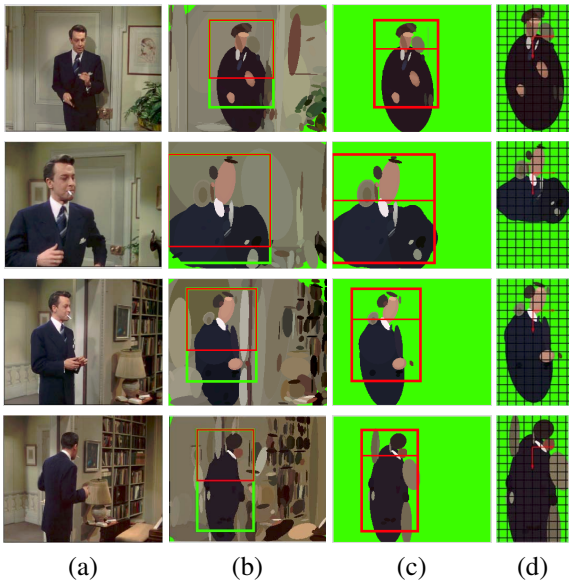


Figure 1. Illustration of training process of an actor, for each row (a) Given training frame. (b) Detected upper body and the MSCR features. The red window shows the upper body detection and green window shows the extended window to include the torso, when available. (c) The blobs chosen for training shown with the head and the torso partition. (d) The color blobs scaled and shifted to the normalized actor coordinate system which is represented by the red axis.

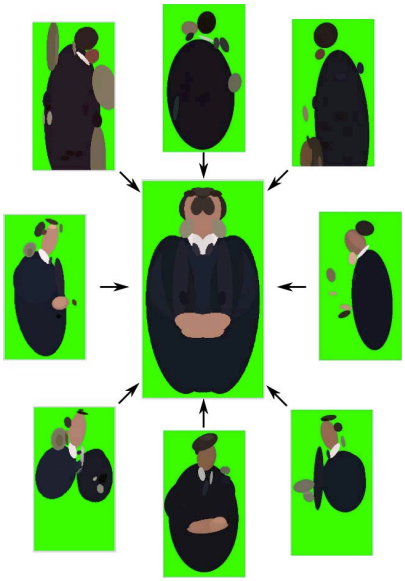


Figure 2. Training images from different viewpoints are merged to obtain an actor model.

ple can be tracked only about 50% of the time using such an approach. While the work in [18] and [15] focuses on first obtaining tracks and later classifying or hand labeling them, our method can directly perform actor specific detections on individual frames. Closer to our approach, Sivic et al. represent people in a scene with a simple pictorial structures with 3 rectangular image regions (face, hair and torso) and use that model for finding the same people in repeated shots of the same scene [19] . Starting from face detection regions, they use likely positions for the hair and torso regions and cluster them into actors. Instead, we use a training set containing front views, side views and even back views of actors, as shown in Figure 1. Our actor model is simpler since we only model two body parts (head and shoulders) but each part is an arbitrarily complex constellation model of color blobs.

Our generative model of actor's appearance is closely related to recent work in object detection and recognition [20, 8]. In the classical constellation model, image features are generated in the vicinity of detected interest points and the features are clustered using k-means, which builds a visual vocabulary of "object part appearances". Appropriate feature detectors are then trained using these clusters, which can be used to obtain a set of candidate parts from images. It is difficult to apply such models to actor appearances because interest points and their image features are typically not very stable on actors, due to fast motion and complex clothing patterns, and the density of interest points and their features can be very low. We resolve this issue by representing parts of actors with color regions rather than local features. Color regions have been shown to give good results on the problem of person re-identification [6], where the goal is to identify a person given its detection window. We extend such previous work to the more difficult problem of "re-detecting" actors where the generic detectors fail due to variations in pose and viewpoints, partial occlusions, etc. This is similar to the recently-proposed Implicit Shape Model [13, 12] which builds a star shaped structural model, where the position of each local part is only dependent on the object center. While the models in [13] are view specific, we build a single view-independent model of each actor's appearance.

