End-to-end Face Detection and Cast Grouping in Movies Using Erdős-Rényi Clustering

SouYoung Jin[1], Hang Su[1], Chris Stauffer[2], and Erik Learned-Miller[1]
[1]College of Information and Computer Sciences, University of Massachusetts, Amherst
[2]Visionary Systems and Research (VSR)

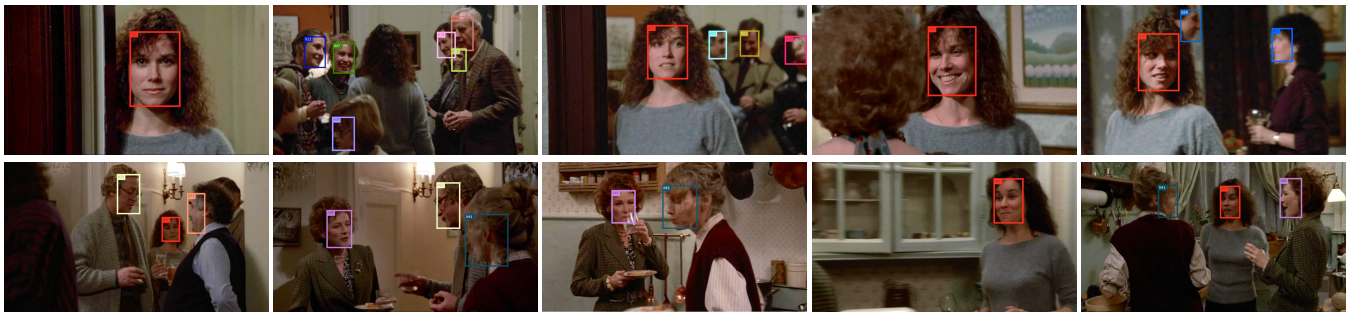


Figure 1: **Clustering results from *Hannah and Her Sisters*.** Each unique color shows a particular cluster. It can be seen that most individuals appear with a consistent color, indicating successful clustering.

Abstract

We present an end-to-end system for detecting and clustering faces by identity in full-length movies. Unlike works that start with a predefined set of detected faces, we consider the end-to-end problem of detection and clustering together. We make three separate contributions. First, we combine a state-of-the-art face detector with a generic tracker to extract high quality face tracklets. We then introduce a novel clustering method, motivated by the classic graph theory results of Erdős and Rényi. It is based on the observations that large clusters can be fully connected by joining just a small fraction of their point pairs, while just a single connection between two different people can lead to poor clustering results. This suggests clustering using a verification system with very few false positives *but perhaps moderate recall. We introduce a novel verification method,* **rank-1 counts verification***, that has this property, and use it in a link-based clustering scheme. Finally, we define a novel end-to-end detection and clustering evaluation metric allowing us to assess the accuracy of the entire end-to-end system. We present state-of-the-art results on multiple video data sets and also on standard face databases.*

Project page: http://souyoungjin.com/erclustering

This research is based in part upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) under contract number 2014-14071600010. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

1. Introduction

The problem of identifying face images in video and clustering them together by identity is a natural precursor to high impact applications such as video understanding and analysis. This general problem area was popularized in the paper "Hello! My name is...Buffy" [9], which used text captions and face analysis to name people in each frame of a full-length video. In this work, we use only raw video (with no captions), and group faces by identity rather than naming the characters. In addition, unlike face clustering methods that start with detected faces, we include detection as part of the problem. This means we must deal with false positives and false negatives, both algorithmically, and in our evaluation method. We make three contributions:

- A new approach to combining high-quality face detection [15] and generic tracking [31] to improve both precision and recall of our video face detection.
- A new method, *Erdős-Rényi clustering*, for large-scale clustering of images and video tracklets. We argue that effective large-scale face clustering requires face verification with fewer false positives, and we introduce *rank-1 counts verification*, showing that it indeed achieves better true positive rates in low false positive regimes. Rank-1 counts verification, used with simple link-based clustering, achieves high quality clustering results on three separate video data sets.
- A principled evaluation for the end-to-end problem of face detection and clustering in videos; until now there has been no clear way to evaluate the quality of such an

