Total Cluster: A person agnostic clustering method for broadcast videos

Makarand Tapaswi[1], Omkar M. Parkhi[2], Esa Rahtu[3]
Eric Sommerlade[2], Rainer Stiefelhagen[1], Andrew Zisserman[2]

[1]Computer Vision for Human Computer Interaction, Karlsruhe Institute of Technology, Germany
[2]Visual Geometry Group, Department of Engineering Science, University of Oxford, UK
[3]Center for Machine Vision Research, University of Oulu, Finland

tapaswi@kit.edu, omkar@robots.ox.ac.uk, erahtu@ee.oulu.fi
eric@robots.ox.ac.uk, rainer.stiefelhagen@kit.edu, az@robots.ox.ac.uk

ABSTRACT

The goal of this paper is unsupervised face clustering in edited video material – where face tracks arising from different people are assigned to separate clusters, with one cluster for each person. In particular we explore the extent to which faces can be clustered automatically without making an error. This is a very challenging problem given the variation in pose, lighting and expressions that can occur, and the similarities between different people.

The novelty we bring is three fold: first, we show that a form of weak supervision is available from the editing structure of the material – the shots, threads and scenes that are standard in edited video; second, we show that by first clustering within scenes the number of face tracks can be significantly reduced with almost no errors; third, we propose an extension of the clustering method to entire episodes using exemplar SVMs based on the negative training data automatically harvested from the editing structure.

The method is demonstrated on multiple episodes from two very different TV series, Scrubs and Buffy. For both series it is shown that we move towards our goal, and also outperform a number of baselines from previous works.

Keywords

Face track clustering, TV shows, Video-editing structure

1. INTRODUCTION

Television broadcasting has seen a paradigm shift in the last decade as the internet has become an increasingly important distribution channel. Delivery platforms, such as the BBC's iPlayer, and distributors and portals like Netflix, Amazon Instant Video, Hulu, and Youtube, have millions of users every day. These new platforms include search tools for the video material based on the video title and metadata – however, for the most part it is not possible to search directly on the video content, e.g. find clips where a certain actor appears. Enabling such search services requires annotating the video content, and this is the goal towards which we work here.

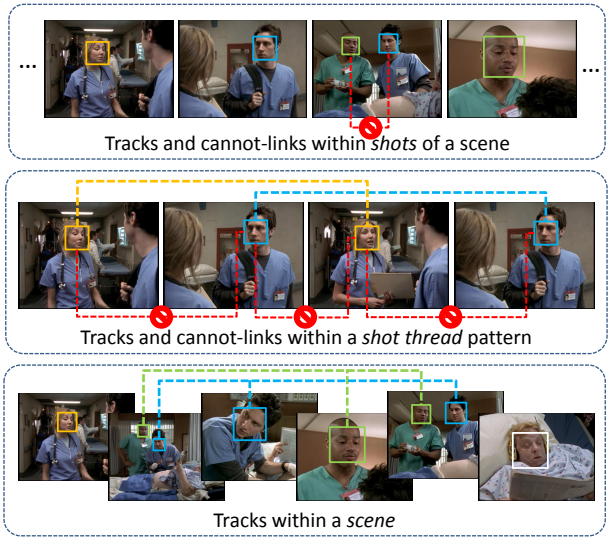

Figure 1: Overview of the video-editing structure of a TV series episode. Face tracks are shown for single shots of a scene (top row), in a threading pattern (middle row) and in a scene (bottom row). Face tracks with the same color denote the same person. Must-not links between tracks are denoted by red edges. (Best viewed in colour)

Our objective is to automatically cluster face tracks throughout a broadcast according to identity, i.e. to associate all the face tracks belonging to the same person. If successful, then annotating the video content for all actors simply requires attaching a label (either manually or automatically) for each cluster. There are many previous works that consider automatic face labelling in broadcast videos [2, 6, 5, 7, 9, 10, 25, 27, 29] but all of these require supervision in some form, typically subtitles and transcripts. Our approach of unsupervised clustering is complementary to these and the transcript supervision can be applied to automatically label the clusters. Furthermore, our clustering is also applicable in cases where transcripts are not available, e.g. for older or archive material. Here, manual effort can be greatly reduced compared to labelling all face tracks individually.

ICVGIP '14, December 14-18, 2014, Bangalore, India

ACM 978-1-4503-3061-9/14/12 ...$15.00
http://dx.doi.org/10.1145/2683483.2683490.

The novelty we bring is to take account of the editing structure of the video (Sect. 3) when considering candidate face tracks to cluster. These cues have not been used in previous works (Sect. 2 on related work). In particular we show that: (i) the thread structure can be used both to determine must-not and do links between face tracks; (ii) clustering first *within* scenes allows pure clusters to be obtained whilst significantly reducing the number of unassigned face tracks; and (iii) these clusters can subsequently form strong classifiers for *between* scene clustering.

In addition, we make two technical contributions: (i) we improve on the standard frontal face detectors used in face clustering by also including profile detections and upper-body detections; and (ii) we make use of the recently introduced Fisher Vector based face track representation with discriminative dimensionality reduction [19] (Sect. 7).

As will be seen (Sect. 6), we are able to substantially reduce the number of face tracks that need to be labelled. Though we do not reach the goal of one cluster per actor, the proposed method obtains better clustering performance compared to a number of baselines based on previous approaches.