3. Generative model

In this section, we introduce our generative model for the appearance of actors and describe a method for learning the model from a small number of individual keyframes or short video tracks. Our model is designed to incorporate the costume of the actor and to be robust to changes in viewpoint and pose. We make one important assumption that the actor is in an upright position and that both the head and the shoulders are visible. As a result, we model the actor with two image windows for the head and shoulders, in a normalized coordinate system with the origin at the actor's neck, and with unit size set to twice the height of the actor's eyes relative to the origin. The head region

(a) Brandon (b) Philip (c) Janet (d) Kenneth

(d) Mrs. Wilson (e) Mrs. Atwater (f) Rupert (g) Mr. Kentley

Figure 3. Appearance models for all 8 actors in the movie "Rope" [11]

extends from $(-1,-1)$ to $(1,0)$ and the shoulder region extends from $(-1,0)$ to $(1,3)$. Examples are shown in Figures 1 and 3.

More specifically, we associate with each actor a visual vocabulary of color blobs C_i described in terms of their normalized coordinates x_i, y_i, sizes s_i, colors c_i and shapes m_i, and their frequencies H_i. Color blobs above the actor's origin are labeled as "head" features and color blobs under the origin are labeled as "shoulder" features. Contrary to previous work [16, 14, 1, 9], our head and shoulder models are not based on a fixed number of body parts but on an arbitrary number of salient color regions. This allows to accommodate ample clothing not revealing the body parts, and to incorporate optional or even unusual costume elements such as glasses, hats and helmets, to name just a few.

Formally, our generative model for each actor consists in the following three steps:

1. Choose screen location and window size for the actor on the screen, using the detections in the previous frame as a prior.

2. Choose visible features C_i in the "head" and "shoulder" regions independently, each with a probability H_i

3. For all visible features C_i , generate color blob B_i from a gaussian distribution with mean C_i and covariance Σ_i, then translate and scale to the chosen screen location and size

In effect our model is an AND/OR graph with one AND node and multiple OR nodes [21]. Example of randomly generated blob images are shown in Fig 4. We learn the model parameters from image examples by computing maximal color regions in all examples, clustering those regions in x, y, s, c, m space and counting the frequency H_i of appearance for each cluster.We now describe each of those steps in more details.

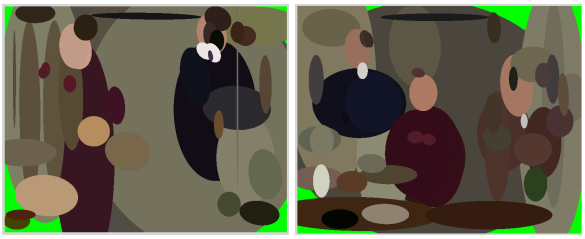
3.1. Maximally stable color regions

The maximally stable color regions (MSCR) feature is a color extension of the maximally stable extremal region (MSER) feature [10]. It is an affine covariant region detector which uses successive time steps of an agglomerative clustering of pixels to define a region. The raw moments up to order two are calculated for each detected region, which are then used to calculate the regions area, the centroid, the inertia matrix and the average color. Each region is thus represented by 9 parameters i.e. the centroid, average color and 4 raw moments representing the size and the shape of the region. An approximated ellipse is then defined over each detected region. These approximated ellipses are termed as color blobs in the later part of the text. Examples of detected MSCR features over an image are shown in Fig 1.

3.2. Clustering

We build a view-independent model of an actor's appearance by choosing a small number of front views, side views and back views for training (Fig.2). The actual choice of samples is not very important, as long as we cover the entire range of appearances of the actor (we try to keep the training set equally sampled across different views i.e. front, back and side views). Ideally a sequence of each actor performing a 360 degree turn would be sufficient to build such models. We manually draw the upper body bounding boxes for all training examples and label them with the actor's names.