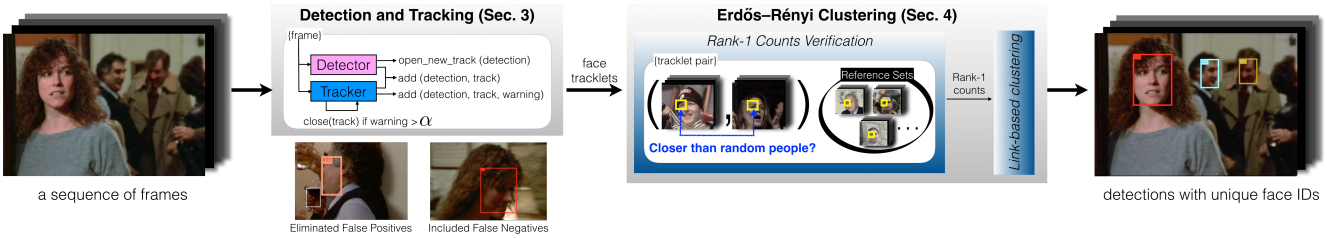


Figure 2: Overview of approach. Given a movie, our approach generates tracklets (Sec. 3) and then does Erdős-Rényi Clustering and FAD verification between all tracklet pairs. (Sec. 4) Our final output is detections with unique character Ids.

end-to-end system, but only to evaluate its individual parts (detection and clustering).

We structure the paper as follows. In Section 2 we discuss related work. In Section 3, we describe the first phase of our system, in which we use a face detector and generic tracker to extract face *tracklets*. In Section 4, we introduce Erdős-Rényi clustering and rank-1 counts verification. Sections 5 and 6 present experiments and discussions.

2. Related Work

In this section, we first discuss face tracking and then the problem of naming TV (or movie) characters. We can divide the character-naming work into two categories: fully unsupervised and with some supervision. We then discuss prior work using reference images. Related work on clustering is covered in Section 5.2.

Recent work on *robust face tracking* [36, 29, 24] has gradually expanded the length of face tracklets, starting from face detection results. Ozerov *et al*. [24] merge results from different detectors by clustering based on spatio-temporal similarity. Clusters are then merged, interpolated, and smoothed for face tracklet creation. Similarly, Roth et al. [29] generate low-level tracklets by merging detection results, form high-level tracklets by linking low-level tracklets, and apply the Hungarian algorithm to form even longer tracklets. Tapaswi et al. [36] improve on this [29] by removing false positive tracklets.

With the development of multi-face tracking techniques, *the problem of naming TV characters*[1] has been also widely studied [35, 13, 9, 2, 39, 40, 37]. Given precomputed face tracklets, the goal is to assign a name or an ID to a group of face tracklets with the same identity. Wu et al. [39, 40] iteratively cluster face tracklets and link clusters into longer tracks in a bootstrapping manner. Tapaswi et al. [37] train classifiers to find thresholds for joining tracklets in two stages: within a scene and across scenes. Similarly, we aim to generate face clusters in a fully unsupervised manner.

[1]Another related problem is *person re-identification* [44, 18, 6] in which the goal is to tell whether a person of interest seen in one camera has been observed by another camera. Re-identification typically uses the whole body on short time scales while *naming TV characters* focuses on faces, but over a longer period of time.

Though solving this problem may yield a better result for face tracking, some forms of supervision specific to the video or characters in the test data can improve performance. Tapaswi et al. [35] perform face recognition, clothing clustering and speaker identification, where face models and speaker models are first trained on other videos containing the same main characters as in the test set. In [9, 2], subtitles and transcripts are used to obtain weak labels for face tracks. More recently, Haurilet et al. [13] solve the problem without transcripts by resolving name references only in subtitles. Our approach is more broadly applicable because it does not use subtitles, transcripts, or any other supervision related to the identities in the test data, unlike these other works [35, 13, 9, 2].

