Towards Unlocking Web Video: Automatic People Tracking and Clustering

Alex Holub*, Pierre Moreels*, Atiq Islam*, Andrei Makhanov*, Rui Yang*
Ooyala Inc, 800 W. El Camino Real, Suite 350, Mountain View, CA 94040
*All authors contributed equally to this work

Abstract

This paper describes a system for automatically extracting meta-information on people from videos on the web. The system contains multiple modules which automatically track people, including both faces and bodies, and clusters the people into distinct groups. We present new technology and significantly modify existing algorithms for body-detection, shot-detection and grouping, tracking, and track-clustering within our system. The system was designed to work effectivty on web content, and thus exhibits robust tracking and clustering behavior over a broad spectrum of professional and semi-professional video content. In order to quantify and evaulate our system we created a large ground-truth data-set of people within video. Finally, we provide actual video examples of our algorithm and find that the results are quite strong over a broad range of content.

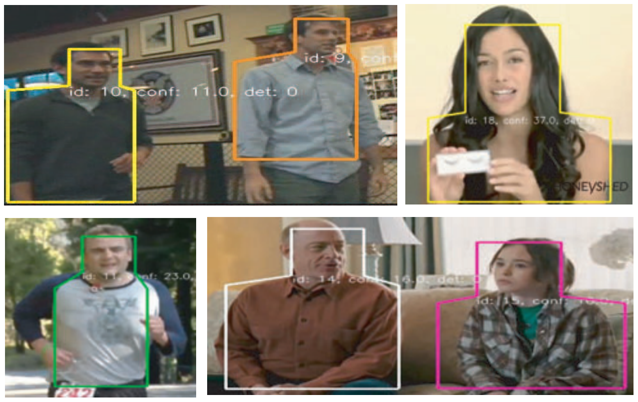

Figure 1. Examples of automatically detected people in both professional and semi-professional content on the web. Our system automatically finds and clusters images of similar people within video. The outlines shown correspond to both the body and face of individuals which were found automatically. The 'id' keyword and color of the outlines corresponds to the cluster which the person was assigned to within this video.

1. Introduction

Automatically extracting meaning from rich media content, such as images and video, on the web remains one of the great challenges for computer vision. The use of media content online is becoming ever more prevalent, further necessitating compelling solutions. Successful algorithms could enable many applications including inter and intra-media search, clickable and interactive content, and more effective monetization methods based on the content of media. However, both the difficulty of the problems as well as the infancy of the field, have resulted in relatively little research and progress in the area. Notable exceptions include [5, 15] and Google Image Search [6].

There are many challenges when working with web content compared to data-sets which have been collected and filtered for evaluating computer vision algorithms. In general, the data encountered on the web exhibits a large degree of variability when compared with data-sets which are more prevalent within the computer vision community. The Caltech101/256 [7] are good examples of data-sets which do not exhibit a high degree of variability yet are frequently used to assess object recognition performance (see [11] for a good discussion surrounding data-set issues in computer vision). Web variability includes lighting, pose, image quality, and image resolution.

This paper describes a complete system for automatically parsing and extracting meta-information from online video. The system focused on people, both because solid technology exists for locating some aspects of people (for instance faces [16]) and also because a large percentage of video encountered on the web contains people within it. Our system focuses not only detecting people, but taking steps towards recognizing their identity within video content by automatically grouping together frames containing images of the same person.

We chose to focus on clustering together images of the same person, rather than the more traditional recognition problem, due to the variability of the content we encountered on the web. Quite simply, no scalable solution exists for recognition of arbitary people in web content. In addition, even if a solution did exist the identity of individuals in video content is often not known. However, effectively clustering together images of the same person within a video, makes the identity assignment process easy. One label assignment can be propagated to every occurance of the individual within the video if accurate people-clusters have been identified.

Since our system relies on clustering together images of the same person within a video, we chose to utilize a frontal face detector to initiate our automatic tracking process in order to have somewhat aligned features to compare. Solid research and implementation exist for frontal and near-frontal face detection, including [16, 14, 13] resulting in good precision and recall.

There has been some work automatically tracking and extracting information about people. Perhaps most interesting is the recen work from Everingham et al. [3] which demonstrates the automatic identification of speakers in video using the captions of the video along with face detections. Although intriguing work, we note that captions are not easily available for most professional and semi-professional content on the web.