We compute the MSCR features over all keyframes in the training set. We then collect the color blobs in all training windows, center and resize them, and assign them to ac-



(a) (b)

Figure 4. Example of two independent randomly generated blob images given the actor and background models. (a) Brandon and Janet. (b) Brandon, Janet and Kenneth

tors. We cluster the blobs for all actors using a constrained agglomerative clustering. For every actor in n training images we get n set of blobs $(f_1, f_2, ..., f_n)$ with varying number of blobs in each set, where each blob is represented as a 9 dimensional vector in normalized actor coordinates. An example of this process is illustrated in Fig 1. Each blob in the first set f_1 is initialized as a singleton cluster. We then compute pairwise matching between those clusters and the blobs in the next set f_2. At each step, for each cluster, we assign at most one blob (its nearest neighbor) if it is closer than a threshold. Each cluster is represented by the mean value of its blobs. Blobs not assigned to an existing cluster are assigned to their own singleton cluster. The process is repeated for all frames and for all actors. Finally, we eliminate the remaining singleton clusters. The cardinality of a cluster is the number of its occurrences in the training set. We interpret this value as a frequency, i.e. the probability of the cluster being visible in the actor. Note that the number of clusters per actor is variable. As a result, actors with more complex appearances can be represented with a larger number of clusters. The appearance models for eight different actors are shown in Fig 3.

4. Actor detection

Given the learned appearance models for a cast of actors, we now describe a framework for detecting them in any given video frame. Our framework searches for actors over a variety of scales, from foreground (larger scales) to background (smaller scales). For each actor we first perform a search space reduction using kNN-search. Then, we scan a sliding window and search for the most likely location for the actor. Then we perform a multi-actor non maxima suppression over all the detected actors. Finally, we report the best position and scale for each detected actor. The complete detection process is illustrated in Fig 5.

4.1. Search space reduction

Given an input frame first we calculate the maximally stable color regions over it as illustrated in Fig 5(b) and Fig

Algorithm 1 Detection and localization algorithm

Given actor models(C^a, Σ^a, H^a) and the image features B.
for each actor a **do**
  for each scale s **do**
    Normalize image features w.r.t scale.
    $[IDX, D7]$ = kNN-SEARCH(B,C^a,k).
    Build inverted index i.e. for each unique blob B' in the Knn refined set, store corresponding clusters in C^a and respective distances using IDX and $D7$.
    for each position (x, y) **do**
      Find blob indices J_{head} and $J_{shoulders}$
      Compute m_{ij} using blob indices and inverted indices
      $score(x, y, s, a) = \prod_k (\sum_{j \in J_k} P(B_j, m_{ij}, a))$
    end for
  end for
end for
$[x^*(a), y^*(a), s^*(a)] = \text{argmax} \sum_a (score(x, y, s, a) - t_0)$

5(c). This gives us a initial set of blobs B over which we perform a refinement step using kNN search given the actor model and the particular scale. This pre-refinement is done only based on the color, size and shape parameters. The kNN-SEARCH(B,C^a,k), calculates k nearest neighbours in B for each cluster center in C^a by performing an exhaustive search over the euclidean distances in the 7 dimensional space of appearance. It returns IDX and D_7, both $N \times k$ matrices, where N is the number of clusters in the given actor model. Each row in IDX contains the indices of the k closest neighbours in B corresponding to the k smallest distances in D_7.

This is further used to build inverted indices i.e. for each unique blob B' in the kNN refined set IDX, we store corresponding clusters in C^a and respective distances, ensuring that the distance is less then a threshold τ_1. This boosts the efficiency of the matching step in two ways. Firstly for each blob within the sliding window we only require to compare it with its corresponding entries in the inverted index table instead of doing an exhaustive search. Secondly, the distances in appearance are pre-calculated and distance in position D_2 can be directly added to calculate the full distance. An example of pre-refinement is illustrated in Fig 5(d). The figure shows the refinement for a particular scale but this is performed independently for each scale in sliding window search.

After the search space reduction, only sparse set of features remain for the matching step which leads to the speed up of the detection process in two ways, firstly many windows can be rejected without any processing using a importance term based on a binary map as shown in Fig 5(e)

(a) (b) (c) (d) (e) (f) (g)

Figure 5. Illustration of the detection process. (a) Given Actor model. (b) Input frame and obtained detection for given Actor model with the proposed method. (c) MSCR features on the input frame. (d) Refined set of blobs for the given actor at given scale, based on appearance. (e) Binary map which can be used to reject some of the windows without processing, in practice we use separate binary maps for head and torso features. (f) The features in the image which were finally matched and considered for scoring for the final detection window as shown in (b). (g) The corresponding matched cluster centers in the given actor model.

which can be efficiently implemented using integral images. Secondly, it reduces the number of matches to be performed by a large margin over a naive exhaustive matching. Notice that the number of required comparisons become independent of number of features in the image after the kNN refinement, which is done once in the beginning. Hence, the efficiency will go higher with increasing number of features. In addition to computational benefits, kNN refinement also makes the matching step much more robust by filtering out the background clutter, which is a major benefit while using low dimensional features like MSCR.