As in the proposed verification system, some existing work [4, 12] uses reference images. For example, index code methods [12] map each single image to a code based upon a set of reference images, and then compare these codes. On the other hand, our method compares the relative distance of two images with the distance of one of the images to the reference set, which is different. In addition, we use the newly defined rank-1 counts, rather than traditional Euclidean or Mahalanobis distance measures to compare images [4, 12] for similarity measures.

3. Detection and tracking

Our goal is to take raw videos, with no captions or annotations, and to detect all faces and cluster them by identity. We start by describing our method for generating *face tracklets*, or continuous sequences of the same face across video frames. We wish to generate clean face tracklets that contain face detections from just a single identity. Ideally, exactly one tracklet should be generated for an identity from the moment his/her face appears in a shot until the moment it disappears or is completely occluded.

To achieve this, we first detect faces in each video frame using the *Faster R-CNN* object detector [28], but retrained on the WIDER face data set [41], as described by Jiang et al. [15]. Even with this advanced detector, face detection sometimes fails under challenging illumination or pose. In videos, those faces can be detected before or after the chal-

lenging circumstances by using a tracker that tracks both forward and backward in time. We use the *distribution field tracker* [31], a general object tracker that is not trained specifically for faces. Unlike face detectors, the tracker's goal is to find in the next frame the object most similar to the target in the current frame. The extra faces found by the tracker compensate for missed detections (Fig. 2, bottom of block 2). Tracking helps not only to catch false negatives, but also to link faces of equivalent identity in different frames.

One simple approach to combining a detector and tracker is to run a tracker forward and backward in time from *every single face detection* for some fixed number of frames, producing a large number of "mini-tracks". A Viterbi-style algorithm [10, 5] can then be used to combine these mini-tracks into longer sequences. This approach is computationally expensive since the tracker is run many times on overlapping subsequences, producing heavily redundant mini-tracks. To improve performance, we developed the following novel method for combining a detector and tracker. Happily, it also improves precision and recall, since it takes advantage of the tracker's ability to form long face tracks of a single identity.

The method starts by running the face detector in each frame. When a face is first detected, a tracker is initialized with that face. In subsequent frames, faces are again detected. In addition, we examine each current tracklet to see where it might be extended by the tracking algorithm in the current frame. We then check the agreement between detection and tracking results. We use the intersection over union (IoU) between detections and tracking results with threshold 0.3, and apply the Hungarian algorithm[16] to establish correspondences among multiple matches. If a detection matches a tracking result, the detection is stored in the current face sequence such that the tracker can search in the next frame given the detection result. For the detections that have no matched tracking result, a new tracklet is initiated. If there are tracking results that have no associated detections, it means that either **a)** the tracker could not find an appropriate area on the current frame, or **b)** the tracking result is correct while the detector failed to find the face. The algorithm postpones its decision about the tracked region for the next α consecutive frames ($\alpha = 10$). If the face sequence has any matches with detections within α frames, the algorithm will keep the tracking results. Otherwise, it will remove the tracking-only results. The second block of Fig. 2 summarizes our proposed face tracklet generation algorithm and shows examples corrected by our joint detection-tracking strategy. Next, we describe our approach to clustering based on low false positive verification.

4. Erdős-Rényi Clustering and Rank-1 Counts Verification

In this section, we describe our approach to clustering face images, or, in the case of videos, face tracklets. We adopt the basic paradigm of *linkage clustering*, in which each pair of points (either images or tracklets) is evaluated for linking, and then clusters are formed among all points connected by linked face pairs. We name our general approach to clustering *Erdős-Rényi clustering* since it is inspired by classic results in graph theory due to Erdős and Rényi [7], as described next.

Consider a graph G with n vertices and probability p of each possible edge being present. This is the Erdős-Rényi random graph model [7]. The expected number of edges is $\binom{n}{2}p$. One of the central results of this work is that, for $\epsilon > 0$ and n sufficiently large, if

$$p > \frac{(1+\epsilon)\ln n}{n}, \tag{1}$$

then the graph will almost surely be connected (there exists a path from each vertex to every other vertex). Fig. 3 shows this effect on different graph sizes, obtained through simulation.