2. RELATED WORK

In this section, we review previous work on unsupervised face track clustering.

One approach is to cast face clustering as a pure data clustering problem and attempt to solve it using general purpose algorithms. For instance, a hierarchical bottom up agglomerative clustering was applied in [3], [14], and [22]. Cinbis *et al*. [3] utilise a special distance metric that is learned using automatically obtained positive and negative face pairs. Similarly, Khoury *et al*.[14] learn a metric based on a combination of appearance cues and Gaussian Mixture Models.

A second approach is to construct a graphical model of the face detections or tracks and formulate the clustering as an optimization problem on the graph. As in [3], constraints between nodes of a graph are obtained by assuming that all detections in one face track belong to the same actor and that one person cannot appear twice in the same frame. Wu *et al*.[31, 32] use a Hidden Markov Random Field model to improve face tracking, by first detecting faces, creating small "tracklets" and then joining them using a constrained clustering method, with [32] extending [31] by clustering across shots. However, both [31, 32] require the number of output clusters K to be specified, and this information is not available in our case.

A third approach is to explicitly utilise the video structure in clustering. For instance, Ramanan *et al*. [21] use a notion of shots and scenes and first perform agglomerative clustering in each shot, then in each scene, and finally in the episode. At each level, they apply a different feature weighting (on faces, hair and clothing).

We also note that others have used video editing structure, such as threads, for various tasks [4, 6], e.g. for detecting continuity errors [20] or generating story graphs [28], but as far as we know we are the first to use the editing structure in this manner to provide supervision for unsupervised face clustering.

3. VIDEO STRUCTURE

Movies and TV broadcasts are generally filmed and edited using a well defined set of rules [18, 26], for example: the "$ABAB$" shot structuring; the "over-the-shoulder" shot for dialogue exchange between two characters; the 180 degree rule, etc. In this section, we briefly describe a few typical structures used in the video editing process – namely shots, threads and scenes. Figure 1 provides examples of face tracks seen in shots, threads and scenes of an episode.

3.1 Shots

A shot refers to a group of consecutive frames filmed from the same camera. In this paper, we segment shots by detecting their boundaries using a normalized version of the Displaced Frame Difference (DFD) [33]. The DFD captures the difference between consecutive motion compensated frames and produces peaks at shot boundaries as the optical flow cannot compensate the sudden change in camera.

In our case, a shot also limits the temporal extent of any face track. In other words, all detections in any face track must belong to the same shot. In previous work, e.g. [3, 32], temporal co-occurrence of two face tracks in a shot has been used to form examples of track pairs that must depict different characters. We will also use this as one approach for producing cannot-link constraints and negative training examples.

3.2 Threads

A thread is a sequence of shots, obtained from the same camera angle and located in a short temporal window (see Fig. 2). Typically threads form patterns like "$ABAB$" where the story is presented using alternating shots from two different cameras that form threads A and B. Such patterns work well with the previously mentioned "over-the-shoulder" shots as they depict dialogues between characters standing opposing each other. Note that A and B in the "$ABAB$" denote threads (cameras) and not two different people. It is not uncommon to have more than one visible character in each thread.

A threading pattern can also be more complex and involve more than two cameras (see Fig. 2). For example, during a dialogue between two people, two different cameras can provide close-up face shots of the speakers, while a third one gives an overview of the setting (a group shot) in which they are conversing. Such a setup can form threading patterns such as "$ABCACBCA$" where A and B are as above and C is the thread obtained from the third view.

While many shots (and thus tracks) are part of a threading pattern, there are others which do not form any thread. An example of a shot which does not belong to a thread is a *long take* – a shot that pans across a large part of the set and follows the actors on screen.

An analysis of our Scrubs and Buffy data sets (*cf*. Tab. 1 and 2 respectively) shows that about 60-70% of shots belong to a thread and correspondingly more than 68% of all face tracks belong to a shot which is in a threading pattern.

We detect shot threads using a method proposed in [28] that computes SIFT matches between the last frame of a shot and the first frame of F subsequent shots. The threads are used to generate both highly confident must-link (positive) pairs and cannot-link (negative) pairs between face tracks.

3.3 Scenes

A group of shots that have continuity in location, people and appearance such as interior/exterior constitute a scene.

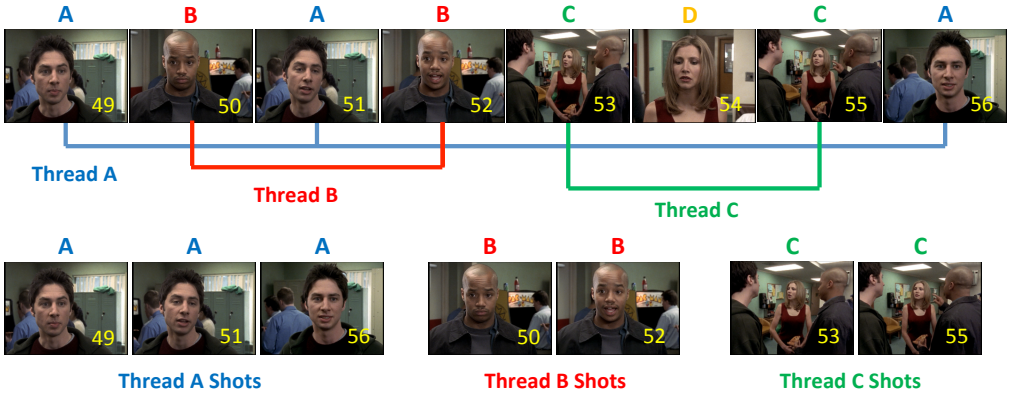

Figure 2: Editing structure in a video: Shots are the atomic entities in editing patterns. Each image displays one frame from a shot with its representative shot number written in yellow. Threads are interlaced repetition of shots in a temporal neighbourhood. Shots in threads appear to repeat themselves. For example, shots 49, 51, 56 form one thread *A*, while shots 50, 52 form thread *B* and 53, 55 form thread *C*. It is also common for some shots to be not part of any threads (shot 54). Multiple single shots and threads form a scene and an episode is a collection of all scenes.