The paper is organized as follows. We first describe the various components of our system. Then we describe the large ground-truth data-set collected for evaluation and the results of evaluating our system on that data.

2. System Overview

Our system is composed of numerous modules described below. The modules can be broadly broken into the following categories: (A) Detection modules which detect faces, bodies, and shot-boundaries. (B) Tracking modules which act to both create coherent 'face tracks' and increase the recall and precision of the raw face detector. (C) Clustering modules which group together the 'face tracks' to form homogenous groups of characters from a video and group together similar shots.

2.1. Object Detection: Faces

Consider the set of f^k frames within a video V, $V = f^1, f^2 ...$. Our face-detection module runs the Viola and Jones [16] face-detector on every frame f^k of the video. We changed the parameters of the detector to increase the overall recall of the detector, i.e. we generated more detections for each image than typical parameter settings generate. In addition we ran a detector trained on both frontal and side-views of faces. The frontal face-detector was superior in both recall and precision to the side-view face-detector. The frontal and profile detector often fired in similar regions of images. If the overlap betwen detections was greater that 40%, we combined the detections by keeping only the frontal detection and disregarding the profile detections. We chose to keep the frontal detections as they were found empirically to be more accurate in describing the true location of a face. We use tracking to increase the precision as described in the sections below and found that, in the context of our complete face-tracking system, increasing face-detector-recall increased the overall recall and performance of the system significantly.

2.2. Object and Image Representation

Images and regions in images are represented by color histograms in HSV color space. Each color channel is divided in 16 bins. Separate histograms are computed for the region of interest, in the H, S, and V channels. These 3 histograms are concatenated, to form our representation of images or regions. Prior to concatenation, each histogram is smoothed by a low-pass filter in order to reduce boundary issues caused by discretizing into histogram bins.

In contrast with a 'straight' representation in HSV space, this representation is significantly less sparse, as we are dealing with a $16 \times 3 = 48$-dimensional space instead of a $16^3 = 4096$-dimensional space. Decreased sparsity helps when matching regions representing the same object but subject to different lighting conditions. One can observe that concatenated histograms do not define a proper probability density as they sum to 3, this can easily be corrected by normalizing all representation vectors by 3.

While easy to compute and quite efficient, such a histogram representation has limitations and is easy to fool since it does not incorporate any geometric structure. For example, shuffling sub-regions of the object of interest will not modify the histograms. This 'shuffled regions' case can occur when the object of interest is part of a texture, like a tree in a forest or a cow in a herd. In order to enrich the representation with some geometric information, regions are divided into four quadrants. Histograms are computed independently in each quadrant, then concatenated to form our final representation.

2.3. Tracking Algorithm

Our tracking system is extremely simple - it consists of template matching at the nodes of a grid, and selects the candidate location that provides the best match. Starting from the reference position of the object at time t, at $t + 1$ we compare it to the histograms obtained at shifted positions along a grid, as well as scaled and stretched outlines. The grid density varies from 2 to 20 pixels, with the highest density about the reference position from time t.

Let m_t the region tracked at time t. Our template at time t incorporates a component that relates to the 'ground truth model' m_0 at time $t = 0$, and a component that expresses the temporal evolution:

$$m_t = \alpha \cdot m_0 + (1 - \alpha) \cdot m_{t-1} \quad (1)$$

We varied α and obtained the best tracking results for $\alpha = 0.7$. Low values of α lead to drift, while values of α too close to 1 are too sensitive to variations in pose or lighting conditions.

The distance used to compare representation vectors is an important component of the tracking system. We experimented with the Bhattacharya distance, the Kullback-Leibler divergence, the Euclidean distance and Histogram Intersection. The most robust tracking results were obtained using Histogram Intersection.

Efficiency of the tracking algorithm. The tracking system described above seems to be significantly more computation intensive than, for example, the 'mean-shift' algorithm from Comaniciu et al. [1]. The main computation bottleneck is the computation of histograms. In order to compute histograms quickly, we use 'integral histograms', the multi-dimensional equivalent of the classical integral images [12]. Thus, computing a single histogram requires only 3 additions/subtractions for each histogram channel. The tracking system, implemented in C++, runs at about 20 frames/second on DVD-quality sequences (frame resolution 720x480 pixels)