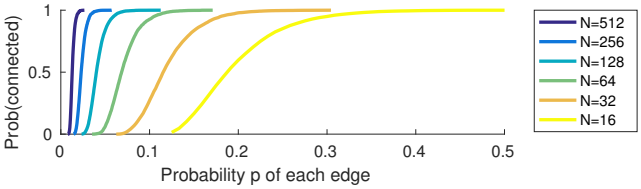


Figure 3: Simulation of cluster connectedness as a function of cluster size, N, and the probability p of connecting point pairs. The figure shows that for various N (different colored lines), the probability that the cluster is fully connected (on the y-axis) goes up as more pairs are connected. For larger graphs, a small probability of connected pairs still leads to high probability that the graph will be fully connected.

Consider a clustering system in which links are made between tracklets by a *verifier* (a face verification system), whose job is to say whether a pair of tracklets is the "same" person or two "different" people. While graphs obtained in clustering problems are not uniformly random graphs, the results of Erdős and Rényi suggest that this verifier can have a fairly low recall (percentage of same links that are connected) and still do a good job connecting large clusters. In addition, false matches may connect large clusters of different identities, dramatically hurting clustering performance. This motivates us to build a verifier that focuses on low false positives rather than high recall. In the next section, we present our approach to building a verifier that is designed to have good recall at low false positive rates,

and hence is appropriate for clustering problems with large clusters, like grouping cast members in movies.

4.1. Rank-1 counts for fewer false positives

Our method compares images by comparing their multidimensional feature vectors. More specifically, we count the number of feature dimensions in which the two images are closer in value than the first image is to any of a set of reference images. We call this number the *rank-1 count* similarity. Intuitively, two images whose feature values are "very close" for many different dimensions are more likely to be the same person. Here, an image is considered "very close" to a second image in one dimension if it is closer to the second image in that dimension than to any of the reference images.

More formally, to compare two images I_A and I_B, our first step is to obtain feature vectors A and B for these images. We extract 4096-D feature vectors from the *fc7* layer of a standard pre-trained face recognition CNN [26]. In addition to these two images, we use a fixed reference set with G images (we typically set $G = 50$), and compute CNN feature vectors for each of these reference images.[2] Let the CNN feature vectors for the reference images be $R^1, R^2, ..., R^G$. We sample reference images from the *TV Human Interactions Dataset* [27], since these are likely to have a similar distribution to the images we want to cluster.

For each feature dimension i (of the 4096), we ask whether

$$|A_i - B_i| < \min_j |A_i - R_i^j|.$$

That is, is the value in dimension i closer between A and B than between A and all the reference images? If so, then we say that the ith feature dimension is *rank-1* between A and B. The cumulative *rank-1 counts* feature $\mathbf{R}$ is simply the number of rank-1 counts across all 4096 features:

$$\mathbf{R} = \sum_{i=1}^{4096} I\left[|A_i - B_i| < \min_j |A_i - R_i^j|\right],$$

where $I[\cdot]$ is an indicator function which is 1 if the expression is true and 0 otherwise.

Taking inspiration from Barlow's notion that the brain takes special note of "suspicious coincidences" [1], each rank-1 feature dimension can be considered a suspicious coincidence. It provides some weak evidence that A and B may be two images of the same person. On the other hand, in comparing all 4096 feature dimensions, we expect to obtain quite a large number of rank-1 feature dimensions even if A and B are *not* the same person.

When two images and the reference set are selected randomly from a large distribution of faces (in this case they are usually different people), the probability that A is closer to B in a particular feature dimension than to any of the reference images is just

$$\frac{1}{G+1}.$$

Repeating this process 4096 times means that the expected number of rank-1 counts is simply

$$E[\mathbf{R}] = \frac{4096}{G+1},$$

since expectations are linear (even in the presence of statistical dependencies among the feature dimensions). Note that this calculation is a fairly tight *upper bound* on the expected number of rank-1 features *conditioned on the images being of different identities*, since most pairs of images in large clustering problems are different, and conditioning on "different" will tend reduce the expected rank-1 count. Now if two images I_A and I_B have a large rank-1 count, it is likely they represent the same person. The key question is how to set the threshold on these counts to obtain the best verification performance.