A scene typically consists of multiple intertwined shot thread patterns.

To partition an episode into scenes, we employ the method of [28]. This locates scene boundaries by optimizing a cost function such that the within-scene content similarity is maximized while the between-scene similarity is small. The cost is based on shot appearance cues in the form of colour histograms, and video editing structure in the form of shot threads where scene boundaries are placed to minimize breaking of threads. The method uses dynamic programming to efficiently determine the optimal scene boundary localization, and also to automatically determine the number of scenes. Note that scene boundaries are restricted to only occur at shot boundaries.

Scene partitioning allows us to simplify the large problem of clustering about 500 tracks to dealing with only 20-40 tracks at a time. We show that by first clustering within scenes, we can build pure clusters more effectively than clustering directly across an episode and, in turn, these within scene clusters are more suitable for between scene clustering than the original tracks.

4. THE CLUSTERING PROCESS

We follow a three-step procedure to obtain the final face track clusters: (1) obtain *must-not-link* face track pairs (negative pairs) using the shots and threading structure of the video, (2) group the face tracks within each scene (*part 1 clustering*), and (3) merge the obtained scene level clusters at the episode level (*part 2 clustering*). In this section, we explain the details related to each stage of the proposed method. For readability, details of the face tracking and feature extraction are postponed until Section 7.

4.1 Obtaining negative pairs

The process of obtaining negative pairs is a pre-processing step crucial for the success of the following clustering steps. In this work, we consider three different sources for obtaining the *must-not-link* face tracks pairs.

1. Temporal co-occurrence: As in previous works [3, 32], all pairs of temporally co-occurring face tracks are assumed to depict different characters. This assumption holds in most cases, but may fail e.g. if there are reflections from a mirror. An example of this type of must-not-link is shown in the top row of Figure 1.

2. ABAB threading patterns: A simple alternating threading pattern such as "*ABAB*" typically contains different characters in the threads *A* and *B*. In such patterns, we form must-not-link constraint between every track in thread *A* versus every track in thread *B*. Shots 49-52 in Figure 2 show an example of this pattern.

3. Complex threading patterns: The simple rule applied to "*ABAB*" does not generalize to more complex patterns like "*ABCACABCA*". Nevertheless, they can be used to form must-not-link pairs as follows. First, we construct all possible pairs where the face tracks originate from different threads. Then we compute the face descriptor distance within each pair and discard pairs where the distance is below a given threshold. The remaining pairs are considered as must-not-link samples. Shots 53-56 in Figure 2 show examples of this case.

Negative pairs play an important role in our two stage clustering approach. For the first part of clustering within a scene, negative pair face descriptor distances are reset to ∞. In the second stage of clustering across scenes, negative pairs are used to train discriminative models between clusters.

4.2 Part 1 Clustering: within a scene

For a given episode, consider a scene s. Let $\mathcal{T} = \{t_i\}$ be the set of face tracks in scene s. We represent a face track t_i by its appearance descriptor $\phi_i \in \mathbb{R}^{D\times 1}$. Given the video-editing structure and the cannot-link constraints described above, we first set distances of track pairs which belong to the cannot-link category to ∞.

Next, we exploit the fact that track pairs originating from the same shot thread should allow for a relaxed distance threshold, as the number of options to which they can be matched is restricted. To obtain this threshold, we train a linear classifier (SVM) which combines the descriptor distance $d(\cdot,\cdot)$ with the intersection-over-union measure $\gamma(\cdot,\cdot)$ computed on average face locations, and predict a track pair as arising from the same person when

$$-w_d \cdot d(t_i, t_j) + w_\gamma \cdot \gamma(t_i, t_j) > \theta_{th} \ . \qquad (1)$$

Additionally, we learn a tight scene-level distance threshold θ_g to merge track pairs which are very similar in appearance

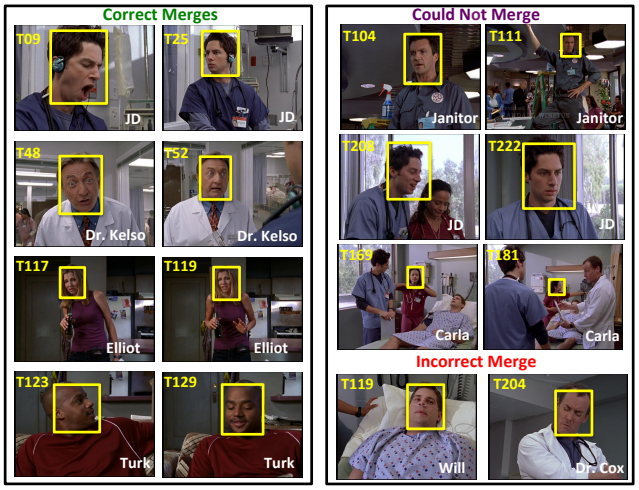

Figure 3: Output of Part-1 of our clustering method. Each image represents a track with given track id and label. Track ids indicate temporal closeness of each track in a pair. Left: examples of correct merges. Note tracks which contain quite different head poses (frontal and near profile) are able to be merged. Right: examples of track pairs that could be merged but are not because of the head pose variation and lower temporal proximity ruling out the thread based merging. The last pair shows the first mistake that is made in episode 2 of Scrubs.