4.2. Sliding window search

We now proceed to define the detection scores for all actors at all positions and scales using a sliding window approach. Each actor detection score is based on the likelihood that the image in the sliding window was generated by the actor model using the previous frame detections as prior information. In practice we compute MSCR features in the best available scale and then shift and scale the blobs respectively while searching at different scales. Remember that the model was normalized to a fixed scale during training. During recognition, we similarly normalize the size, shape and position of blobs relative to the sliding window. This ensures that all computations are performed in reduced actor coordinates. The detection procedure is illustrated in Algorithm 1.

We represent B as the set of all blobs detected in the image and C^a as the set of cluster centers in the model for a given actor a. Given a sliding window at position (x, y) and scale s, we find all blobs centered within the sliding window and assign the blobs indices J_{head} and $J_{shoulders}$. Using these indices and a matching function m_{ij}, we define the score as:

$$score(x, y, s, a) = \prod_k (\sum_{j \in J_k} P(B_j, m_{ij}, a))$$

which is a product of parts, where each part is a sum of optional MSCR regions. Only two parts i.e. the head and the shoulder are considered in this case. The term $P(B_j, m_{ij}, a)$ is the similarity function between the model cluster C_i^a and the corresponding matched blob B_j in nine dimensional space (position, size, color and shape), which is defined as follows:

$$P(B_j, m_{ij}, a) = \sum_i H_i \cdot m_{ij} \cdot \exp\left\{-(C_i^a - B_j)^T \Sigma_i^{a\,-1} (C_i^a - B_j)\right\}$$

where C_i^a is the center for cluster i in the actor model and Σ_i^a is its covariance matrix. A distinctive feature of our detection framework is that it requires us to find a partial assignment m_{ij} between blobs in the the sliding window and clusters in the model. More precisely, we compute m_{ij} such that each blob in the sliding window is assigned to at most one cluster, each cluster is assigned to at most one blob, and the assignment maximizes the total likelihood $\prod_k (\sum_{j \in J_k} P(B_j, m_{ij}, a))$ of the matched blobs. Although potentially more computationally intensive, we have found that this method produces significantly better results than computing the average score over all possible blob-to-cluster assignments, where the same blob may be assigned to multiple clusters, and the same cluster to multiple blobs, which is prone to detection errors. A key to our algorithm is therefore a fast approximate solution to the assignment/matching problem for evaluating the detection score benefiting from a pre-refinement step.

We further benefit from the previous frame detections D_{t-1} to modify the scores as follows:

$$score(x, y, s, a) = \begin{cases} l_{1,1} \cdot score(x, y, s, a) \cdot pr_{t,t-1} & \text{if } a \in D_{t-1} \\ l_{0,1} \cdot score(x, y, s, a) & otherwise. \end{cases}$$

where, $l_{1,1}$ and $l_{1,0}$ measures the probability that the same actor is observed in consecutive frames. When the actor is not present in the previous frame, all positions in next frame are equally probable. When the actor is present in both frames, we assume the new position to be close to the previous position, within some covariance term Σ_{pos}. Empirically, we have found that the terms $l_{1,1}$, $l_{1,0}$ and Σ_{pos}

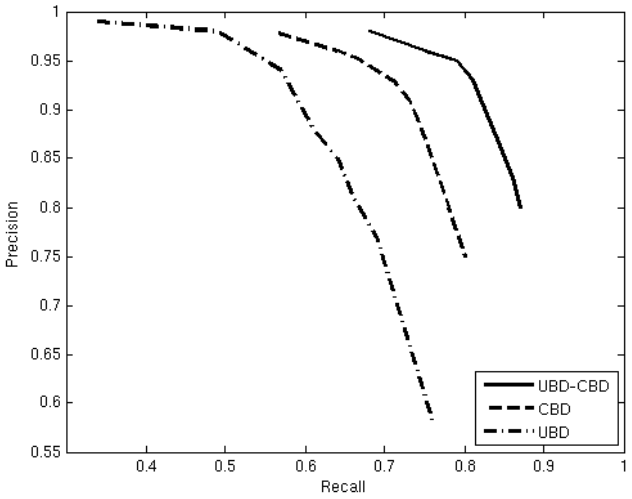


Figure 6. Comparison of results on recall and precision for actor detection using Upper body detector(UBD), Color Blob detector(CBD) and Combined method (UBD-CBD)