Recall that our goal, as guided by the Erdős-Rényi random graph model, is to find a threshold on the rank-1 counts $\mathbf{R}$ so that we obtain very few false positives (declaring two different faces to be "same") while still achieving good recall (a large number of same faces declared to be "same"). Fig. 4 shows distributions of rank-1 counts for various subsets of image pairs from Labeled Faces in the Wild (LFW) [14]. The red curve shows the distribution of rank-1 counts for *mismatched* pairs from all possible mismatched pairs in the entire data set (not just the test sets). Notice that the mean is exactly where we would expect with a gallery size of 50, at $\frac{4096}{51} \approx 80$. The green curve shows the distribution of rank-1 counts for the matched pairs, which is clearly much higher. The challenge for clustering, of course, is that we don't have access to these distributions since we don't know which pairs are matched and which are not. The yellow curve shows the rank-1 counts for *all* pairs of images in LFW, which is nearly identical to the distribution of mismatched rank-1 counts, *since the vast majority of possibe pairs in all of LFW are mismatched*. This is the distribution to which the clustering algorithm has access.

If the 4,096 CNN features were statistically independent (but not identically distributed), then the distribution of rank-1 counts would be a binomial distribution (blue curve). In this case, it would be easy to set a threshold on the rank-1 counts to guarantee a small number of false positives, by simply setting the threshold to be near the right end of the mismatched (red) distribution. However, the dependencies among the CNN features prevent the mismatched rank-1 counts distribution from being binomial, and so this approach is not possible.

[2] The reference images may overlap in identity with the clustering set, but we choose reference images so that there is no more than one occurrence of each person in the reference set.

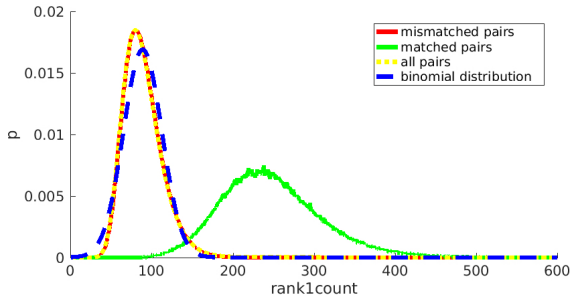


Figure 4: LFW distribution of rank-1 counts. Each distribution is normalized to sum to 1.

Table 1: Verification performance comparisons on all possible LFW pairs. The proposed rank-1 counts gets much higher recall at fixed FPRs.

FPR	**Rank1count**	L2	Template Adaptation [4]	Rank-Order Distance [45]
1E-9	**0.0252**	0.0068	0.0016	0.0086
1E-8	**0.0342**	0.0094	0.0017	0.0086
1E-7	**0.0614**	0.0330	0.0034	0.0086
1E-6	**0.1872**	0.1279	0.0175	0.0086
1E-5	**0.3800**	0.3154	0.0767	0.0427
1E-4	**0.6096**	0.5600	0.2388	0.2589
1E-3	0.8222	0.7952	0.5215	**0.8719**
1E-2	0.9490	0.9396	0.8204	**0.9656**
1E-1	**0.9939**	0.9915	0.9776	0.9861

4.2. Automatic determination of rank-1 count threshold

Ideally, if we could obtain the rank-1 count distribution of mismatched pairs of a test set, we could set the threshold such that the number of false positives becomes very low. However, it is not clear how to get the actual distribution of rank-1 counts for mismatched pairs at test time.

Instead, we can estimate the shape of the mismatched pair rank-1 count distribution using one distribution (LFW), and use it to estimate the distribution of mismatched rank-1 counts for the test distribution. We do this by fitting the *left half* of the LFW distribution to the *left half* of the clustering distribution using scale and location parameters. The reason we use the left half to fit the distribution is that this part of the rank-1 counts distribution is almost exclusively influence by *mismatched pairs*. The *right side* of this matched distribution then gives us an approximate way to threshold the test distribution to obtain a certain false positive rate. It is this method that we use to report the results in the left-most column of Table 2.