Evaluation of the tracking algorithm. In order to tune the various parameters of the tracking algorithm mentioned above, 40 video sequences of moving objects were collected from two different sitcoms. These video sequences are available upon request. They range from easy-to-track objects (guitar hanging on a white wall while the camera is panning), to difficult objects (cigarette moving quickly in various directions). Ground truth was defined manually by outlining boxes around the object of interest every four frames. The accuracy of the tracking system was measured by computing the overlap between the tracked box and the ground truth box, using the definition of overlap used in the Pascal challenge [4]:

$$overlap(B_1, B_2) = \frac{area(B_1 \cap B_2)}{area(B_1 \cup B_2)} \quad (2)$$

where B_1 and B_2 are the two outlines to be compared. Tracks are reinitialized whenever their overlap with ground truth is lower than the arbitrary value 0.4. This replicates the realistic scenario with a user monitoring the tracking system. Whenever the match between the outline found by the trackin system and the ground truth becomes poor, the user reinitializes the tracker.

2.4. Detecting and Groupings Shots

Most video content consists of series of *shots* which make up a *scene*. Each shot can be defined as the video frames between two different camera angles. In other words, a shot is a consistent view of a video scene in which the camera used to view the scene does not change. Often a sequence of shots contains the same camera angle multiple times within the sequence. These shots contain almost identical objects within them. For instance, consider a conversation between two actors in which the camera toggles between the two actors depending on who is speaking. Figure 2.4 illustrates the concept of shots. We describe an algorithm which both detects shot boundaries as well as groups consistent shots together to form scenes.

Consider a video j which consists of a set of consecutive frames: $V_j = f^1, f^2 ... f^k$. In order to determine whether a shot boundary is present we consider a function $\mathcal{S}$ which returns a boolean value, $\mathcal{S}(f^k, f^{k+1}) \in \{0, 1\}$, depending on whether or not there is a shot boundary between any two frames. By stepping through all the frames within a video we obtain a boolean vector with non-zero values indicating a shot detection.

Next we consider how to compare two consecutive images in order to assess whether a shot boundary is present. Each image is initially divided into a $m \times n$ grid, resulting in a total of $m \times n$ different bins. We consider corresponding bins from consecutive images and determine how different they are. Consider the function $\mathcal{T}$, where $T(f^k, f^{k+1}) = \sum_{m,n} \mathcal{D}(f^k_{m,n}, f^{k+1}_{m,n}) > T$ and $\mathcal{D}$ is the histogram difference for a particular color channel. We are counting the number of grid entries whose difference is above a patricular threshold. If the percentage of different bins is too large, we deem the two frames to be different and declare a shot boundary. In practice we divded our image into 4×4 bins for a total of 16 unique areas and declare a shot-boundary if more than 6 of these areas are deemed different ($\mathcal{D} > T$).

Using the algorithm just described we can step through an arbitary video V_j and find all the shot boundaries. We would also like to determine which shots are the same. We leverage the shot-detection algorithm described above in order to group similar shots together. Consider any two shot boundaries and let us demark the indices of the frames which contain the shot boundaries as f^h and f^j. Now consider the 5 frames at the end of f^h, namely $f^{h-1} ... f^{h-5}$ and the 5 frames after f^j, namely $f^{j+1} ... f^{j+5}$. For every pair of these frames we consider whether $\mathcal{S} == 1$, thereby indicating that there is a shot boundary. If none of the comparisons yields a shot boundary then we declare the two shots to be the same and we group them within the same cluster. We use the word shot cluster and scene interchangeably. A scene is composed of a set of similar shots.

Note that the threshold on histogram similarity is a tradeoff. If it is chosen too low, separate shots will never be connected as there is usually some movement of the actors or the camera between shots. If it is set too high, irrelevant shots will be clustered together.

The result of running our shot detection and shot grouping algorithms are a set of shots, where each shot belongs to a particular scene. For typical content each scene will continue multiple shots. By leveraging scenes as groups of shots we are able to drastically increase our effectiveness in creating clustered groups of people within a video.

2.5. Creating Face-Tracks

Our system benefits heavily from the use of video as we are able to utilize the temporal continuity between frames. Consider a particular face-detection d_i^k in frame f^k. Our goal is to grow this detection into a homogenous 'face track' which follows the face of a particular actor over consecutive frames of within a video.