are in fact independent on the choice of actor or movie. The term $pr_{t,t-1}$ is defined using the covariance term as follows:

$$pr_{t,t-1} = \exp\left\{-(p_t - p_{t-1})' \Sigma_{pos}^{-1} (p_t - p_{t-1})\right\}$$

The above procedure is repeated for all actors individually and after non maximal suppression over $score(x, y, s, a)$ we get the potential detections for each actor. Benefiting from the fact that an actor can only appear once on a frame, we then search for the best possible positions $[x^*(a), y^*(a), s^*(a)]$ which maximizes the total score over all actors.

$$[x^*(a), y^*(a), s^*(a)] = \text{argmax} \sum_a (score(x, y, s, a) - t_0)$$

A threshold t_0 is used to reject all the detections with score below a threshold value.

5. Experiments

5.1. Dataset

As mentioned earlier the previous work in actor specific models have focused on obtaining tracks and then performing classification on the obtained tracks. And in datasets related to the re-identification problem, the bounding boxes are provided and the goal is only to identify the pedestrians. Our method on the other hand can perform direct detection using actor specific models, even on keyframes which is a much harder task than classifying tracks and we target the scenario where it is difficult to obtain face or upper body tracks.

Due to this reason, for a detailed evaluation of our method we propose the ROPE dataset[1] which is set of

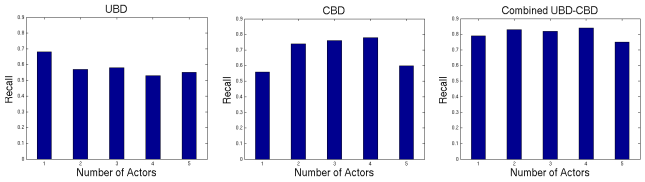


Figure 7. Comparison of recall with increasing number of actors in UBD, CBD and combined cases

keyframes at equal intervals, a frame every 10 seconds from the movie "Rope" [11] by Alfred Hitchcock. This dataset presents significant scale and viewpoint variations for each actor with presence of motion and focus blur. There is significant camera movement and constant background variations from flat regions to cluttered patterns. The lighting changes considerably during the movie and the clothing appearance of all the actors remains consistent which makes it suitable for our experiments. The movie is composed of a single shot and choosing a frame every 10 seconds gives about 443 frames. There are 8 different actors in the entire movie (except the initial victim and Alfred Hitchcock himself). All of them were considered in our experiments. The number of appearances per actor vary between 38 and 275. All 443 frames were hand labeled with the names and screen locations for all actors to serve as ground truth.