but are not part of any thread. Track pairs t_i, t_j are classified as belonging to the same class when

$$d(t_i, t_j) = \|\phi_i - \phi_j\| < \theta_g \ . \tag{2}$$

In Section 6, we show that this approach leads to a reduction in the number of clusters whilst keeping the cluster purity very high. After this process, we update the must-not link constraints using newly formed clusters. This helps us in the next stage of clustering.

Discussion. Clustering within a scene has been mentioned as a relatively easy problem [21], where a very simple approach based on pair-wise track distances and hierarchical agglomerative clustering (HAC) is used. Whilst this simple technique works well for near frontal faces used in their case, the availability of better face detectors capable of detecting faces in various poses (as is our case), can lead to errors when such a simple distance threshold based clustering method is used. For example we observe that (Fig. 3), the distance between two relatively frontal tracks of two different people (right column row 4) could well be smaller than the distance between a frontal and a profile track of the same person (left column rows 1 and 4). To facilitate merging of such tracks of the same person with varying poses, we first employ the video editing structure to provide clues about clustering. The use of spatial overlap from the thread structure allows us to lower the threshold for merging tracks of the same person within a thread (left column rows 1 and 4) before merging tracks of different people (right column row 4).

4.3 Part 2 Clustering: across scenes

Given the scene-level clusters, we now move towards merging them at the episode level. Towards this goal, we take three important steps. First, to achieve a compact and powerful descriptor, we extend the video pooling idea of [19] to obtain one Fisher vector over all face tracks in a single cluster. Apart from improving the invariance of the features, this reduces the number of comparisons that are otherwise needed in typical agglomerative clustering.

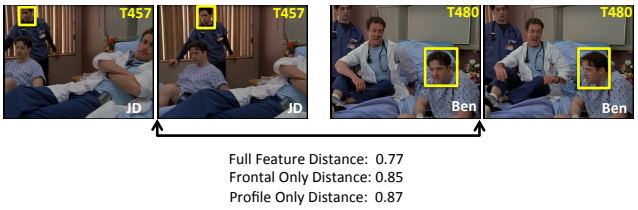

Figure 4: Comparison of frontal and profile feature distances vs. full feature distance. As can be seen, computing pose dependent features avoids making a wrong merge.

Secondly, we partition the detections into two components based on the head pose. This idea is very similar to the mixture model based approach of Deformable Parts Model [12] and to more recent work [1, 23] that also considers pose explicitly in the face representation. In our paper, all frontal detections form one component and those having yaw angle (rotation about a vertical axis) larger than 45^o form another component. We compute one high dimensional Fisher vector for each of these components. Separating detections in this way makes the task of comparing two clusters much easier as pose dependent features are compared separately for a pair of clusters.

Figure 4 shows our motivation behind this. We see that the complete feature has a smaller distance between two different characters, while splitting the feature into components prevents this erroneous merge. Splitting a track or a cluster according to pose not only prohibits false merges, but also allows merging thresholds to be slightly looser in order to merge some other clusters which might not have merged.

Note that it is not necessary that each cluster will have both frontal and profile parts. We observe that performing this split on clusters after within-scene clustering yields about 15-20% of clusters which contain only frontal faces and thus an only frontal feature; about 5% of clusters consist only of profile faces generating only profile features; while the remainder 75-80% form the bulk and have both frontal as well as profile features. We design our merging strategy to take care of this. We merge two clusters if either of the distances is within threshold. So even if a cluster is missing one component, the other component is sufficient to merge the two clusters.

Finally, rather than the distance metric based comparison from the previous part, we use exemplar Support Vector Machine (e-SVM) classifiers as our scoring scheme. This lets us use the negative information obtained from the video structure and part-1 clustering. In addition to this, we use the tracks from YouTube Faces In The Wild dataset [30] as stock negatives. We train one e-SVM per cluster and test it on all other clusters. The e-SVM is trained separately for frontal and profile components.

Calibration. The e-SVM scores are uncalibrated, thus finding one threshold for merging clusters is impossible. To overcome this problem, we normalize these scores using a sigmoid function.

$$\phi(x) = \frac{1}{1 + \exp(-ax + b)} \ . \tag{3}$$

We wish to learn parameters a and b on a training episode. However, as the number of training samples to learn these parameters is small, we use scores from all e-SVMs in the training episode and adapt them for a particular e-SVM using the score statistics

$$a_k = \alpha \cdot \mu(S_k)/\sigma(S_k) \text{ and } b_k = \beta \cdot 1/\sigma(S_k) \quad (4)$$

where S_k is the set of e-SVM scores of all other clusters obtained for an e-SVM model k and $\mu(\cdot)$ and $\sigma(\cdot)$ compute the mean and standard deviation respectively. We optimize for parameters $\alpha^{\{f,p\}}$ and $\beta^{\{f,p\}}$ (respectively for frontal and profile models) such that we maximize the number of correctly identified positive pairs and at the same time keep the erroneous merges at zero.