A key property of our rank-1 counts verifier is that it has good recall across a wide range of the low false positive regime. Thus, our method is relatively robust to the setting of the rank-1 counts threshold. In order to show that our rank-1 counts feature has good performance for the types of verification problems used in clustering, we construct a verification problem using *all possible pairs* of the LFW database [14]. In this case, the number of mismatched pairs (quadratic in N) is much greater than the number of matched pairs. As shown in Table 1, we observe that our verifier has higher recall than three competing methods (all of which use the same base CNN representation) at low false positive rates.

Using rank-1 counts verification for tracklet clustering. In our face clustering application, we consider every pair (I, J) of tracklets, calculate a value akin to the rank-1 count R, and join the tracklets if the threshold is exceeded. In order to calculate an R value for tracklets, we sample a random subset of 10 face images from each tracklet, compute a rank-1 count R for each pair of images, and take the maximum of the resulting R values.

4.3. Averaging over gallery sets

While our basic algorithm uses a fixed (but randomly selected) reference gallery, the method is susceptible to the case in which one of the gallery images happens to be similar in appearance to a person with a large cluster, resulting in a large number of false negatives. To mitigate this effect, we implicitly average the rank-1 counts over an exponential number of random galleries, as follows.

The idea is to sample random galleries of size g from a larger *super-gallery* with G images; we used $g = 50, G = 1000$. We are interested rank-1 counts, in which image A's feature is closer to B than to any of the gallery of size g. Suppose we know that among the 1000 super-gallery images, there are K (e.g., $K = 3$) that are closer to A than B is. The probability that a random selection (with replacement) of g images from the super-gallery would contain none of the K closer images (and hence represent a rank-1 count) is

$$r(A, B) = \left(1.0 - \frac{K}{G}\right)^g.$$

That is, $r(A, B)$ is the *probability* of having a rank-1 count with a random gallery, and using $r(A, B)$ as the count is equivalent to averaging over all possible random galleries. In our final algorithm, we sum these probabilities rather than the deterministic rank-1 counts.

4.4. Efficient implementation

For simplicity, we discuss the computational complexity of our fixed gallery algorithm; the complexity of the average gallery algorithm is similar. With F, G, and N indicating the feature dimensionality, number of gallery images, and

number of face tracklets to be clustered, the time complexity of the naive rank-1 count algorithm is $\mathcal{O}(F * G * N^2)$.

However, for each feature dimension, we can sort N test image feature values and G gallery image feature values in time $\mathcal{O}((N+G)\log(N+G))$. Then, for each value in test image A, we find the closest gallery value, and increment the rank-1 count for the test images that are closer to A. Let Y be the average number of steps to find the closest gallery value. This is typically much smaller than N. The time complexity is then $\mathcal{O}(F * [(N+G)\log(N+G) + N * Y])$.

4.5. Clustering with do-not-link constraints

It is common in clustering applications to incorporate constraints such as *do-not-link* or *must-link*, which specify that certain pairs should be in separate clusters or the same cluster, respectively [38, 32, 19, 17, 21]. They are also often seen in the face clustering literature [3, 39, 40, 25, 37, 43]. These constraints can be either rigid, implying they must be enforced [38, 32, 21, 25], or soft, meaning that violations cause an increase in the loss function, but those violations may be tolerated if other considerations are more important in reducing the loss [19, 17, 39, 40, 43].

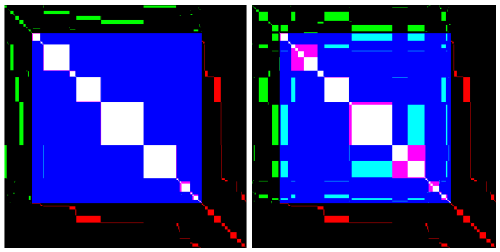
In this work, we assume that if two faces appear in the same frame, they must be from different people, and hence their face images obey a do-not-link constraint. Furthermore, we extend this hard constraint to the tracklets that contain faces. If two tracklets have any overlap in time, then the entire tracklets represent a do-not-link constraint.