5.2. Results

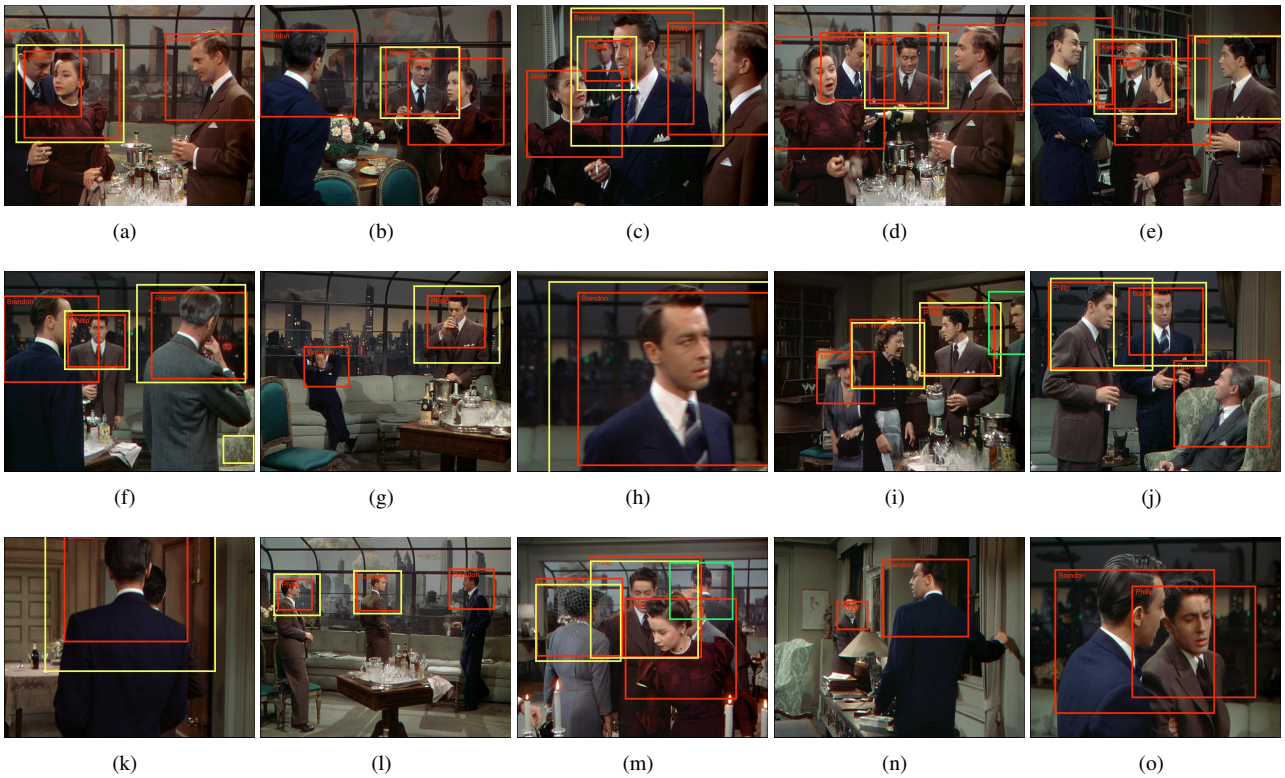
	A1	A2	A3	A4	A5	A6	A7	A8
A1	99	0	0	0	1	0	0	0
A2	0	78	0	16	0	2	3	1
A3	0	1	96	0	3	0	0	0
A4	0	25	0	74	0	0	1	0
A5	0	0	5	0	95	0	0	0
A6	0	4	0	0	0	93	0	3
A7	3	4	0	0	1	2	81	9
A8	0	0	0	0	0	6	22	72

Table 1. Actor identification results for all 8 actors in Rope dataset (percentages).

This section describes the results on the proposed dataset. We ran our detection and recognition algorithm using the built actor models, on all 443 frames and compared the results with the ground truth. Fig 6 shows the results on recall and precision of actor detection using the proposed method (CBD) and it is compared with the state of the art Upper Body Detector (UBD)[2] and the combined case where we merged the detections obtained from both the methods individually. Results demonstrate an increase in recall from about 57 percent in UBD to 70 percent in proposed Color blob detector (CBD) to about 81 percent in combined approach for a similar precision. Thus, the coverage is significantly improved keeping the same false positive rate. In

[1] http://imagine.inrialpes.fr/people/vgandhi/CVPR_2013/

[2] http://www.vision.ee.ethz.ch/~calvin/calvin_upperbody_detector/



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

(k) (l) (m) (n) (o)

Figure 8. Some detection results from Rope dataset using proposed method CBD (in red) with recognized actor names on top left, UBD (in yellow) and not detected by both (in green). Notice how our method is able to detect and identify the actors in the presence of multiple actors with partial occlusions [e.g. (a), (d), (e), (o)], varying viewpoint and posture [e.g. (f), (m), (g), (j)], varying illumination [e.g. (b), (l)], motion blur [e.g. (h)] etc.

(a) (b) (c) (d) (e)

Figure 9. Few examples where our method wrongly detects or fails to detect the actors, the typical failure cases are due to shadows [e.g. (e)], merging of foreground blobs with the background due to low illumination [e.g. (a)] or heavy blur [e.g. (b)], confusing the hands as the head [e.g. (d)] and large occlusions [e.g. (c)]. This also demonstrates the difficulty of the proposed dataset.

Fig 7 we plot recall rates for both UBD and CBD and combined case, with different number of actors present in the frame and it shows that the proposed method gives consistent results with varying number of actors.