Please note that while we use e-SVMs and supervised training, there is no manual effort in gathering the labels as the negative data is mined automatically. Thus the method is fully automatic and unsupervised.

5. EVALUATION PROTOCOL

5.1 Dataset information

We evaluate our techniques on two varied TV series datasets consisting of a sitcom *Scrubs* and a fantasy drama *Buffy the Vampire Slayer*.

Scrubs: In this sitcom, the story of the protagonist takes place primarily at a hospital (and his apartment) providing many challenging scenarios such as: large number of background characters, characters moving through corridors, etc. For the evaluation, we pick season 1, episodes 1–5 (SCR-1, ..., SCR-5). We chose episode 23 (SCR-23) towards the end of the season to train our parameters. For completeness, we also show clustering results for this episode.

Buffy the Vampire Slayer: The series, categorized as a supernatural drama, presents multiple challenges from illumination, magic (character duplication), fast action, etc. While another dataset on Buffy tracks [3] has been used for face clustering before, we cannot use it since the data is obtained from multiple episodes and essentially treats face tracks as independent data points. The face tracks in our scheme have strong ties to the video-editing structure and benefit most when the entire episode is analyzed. We pick season 5, episodes 1–6 (BF-1, ..., BF-6) and select one episode (BF-4) for learning the parameters and thresholds.

Face tracks are obtained automatically and then labelled manually to obtain ground truth. We label all primary characters and assign all background characters a single label. These background characters are not considered for evaluation in our case. More details on how we obtain tracks and compute features are discussed in Section 7.

Table 1 and 2 present statistics on the *Scrubs* and *Buffy* data sets respectively. We show the number of video-editing elements (shots, threads and scenes), and the number of tracks in the episode. We also display the number of shots and tracks which are part of a shot thread. The final section of the tables presents the number of mined negative pairs.

5.2 Evaluation criteria

We use two measures to evaluate the quality of clustering:

Weighted Clustering Purity (WCP) The weighted clustering purity (WCP) is our primary measure as we want to perform clustering with no errors. For a given clustering $\mathcal{C}$

$$WCP = \frac{1}{N} \sum_{c \in \mathcal{C}} n_c \cdot purity_c \quad (5)$$

where each cluster c contains n_c elements and its purity is measured as a fraction of the largest number of tracks which belong to the same character to the number of tracks in the cluster n_c. N denotes total number of tracks in the video.

Operator Clicks Index (OCI-k) [13] Along with WCP, we also report the clustering quality in terms of the number of clicks required to label all the face tracks for a given clustering. The lower and upper bounds for this metric are determined by the number of characters and face tracks, respectively.

An advantage of this measure is that it simultaneously incorporates the number of clusters *and* cluster quality in one number. However, the advantage is also a drawback, since we can argue that correcting an error in clustering is *not* equivalent to labelling a pure cluster. A single click corresponds to the effort needed to correctly label an individual face track in a wrong cluster, or all tracks in one cluster. For example, if we have 19 pure clusters (i.e. containing only tracks of the same character) and one cluster of 15 more data samples of which 5 are wrongly clustered samples, OCI-k needs a total of 25 (20 clusters + 5 errors) clicks to label all face tracks correctly.

Table 1: Statistics for the Scrubs face track data set along with video-editing structure cues.

	SCR-1	SCR-2	SCR-3	SCR-4	SCR-5	SCR-23
shots	450	370	379	319	360	315
threads	91	70	79	53	73	69
scenes	27	21	24	25	23	21
named char.	17	12	15	17	16	19
tracks for named	495	413	376	365	404	419
shots in thread	295	242	260	223	264	242
tracks in thread	340	271	254	230	279	309
negs in-shot	232	192	116	212	222	310
negs abab thread	136	104	114	88	174	316
negs complex thread	1146	432	370	552	1078	596

Table 2: Statistics for the Buffy face track data set along with video-editing structure cues.

	BF-1	BF-2	BF-3	BF-4	BF-5	BF-6
shots	678	616	820	714	675	745
threads	121	114	165	120	105	151
scenes	37	35	37	48	39	37
named char.	11	15	13	15	18	18
tracks for named	630	779	973	668	646	842
shots in thread	419	383	550	413	363	494
tracks in thread	449	552	668	425	408	592
negs in-shot	282	620	752	216	192	582
negs abab thread	176	240	172	58	170	164
negs complex thread	1742	1672	1916	212	712	4114

6. EXPERIMENTS

We now evaluate the performance of our method and compare it against different baseline approaches.

6.1 Negative Pairs

In the first stage of our clustering, we form negative links between different tracks. Here we compare our strategy de-

scribed in Section 4.1 to previous methods of finding negatives only by temporal co-occurrence. Tables 1 and 2 show that our method obtains significantly more negatives than just the pairs of temporally co-occurring tracks as negatives.

For example, on the first episode of Scrubs (SCR-1), our method finds 1146 negative pairs in complex threads and 136 pairs in simple threads in addition to only 232 pairs found by temporal co-occurrence in a shot.