Consider again the detection d_i^k. We use the tracking algorithm described in Section 2.3 to predict the location of the track in frame f^{k+1}. Now consider the set of n face-detections in frame f_n^{k+1}. If any of those n detections is close to the location predicted by tracking, we use that detection as the location of the track in frame f^{k+1}. We continue in this manner both forwards and backwards in frame indices, thereby building up a homogenous 'object track' which specifies the location of an object over time. Figure 3 illustrates the procedure of choosing between a predicted location from tracking and a face-detection. Figure 4 draws a schemtic diagram containing more details of the face tracking procedure.

Why do we favor the face-detection, when available, over the predicted location by tracking? All tracking algorithms suffer from drift unless they are re-initialized. We use the face-detections as a re-initialization of the tracking algorithm by noting that they tend to be more reliable indicators of the true location of the face than the predicted locations from tracking.

Track termination. There are two possible criteria we use to terminate a track. (A) Consider a face-track whose outline in frame k is denoted by i, d_i^k. If the predicted region from tracking is below a specified threshold and there is no face-detection near the predicted region (as defined above), the track is deamed terminated. (B) If the face-track continues to grow for a long period of time without encountering a face-detection, the track is deamed lost. We found this to be useful to avoid drift with tracking. For instance the face-track can grow over many frames by tracking an innapropriate object. By enforcing that a face-track periodically contains a face-detection, we are able to obtain more homogenous tracks with little drift.

Track Collisions. Consider that two tracks may cross one another. This happens when one person walks in front of another. We split each track into two separate tracks at the point of collision. This results in 4 unique tracks being created. In Section 2.6 we show how the tracks are grouped together again as a post-processing step using a clustering algorithm.

Tracking across shot boundaries. In section 2.4 we described how we group together similar shots within a video. This allows us to do tracking across 'shot-jumps', i.e. tracking continues after the camera moves away and comes back to a similar shot. For video content that switches regularly between several cameras (e.g. sitcoms, talk shows), this drastically extends the length of tracks.

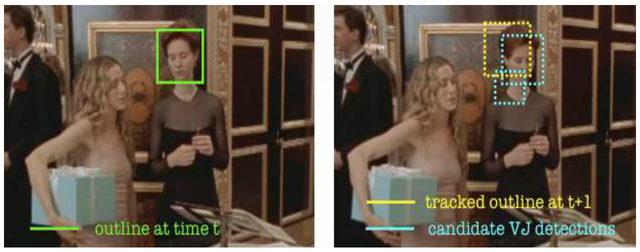

Figure 3. Overview of Face-Auto-Tracking. This figures shows the key-step within our tracking algorithm: the decision of where to place an outline in a succeeding frame. The outline from time t is tracked at time $t + 1$. If the face detector generates results at time $t + 1$, the detection leading to the best overlap with the tracked outline is selected as outline for frame $t + 1$, provided this overlap is above a pre-defined threshold. If no detection meets this threshold, the tracking result is selected as the new outline.

Filtering resulting tracks. One final post-processing consists of removing face-tracks which did not incorporate enough face detections. We found that this reduced our false positive rate significantly. We require at least 5 detections within a track. Besides, for tracks over 25 frames, we require that at least 10% of the frames making up the track contain a face-detection.

The result of the tracking as described above is a set of face-tracks, where each face-track contains a homogenous set of faces corresponding to a particular individual over consectuive frames. In the process we have removed many of the spurious face-detections found as they were never included witin a face-track.

2.6. Track Clustering

This section describes our procedure for clustering together tracks of the same person. The result of the face-tracking performned in Section 2.5 is a set of tracks of people within a piece of video content. In this section we show how to group those tracks into homogenous clusters where each cluster is a unique individual. We first compute a similarity matrix between tracks, then use hierarchical agglomertative clustering to cluster the tracks.

Distance between tracks. The distance between two tracks is taken as the minimum pairwise distance between faces belonging to these tracks.

Distance between faces. Faces are first normalized in order to align their features. Face features are detected using [9], and each face is rotated and scaled so that the corners of the eyes have a constant position. Note that this rectification preserves the shape of the face: a long or round