Recognition results on the detected actors are presented in Table 5.2. As can be seen, our method not only increases the average recall rate for all actors, but also correctly names all actors with an average precision of 89 percent despite the large number of back views and partial occlusions.

Some of the example detections results are shown in Fig 8 and they demonstrate how our method performs well even with severe cases of occlusions, viewpoint scale and pose variations in a multi actor scenario. Fig 9 shows some failure cases. Most failure cases are caused by insufficient illumination or severe occlusions. In 9(a) the blobs in the torso region of the undetected actor gets merged with the background, heavy blur causes mis-detection in 9(b), torso is largely occluded in 9(c). In fourth instance 9(d), the hand gets detected as the head and blobs below as torso, leading to a false detection. Note that there is very little temporal coherence between frames in our dataset. As a result, we cannot rely on context to resolve such problems. Detecting actors at a finer temporal scale (every second or every frame) may help alleviate the problem but that requires a

stronger model of temporal coherence, taking into account the inter-frame motion of actors, the temporal coherence of the blobs, and the probability of visual events such as entrances and exits. This is left for future work.

6. Conclusions

We have presented a generative appearance model for detecting and naming actors in movies that can be learned from a small number of training examples. We have shown that low dimensional features like MSCR that were previously used for actor re-identification can also support actor detection, even in difficult multiple actors scenarios. Results show significant increase in coverage (recall) for actor detection maintaining high precision. To our knowledge, this is the first time that a generative appearance model is demonstrated on the task of detecting and recognizing actors from arbitrary viewpoints. Our method also appears to be a good candidate for tracking multiple actors constantly changing viewpoints and occluding each other in long video sequences such as "Rope", which include important application scenarios, such as unedited, raw video footage and recordings of live performances. We also plan to investigate weakly supervised methods by extracting actor labels from temporally aligned movie scripts [17, 2].
One obvious limitation of our method is that it only handles cases where the appearance of actors does not change much over time. In future work, we are planning to investigate extensions with mutually-exclusive appearances per actor, so that actors can change their appearances and costumes over time. Because each appearance requires so few training examples to be learned, we believe this extension is in fact possible.

7. Acknowledgment

This work was supported by Region Rhone Alpes (project Scenoptique), ANR Corpus (project Spectaclenlignes) and ERC advanced grant EXPRESSIVE.

References

[1] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *CVPR*, 2009.

[2] T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar. Movie/script: Alignment and parsing of video and text transcription. In *ECCV*, 2008.

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

[4] M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *BMVC*, 2009.

[5] M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is... Buffy" – automatic naming of characters in TV video. In *BMVC*, 2006.

[6] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. Person re-identification by symmetry-driven accumulation of local features. In *CVPR*, pages 2360–2367, 2010.

[7] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627 –1645, sept. 2010.

[8] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, 2005.

[9] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Pose search: Retrieving people using their pose. In *CVPR*, 2009.

[10] P.-E. Forssen. Maximally stable colour regions for recognition and matching. In *CVPR*, 2007.

[11] A. Hitchock. The rope, 1948.

[12] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 17–32, 2004.

[13] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, 2008.

[14] A. S. Micilotta, E. jon Ong, and R. Bowden. Real-time Upper Body Detection and 3D Pose Estimation in Monoscopic Images. In *ECCV*, 2006.

[15] D. Ramanan, S. Baker, and S. Kakade. Leveraging archival video for building face datasets. In *ICCV*, 2007.

[16] R. Ronfard, C. Schmid, and W. Triggs. Learning to parse pictures of people. In *ECCV*, 2002.

[17] R. Ronfard and T. Tran-Thuong. A framework for aligning and indexing movies with their script. In *ICME*, 2003.

[18] J. Sivic, M. Everingham, and A. Zisserman. "Who are you?" – learning person specific classifiers from video. In *CVPR*, 2009.

[19] J. Sivic, C. L. Zitnick, and R. Szeliski. Finding people in repeated shots of the same scene. In *BMVC*, 2006.

[20] M. Weber, W. Einhäuser, M. Welling, and P. Perona. Viewpoint-invariant learning and detection of human heads. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[21] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. Yuille. Max margin and/or graph learning for parsing the human body. In *CVPR*, 2008.