6.2 Clustering within a scene

Our goal for the clustering is to minimize the number of clusters while maintaining a very high cluster purity. Our parameter selection using the training episode is tuned towards this goal.

As discussed in Section 4.2, within-scene clustering is performed using both a tight global distance threshold, and also a relaxed threshold learned for tracks within threads which incorporates both appearance and spatial overlap measurements.

We compare to a strong baseline algorithm that merges tracks using agglomerative clustering assisted by negative information. The threshold for the baseline is also learnt on the training episode. This baseline is similar to that of previous works, e.g. [21].

As can be seen from Table 3 row 3 and 4 for the Scrubs dataset, our strategy significantly reduces the number of clusters. For example, in the second episode of the Scrubs dataset (SCR-2), we manage to halve the number of clusters from 413 (i.e. the original number of tracks) to 202. Similar dramatic reductions can also be found in SCR-4 (365 to 181) and SCR-5 (404 to 217).

On the Buffy dataset, Table 4, we are able to reduce the number of clusters by more than 100 while maintaining cluster purity. This suggests that our proposed usage of video-editing structure is widely applicable to other TV series.

We outperform the baseline on both datasets (rows 3 vs. 4), reducing the number of clusters but without sacrificing cluster quality or increasing the number of clicks required to label these clusters.

Tracks can be fragmented due to face detection drop-out or temporary occlusion. This often presents quite an easy clustering situation as the face descriptors can be quite similar for the track portions. Also, clustering within a thread can be relatively easy. The baseline often copes with these cases, and the proposed method goes beyond the baseline.

Figure 5 shows examples of the within-scene clustering on two scenes. We plot the hierarchical agglomerative clustering (HAC) purity obtained at every cluster number and overlay our baseline performance at the best possible threshold. We also show that our proposed method outperforms the baseline.

Note, within-scene comparison figures for all scenes of one episode are included in the supplementary material.

6.3 Clustering across scenes

In this section we look at effects of the clustering strategies discussed in Section 4.3.

As can be seen from Table 3 for the Scrubs dataset, our method significantly reduces the number of clusters while keeping the cluster purity very close to 1 (rows 5 and 7). We evaluate our scheme of comparing pose dependent features for clusters to that of using a single feature for the entire cluster (Tab. 3 rows 6 and 7). Our proposed pose-dependent features reduce the number of clusters without compromising on the cluster purity.

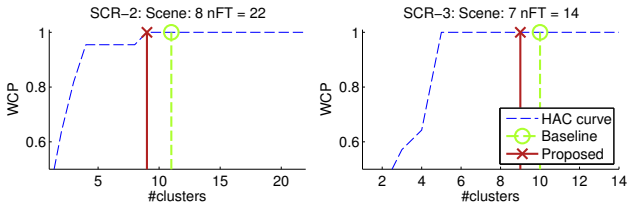

Figure 5: Within-scene clustering results showing the number of clusters vs. WCP for episode 2 scene 8; and episode 3 scene 7. nFT is the number of face tracks in the scene. Similar results for the other scenes are available in the supplementary material.

A baseline for this part is to run hierarchical agglomerative clustering (HAC) on the entire episode. As in the previous section, this resembles the approach followed in [21]. Additionally [3] also followed a similar approach for clustering faces. As can be seen, the OCI-k click index is much lower than the baseline. *E.g.* on episode 4 we reduce 365 original clusters to 147 while the number of clusters obtained by the baseline method (182) is in fact more than that obtained by the first stage of our proposed method. Both these trends can also be observed on episode 2 where we reduce from 413 to 185 while the baseline is in fact worst than our part 1 208 and 202 respectively.

We see a similar effect on the Buffy dataset as well where a significant reduction in the number of clusters is obtained without any loss of purity (Tab. 4 rows 5 and 6).

We show the results of clustering on the full episode in Figure 6. Similar to Figure 5, we plot the HAC curve and mark the point at which the baseline stops. Our proposed two-stage method achieves higher reduction in the number of clusters whilst maintaining the purity of the clusters. We include the figure for the Buffy data set in the supplementary material.

7. IMPLEMENTATION DETAILS

We now discuss a few implementation details such as face detection, tracking and representation which are a pre-cursor for our task of face track clustering.

Face detection. This is the most important step of the pipeline. Compared to frontal detections, detecting faces in extreme poses (profile, even the back of the head) remains very challenging. However, the inability to detect these faces can cause fragmentation of tracks, making the clustering task difficult. Furthermore, false positive face detections – frequently occurring with non-frontal detectors – are another problem affecting the tracking output.

To overcome these limitations we propose two solutions. First, we run the upper-body detector [8] on every frame and restrict the face detections to the upper-body region addressing false positives. Second, to achieve high recall, in addition to the OpenCV Viola-Jones face detectors (frontal and profile), we use the head detector of [17] designed to detect a human head in frontal as well as extreme poses (frontal-left/right, profile-left/right and even the back of head). This detector is based on the Deformable Parts Model of [11].

Face tracking. Face tracks are formed by performing the data association technique described in [10]. Specifically, we

Table 3: Clustering results on Scrubs. Episode SCR-23 is used for learning parameters.