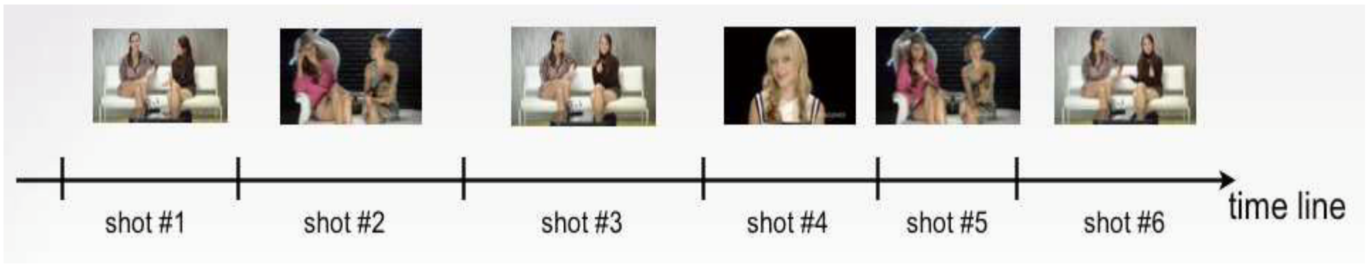

Figure 2. Example of shot clustering within a video. A video is composed of consecutive shots. A representative frame for each shot is shown above the timeline. Notice that shots 1,3,6 are all similar and shots 2,5 are similar as well. Our shot grouping algorithm, as described in Section 2.4 should group together these shots.

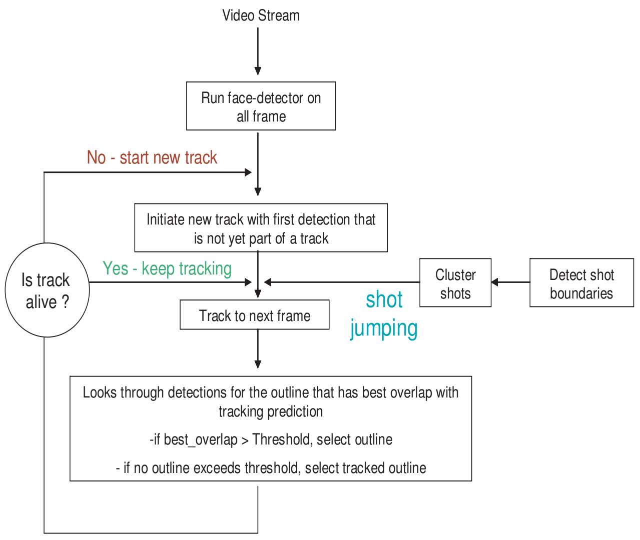
Video Stream
Run face-detector on all frame
No - start new track
Initiate new track with first detection that is not yet part of a track
Is track alive ?
Yes - keep tracking
Track to next frame
shot jumping
Cluster shots
Detect shot boundaries
Looks through detections for the outline that has best overlap with tracking prediction
-if best_overlap > Threshold, select outline
- if no outline exceeds threshold, select tracked outline

Figure 4. Schematic diagram describing how the face-tracking module creates tracks. A face-track is considered 'alive' if it was not lost during tracking and contains enough face-detections. Body-detection and Face-Track-Clustering are performed after the set of face-tracks has been generated.

face stays long and round. Rectified faces are also normalized by the sum of their squared pixel values in order to reduce the influence of lighting conditions. The distance between two rectified and normalized faces, is simply taken as the Euclidean distance between these images.

Agglomerative clustering. The distance described above allows us to compute a similarity matrix between tracks. Hierarchical agglomerative clustering [8] is well suited to form clusters using this distance matrix. Complete-link clustering, where the similarity between two clusters is defined as the similarity betwene their most dissimilar elements, provided the best results in our experiments (as opposed to single-link, group-average and centroid clustering).

A delicate parameter is the threshold that determines how close tracks need to be in order to be clustered together, i.e. when clustering stops. This threshold was determined empirically, as a fixed percentile of the sorted values in the tracks similarity table.

2.7. Body Detection

This section describes how to attach a body outline to each frame within a face-track. The extension of the face outline to the body results in a large interactive region for clickable applications. We note that having the face detection as a prior for the location of the body drastically reduces the possible locations of the body within a particular frame. In addition, the knowledge that a face exists at a particular location is a strong indication that a body exists below it. We utilize this information in the algorithm described below.

The problem we are addressing, namely detecting a body below the face is ill-posed and underconstrained: a person can be wearing anything and the body can often be in a position that is not directly below the face-track. We make use of two implicit priors in our algorithm below: (1) We assume that the body is composed of homogenous regions which can be segmented using traditional segmentation methods. (2) We assume that the body is in some area below a detected face

We begin by taking a region of interest ROI_{body} below the face that is multiple of 3 to 4 times the width and height of the face outline within the face-track. The region of interest is large enough to account for varying body sizes, poses, and the possibility of the body not lying directly below the face (as occurs, for instance, when a person is leaning forward).

We segment ROI_{body} into regions ρ_k of pixels that are similar in color using the Adaptive Clustering Algorithm (ACA) [10]. This algorithm essentially begins with the popular K-Means clustering algorithm and extends it to incorporate pixel location in addition to color.

We consider a subregion of ROI_{body} that is the same width as the face and $1/2$ the height of ROI_{body} that is at the center of ROI_{body}. We will call this region ROI_{hist}

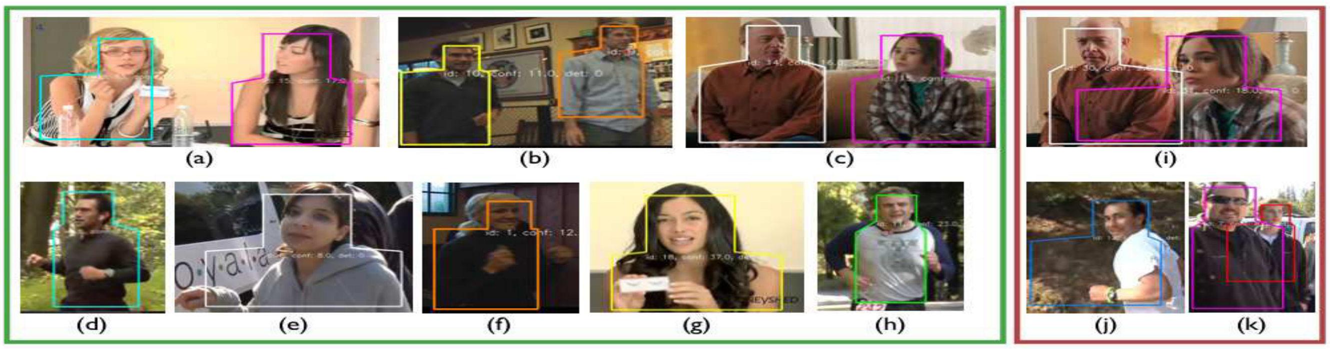

Figure 5. Typical Body-Detection Performance. Outline around each person indicate the tracked face with a body detection attached below it. Examples a-h show strong detection results which exhibit a high degree of overlap with the actual body. Examples i-k show detections which are not as accurate. In particular the algorithm has some difficulty when two bodies are overlapping (i,k) as well as when the face is not in a frontal pose (j).

because we take the histogram of the ρ_k that fall within this subregion. We define the colors C_{ρ_0} and C_{ρ_1} as two colors that occupy the most area within ROI_{hist}. Let P_{C_0} and P_{C_1} be the sets of pixels in ROI_{body} who's R, G, and B values are within 25 of those of either C_{ρ_0} or C_{ρ_1}. Furthermore, we define the ratio

$$\alpha = \frac{|P_{C_0}|}{|P_{C_0}| + |P_{C_1}|} \tag{3}$$

as the relative importance between the top two representative colors. Because these colors were found within ROI_{hist} which is a region just below the face, these two colors are assumed to represent the two dominant colors of the upper torso.

Our final task is to find the largest rectangle in ROI_{body} that maximizes a scoring function S.

$$\begin{aligned} S_{\{B_w,B_h,B_x,B_y\}} = \alpha|P_{C_0}| + (1-\alpha)|P_{C_1}| \\ -\gamma(pix \notin \{P_{C_0} \vee P_{C_1}\}) \end{aligned} \tag{4}$$

Here, a good value for γ was emperically determined to be 1.4. B_w and B_h are the candidate rectangles' width and height, while B_x and B_y are the (x,y) positions of the candidate rectangles' centers. Maximizing S in essence finds the largest rectangle that has the highest density of pixels that belong to either P_{C_0} or P_{C_1}, maintaining their relative importance and the the fewest of the other pixels.

Evaluation. To find the optimal parameters in the many phases of the above algorithm (including parameters passed into the ACA algorithm) , we collected a dataset of about $4,000$ frames in which bodies were labeled with rectagles. We ran the body detection algorithm on the data set using several hundred different permutations of the parameters to find the parameters that made it match closest to the human annotations. Using the overlap equation defined above, the best parameters yielded an overlap of 65% with the human annotations.