	Episodes	SCR-1			SCR-2			SCR-3			SCR-4			SCR-5			SCR-23		
1	#tracks	495			413			376			365			404			419		
2	#ideal	17			12			15			17			16			19		
	Measures	NC	WCP	OCI-k	NC	WCP	OCI-k	NC	WCP	OCI-k	NC	WCP	OCI-k	NC	WCP	OCI-k	NC	WCP	OCI-k
	Within scene clustering																		
3	Baseline	319	0.996	321	233	1.000	233	222	0.997	223	200	1.000	200	251	1.000	251	232	1.000	232
4	Proposed part-1	293	1.000	293	202	0.994	205	205	0.998	206	181	1.000	181	217	1.000	217	212	1.000	212
	Full episode clustering																		
5	Baseline	287	1.000	287	208	0.998	209	196	0.995	198	182	1.000	182	246	1.000	246	208	1.000	208
6	SVM full feature	293	1.000	293	202	0.994	205	205	0.998	206	181	1.000	181	217	1.000	217	210	1.000	210
7	Proposed part-2	244	0.992	248	185	0.992	188	179	0.992	182	147	0.998	148	198	0.998	199	173	1.000	173

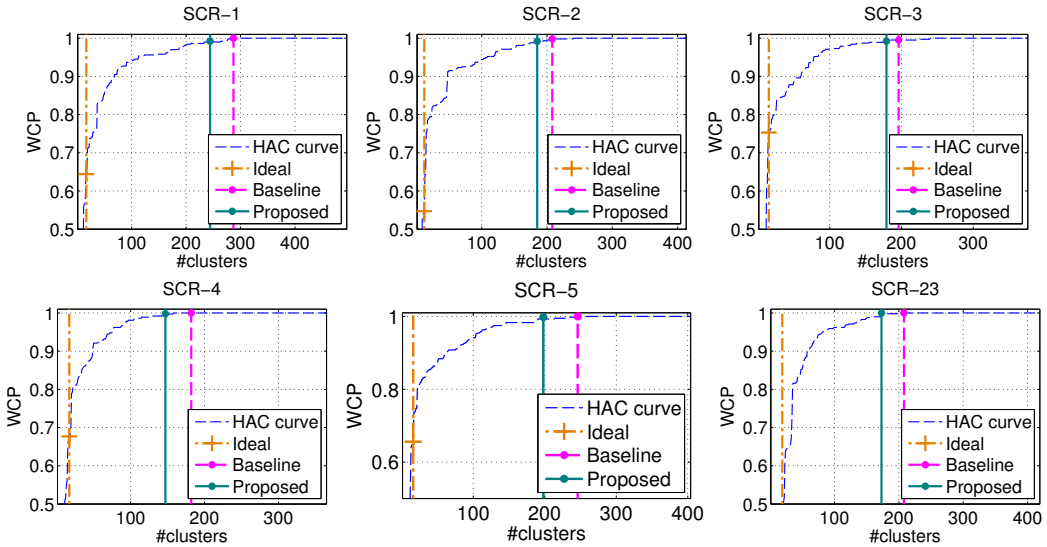

Figure 6: The number of clusters versus the cluster purity for each episode in the Scrubs dataset. It can be observed that our approach consistently results in a smaller number of clusters with similar purity than the baseline. The HAC curve illustrates the clustering results obtained with standard agglomerative clustering with different thresholds. Note that one cannot directly select an operating point from this curve, since the corresponding threshold is not known (threshold needs to be learned using holdout data). Similar curves for the Buffy dataset are available in the supplementary material.

use the Kanade-Lucas-Tomasi (KLT) [24] tracker to obtain feature tracks intersecting with face detections. To achieve robust performance, feature tracking is carried out in both forward and backward directions. After linking detections in a track, misses between two detections are obtained by interpolation based on two neighbouring detections and the spread of KLT feature tracks in that frame. To remove false positive face tracks, we follow an approach similar to [15]. We form a track level feature vector using different track statistics: track features track length, ratio of track length to number of detected faces in a track, mean and standard deviation of head detection scores. This feature vector is then used to classify the track using a linear-SVM classifier trained on ground truth false positive and true positive tracks obtained from data distinct from that used in this work.

Face representation. Fisher vector based representation of face tracks was recently introduced in [19]. The descriptor is computed by first extracting dense SIFT [16] features for every detection in a track and then pooling them together to form a single Fisher vector feature for the whole track. The resulting descriptor is a high dimensional vector (67584) which is reduced to a very low dimension (128) using discriminative dimensionality reduction. To make the descriptor more robust, dense SIFT features are computed on horizontal flips of every detection and pooled into the same Fisher vector. We use the projection matrix learned on "YouTube Faces Dataset" as described in [19] to perform the dimensionality reduction.

8. CONCLUSIONS AND EXTENSIONS

In this paper we have presented a method for unsupervised face track clustering. Unlike many previous methods, we explicitly utilise video structure in clustering. In particular, we take advantage of shot threads, which allow us to form several must-not-link constraints and to use optimized similarity thresholds within the threading patterns. In the experiments, we showed that our approach can greatly reduce the number of face tracks without making almost any mistakes in the process. Such an output is a much better starting point for many supervised methods as well as manual track labelling. In addition, we illustrate a clear improvement over the baseline that also utilises video structure in the form of shots and scenes.

Table 4: Clustering results on Buffy. Episode BF-4 is used for learning parameters.