Using the algorithm described above we are able to accurately identify the position of the body given a face-detection as a prior. Figure 5 shows some examples of detected bodies and the results are, in general, quite strong.

3. Data-Set

One of the main challenges in constructing a complete computer vision system with the goal of deployment in a production environment, is the proliferation of parameters and the difficulty in evaluating the performance of the system as a function of those parameters. In order to test different system-level parameter settings we constructed a testing paradigm which allowed us to quickly iterate and assess performance as we varied parameter values. Such a testing paradigm also allowed us to evaluate the relative importance of each parameter by judging the effect of the parameters on system performance.

In designing our test content suite, we chose video content which exhibited different statistical properties. Different dimensions which we were concerned with included: (a) the length of the content, (b) the type of content as in drama, action, or comedy, (c) the date when the content was created - for instance 'Breakfast at Tiffanies', released in 1961, contains long scenes with few cuts when compared with some modern content which exhibits jerky camera motions and quick shot cutting, (d) the production-quality of the content, for instance professional or semi-professionally made content. The different videos for which we collected ground-truth data are shown in the first column of Table 1.

In collecting the ground-truth data-set we employed the following methodology. For every video, we marked faces in every $5th$ frame. Since consecutive frames are often quite similar in their visual appearance, it was not necessary to collect information on every frame within the video. We marked every face within a frame with two points in the upper right of the face and lower left of the face. We distinguished between faces which were frontal, profile, or 'difficult'. Our ground-truth annotations thus contain the position, height, width, and pose of the faces. Table 1 tabulates the various videos we collected data on as well as the percentage of the different face-types which were found.

4. System Evaluation

We present results of evaluating the performance of our system over the content listed in Table 1 in Table 2. In general our system had very high precision at the face-track level: most tracks created (in the $95\%+$ range) were actually face-tracks. In addition it was very rare to observe our face-tracks switching identity or terminating while the actor was still visible. The recall varied more across our different content. In particular, our system tended to perform better with sitcoms and comedies when compared with action movies which exhibit quick-scence cutting, jerky camera-movements, and unusual camera view-points. Consider that if the view-points is consistently changing from one scene to another we cannot exploit our ability to track across shot boundaries. In addition, face-track clustering becomes more difficult as the faces exhibit more variability in lighting and pose across different camera angles. Notably, our system should be robust to the latter type of content as there is an increasing proliferation of 'fast' content on the web and in the media space. The track length for the face-tracks tended to be lower for short-form content such as trailers as this type of content tends to have frequent shot changes thus breaking a face-track every 30-40 frames.

In additon to the empirical evaluation of the system we invite the reader to view the actual outlines, tracks, and cluster obtained with our system on video content in the supplementary material for this paper. The supplementary material contains numerous videos of actal face and body tracks. These videos, and more, will be made available on the web.

The most time-consuming steps of the system are the modules which run the face-detector on all frames as well as the module which determines the shot-boundaries and grouping of similar shots. These modules are easily parallelized within a MapReduce framework [2], and the resulting system runs at approximately a factor of 2 times the length of the video. Shorter videos tend to exhibit less gain as we have some non-linear dependencies on the video-length.

5. Conclusion

In this paper we showed a complete system which uses real video data from the web which is able to automatically detect, track, and cluster together images of people in video. We would like to stress that the nature of media content on the web is such that our system must be robust to many different types of content. We have performed extensive evaluation, both empirical and subjective, on the system with different types of content and found it to yield quality results over large sets of video.

Future avenues of exploration include increasing the robustness of tracking by leveraging the body detections during tracking, making more use of context, and extending the system to objects other than people. Note that the latter is especially challenging given both the relatively poor performance of arbitrary object-detectors. In addition, most objects do not appear universally in web video (the notable esception being people and perhaps cars), making the applications of an object-based system more restrictive.