	Episodes	BF-1			BF-2			BF-3			BF-4			BF-5			BF-6		
1	#tracks	630			779			974			668			646			843		
2	#ideal	11			15			13			15			18			18		
	Measures	NC	WCP	OCI-k	NC	WCP	OCI-k	NC	WCP	OCI-k	NC	WCP	OCI-k	NC	WCP	OCI-k	NC	WCP	OCI-k
	Within scene clustering																		
3	Baseline	537	1.000	537	697	1.000	697	852	1.000	852	567	1.000	567	584	0.999	585	761	1.000	761
4	Proposed part-1	501	1.000	501	655	1.000	655	814	1.000	814	524	1.000	524	550	0.999	551	717	1.000	717
	Full episode clustering																		
5	Baseline	534	1.000	534	688	1.000	688	852	1.000	852	566	1.000	566	575	0.999	576	751	1.000	751
6	Proposed part-2	466	1.000	466	598	1.000	598	730	1.000	730	494	1.000	494	507	0.998	508	643	1.000	643

We have concentrated in this paper on clustering using faces alone. However, since a scene forms a short and coherent story segment, it is justified to assume that the appearance (e.g. of hair, clothing, etc.) of the characters do not change much within one scene. This can be used to assist the clustering, e.g. by relaxing distance thresholds or using appropriate weighting for different cues [21].

Acknowledgments

This work was supported by ERC grant VisRec no. 228180, EU Project AXES ICT–269980, and a Royal Society Wolfson Research Merit Award.

9. REFERENCES

[1] B. Bhattarai, G. Sharma, F. Jurie, and P. Perez. Some faces are more equal than others: Hierarchical organization for accurate and efficient large-scale identity-based face retrieval. In *ECCV Workshop*, 2014.

[2] P. Bojanowski, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Finding actors and actions in movies. In *Proc. ICCV*, 2013.

[3] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Unsupervised metric learning for face identification in TV video. In *Proc. ICCV*, 2011.

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

[5] T. Cour, B. Sapp, A. Nagle, and B. Taskar. Talking pictures: Temporal grouping and dialog-supervised person recognition. In *Proc. CVPR*, 2010.

[6] T. Cour, B. Sapp, and B. Taskar. Learning from ambiguously labeled images. In *Proc. CVPR*, 2009.

[7] T. Cour, B. Sapp, and B. Taskar. Learning from partial labels. *J. Machine Learning Research*, 2011.

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

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

[10] M. Everingham, J. Sivic, and A. Zisserman. Taking the bite out of automatic naming of characters in TV video. *Image and Vision Computing*, 27(5), 2009.

[11] P. Felzenszwalb, D. Mcallester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. CVPR*, 2008.

[12] P. F. Felzenszwalb, R. B. Grishick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE PAMI*, 2010.

[13] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *Proc. ICCV*, 2009.

[14] E. Khoury, P. Gay, and J.-M. Odobez. Fusing Matching and Biometric Similarity Measures for Face Diarization in Video. In *ICMR*, 2013.

[15] A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman. Human focused action localization in video. In *International Workshop on Sign, Gesture, Activity*, 2010.

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[17] M. Marin-Jimenez, A. Zisserman, and V. Ferrari. "Here's looking at you, kid." Detecting people looking at each other in videos. In *Proc. BMVC.*, 2011.

[18] J. Monaco. *How to Read a Film: The World of Movies, Media, Multimedia – Language, History, Theory.* OUP USA, Apr 2000.

[19] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face track descriptor. In *Proc. CVPR*, 2014.

[20] L. C. Pickup and A. Zisserman. Automatic retrieval of visual continuity errors in movies. In *Proc. CIVR*, 2009.

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

[22] J. See and C. Eswaran. Exemplar Extraction Using Spatio-Temporal Hierarchical Agglomerative Clustering for Face Recognition in Video. In *ICCV*, 2011.

[23] G. Sharma, F. Jurie, and P. Perez. EPML: Expanded Parts based Metric Learning for Occlusion Robust Face Verification. In *ACCV*, 2014.

[24] J. Shi and C. Tomasi. Good features to track. In *Proc. CVPR*, pages 593–600, 1994.

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

[26] T. J. Smith. *An Attentional Theory of Continuity Editing.* PhD thesis, University of Edinburgh, 2006. Unpublished Doctoral Thesis.

[27] M. Tapaswi, M. Bäuml, and R. Stiefelhagen. "Knock! Knock! Who is it?" Probabilistic Person Identification in TV Series. In *Proc. CVPR*, 2012.

[28] M. Tapaswi, M. Bäuml, and R. Stiefelhagen. StoryGraphs: Visualizing Character Interactions as a Timeline. In *CVPR*, 2014.

[29] P. Wohlhart, M. Köstinger, P. M. Roth, and H. Bischof. Multiple instance boosting for face recognition in videos. In *DAGM-Symposium*, 2011.

[30] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proc. CVPR*, 2011.

[31] B. Wu, S. Lyu, B.-G. Hu, and Q. Ji. Simultaneous Clustering and Tracklet Linking for Multi-Face Tracking in Videos. In *ICCV*, 2013.

[32] B. Wu, Y. Zhang, B.-G. Hu, and Q. Ji. Constrained Clustering and Its Application to Face Clustering in Videos. In *CVPR*, 2013.

[33] Y. Yusoff, W. Christmas, and J. Kittler. A Study on Automatic Shot Change Detection. *Multimedia Applications, Services and Techniques*, 1998.