References

[1] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. *CVPR*, 2:142–149, 2000. 3

[2] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004. 7

[3] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy – automatic naming of characters in tv video. *Proceedings of the British Machine Vision Conference*, 2006. 2

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html. 3

[5] R. Fergus, A. Zisserman, and P. Perona. Learning object categories from google's image search. *ICCV*, 2:1816–1823, 2005. 1

[6] Google. Google image search. http://images.google.com/. 1

[7] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. 1

[8] C. Manning, P. Raghavan, and H. Schutze. Introduction to information retrieval. 2008. 5

[9] S. Milborrow and F. Nicolls. Locating facial features with an extended active shape model. *ECCV*, pages 1746–1759, 2008. 4

[10] T. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, 40:901–914, 1992. 5

[11] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, C. K. I. Williams, J. Zhang, and A. Zisserman. Dataset issues in object recognition. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *LNCS*, pages 29–48. Springer, 2006. 1

[12] F. Porikli. Integral histogram: A fast way to extract histograms. *CVPR*, 1:829–836, 2005. 3

Video	Total Num Faces	Num Frontal Faces	Num Profile Faces	Num Hard Faces
The Smiths	1139	639	330	170
Sophia	465	376	35	54
Trailer: Ocean13	1316	773	358	185
Trailer: No Country For...	538	472	28	38
Models on Catwalk	949	809	24	115
Trailer: Love in the Time of ...	424	226	68	130
Ground Hog Day	1848	1006	809	27
Trailer: Golden Compass	558	425	36	97
Breakfast at Tiffanies	1518	526	830	162
Friends	2603	1558	701	340

Table 1. Ground Truth Data. We collected ground truth data on a wide variety of video content ranging from short 2 minute 'MTV' quality content to sitcoms to movies filmed 5 decades ago. Columns 1: The name of the video. Column 2: Total number of faces found in the video. Column 3: Number of frontal faces. Column 4: Number of profile faces. A detection is considered a profile face if it is more than 45 degrees out of the frontal plane. Column 5: Number of faces which are hard to recognize due to lighting, face-size, occlusions, and resolution. Note that the sum of Columns 3-5 should equal the value in Column 2.

Video	Faces Precision	Faces Recall	% True Tracks	Track Length
The Smiths	0.75	0.24	0.52	33
Sophia	0.79	0.55	0.90	55
Trailer: Ocean13	0.91	0.48	0.92	41
Trailer: No Country For...	0.75	0.43	0.18	33
Models on Catwalk	0.87	0.95	0.34	31
Trailer: Love in the Time of ...	0.99	0.36	1.0	22
Ground Hog Day	0.86	0.67	0.90	151
Trailer: Golden Compass	0.73	0.60	0.72	25
Breakfast at Tiffanies	0.64	0.48	0.74	134
Friends	0.80	0.54	0.76	72

Table 2. Results from Evaluating our system on the ground-truth data-set. Column 1: The name of the video evaluated. Columns 2: Precision of face recall. We measure the overlap between the position of faces within a face-track and the ground-truth data collected in Table 1. If the overlap for any face is over 40% we deem the ground-truth face to have been detected by one of our face-tracks. The precision is the actual number of detections within a ground-truth frame divided by the total number of detections in the frame. A higher precision is better. By and large our tracks contained actual faces thus leading to a high value for precision. Column 3: The recall measures our ability to find all the faces marked in our ground-truth set. It is measured as the number of overlapping detections divided by the total number of ground truth faces found. A higher percentage is better as it indicates that more of the ground-truth faces were contained within the face-tracks. In Columns 4-5 we asses performance on the level of tracks, rather than on the level of individual detections. We consider a track to be legitimate, i.e. tracking a face, if there are at least 3 ground-truth faces within the track. Column 4: Of the tracks found by our system, what percentage of those tracks were actually tracks (contained enough ground-truth faces within it to be deemed a track). Higher is better. Ideally we would like all found tracks to be of faces. Note that the numbers are somewhat deceptive. Some detected face-tracks deemed, by our criteria, not to be valid tracks, were indeed of faces when we inspected the results visually. This discrepency is due both to imperfections in the criteria used to evaluate whether a track was an actual face-tracks as well imperfections within our ground-truth data-set: not all faces were consistently marked within frames. However, the metric was still quite useful in determining the *relative* performance of our system across different parameter settings. In practice, rarely observed more that 3% false-positive face-tracks. Column 5: The average number of frames over which a face-track existed. Larger indicates a longer track. Long form content like 'Ground Hog Day' and 'Breakfast at Tiffanies' contained quite large tracks. In patricular, 'Breakfast at Tiffanies' has long drawn-out scenes due to the filming style of the day. Many of the trailers have quite short tracks do to the quit context switching within them.

[13] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. *CVPR*, 2004. 2

[14] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. *CVPR*, pages 1746–1759, 2000. 2

[15] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large data-set for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1958–1970, 2008. 1

[16] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2001. 1, 2