We enforce these constraints on our clustering procedure. Note that connecting all pairs below a certain dissimilarity threshold followed by transitive closure is equivalent to single-linkage agglomerative clustering with a joining threshold. In agglomerative clustering, a pair of closest clusters is found and joined at each iteration until there is a single cluster left or a threshold met. A naïve implementation will simply search and update the dissimilarity matrix at each iteration, making the whole process $\mathcal{O}(n^3)$ in time. There are faster algorithms giving the optimal time complexity $\mathcal{O}(n^2)$ for single-linkage clustering [34, 22]. Many of these algorithms incur a dissimilarity update at each iteration, i.e. update $d(i,k) = \min(d(i,k), d(j,k))$ after combining cluster i and j (and using i as the cluster id of the resulting cluster). If the pairs with do-not-link constraints are initialized with $+\infty$ dissimilarity, the aforementioned update rule can be modified to incorporate the constraints without affecting the time and space complexity:

$$d(i,k) = \begin{cases} \min(d(i,k), d(j,k)) & d(i,k) \neq +\infty \\ & \text{AND } d(j,k) \neq +\infty \\ +\infty & \text{otherwise} \end{cases}$$

5. Experiments

We evaluate our proposed approach on three video data sets: *the Big Bang Theory* (BBT) Season 1 (s01), Episodes 1-6 (e01-e06) [2], *Buffy the Vampire Slayer* (Buffy) Season 5 (s05), Episodes 1-6 (e01-e06) [2], and *Hannah and Her Sisters* (Hannah) [24]. Each episode of the BBT and Buffy data set contains 5-8 and 11-17 characters respectively, while Hannah has annotations for 235 characters.[3] Buffy and Hannah have many occlusions which make the face clustering problem more challenging. In addition to the video data sets, we also evaluate our clustering algorithm on LFW [14] which contains 5730 subjects.[4]



(a) Rank-1 Count (b) Rank-Order Distance [45]

Figure 5: Visualization of the combined detection and clustering metric for the first few minutes of the Hannah set.

An end-to-end evaluation metric. There are many evaluation metrics used to independently evaluate detection, tracking, and clustering. Previously, it has been difficult to evaluate the relative performance of two end-to-end systems because of the complex trade-offs between detection, tracking, and clustering performance. Some researchers have attempted to overcome this problem by providing a reference set of detections with suggested metrics [20], but this approach precludes optimizing complete system performance. To support evaluation of the full video-to-identity pipeline, in which false positives, false negatives, and clustering errors are handled in a common framework, we introduce *unified pairwise precision* (UPP) and *unified pairwise recall* (UPR) as follows.

Given a set of annotations, $\{a_1, a_2, ..., a_A\}$ and detections, $\{d_1, d_2, ..., d_D\}$, we consider the union of three sets of tuples: false positives resulting from unannotated face detections $\{d_i, \emptyset\}$; valid face detections $\{d_i, a_j\}$; and false negatives resulting from unmatched annotations $\{\emptyset, a_j\}$. Fig. 5 visualizes every possible pair of tuples ordered by false positives, valid detections, and false negatives for the first few minutes of the Hannah data set. Further, groups of tuples have been ordered by identity to show blocks of identity to aid our understanding of the visualization, although the order is inconsequential for the numerical analysis.

In Fig 5, the large blue region (and the regions it contains) represents all pairs of annotated detections, where we have valid detections corresponding to their best annotation. In this region, white pairs are correctly clustered, magenta pairs are the same individual but not clustered, cyan pairs are clustered but not the same individual, and

[3] We removed garbage classes such as 'unknown' or 'false_positive'.

[4] All known ground truth errors are removed.

Table 2: Clustering performance comparisons on various data sets. The leftmost shows our **rank1count** by setting a threshold automatically. For the rest of the columns, we show f-scores using optimal (oracle-supplied) thresholds. For BBT and Buffy, we show average scores over six episodes. The full table with individual episode results is given in Supp. Mat. Best viewed in color (**1st place**, **2nd place**, **3rd place**).

		Verification system + Link-based clustering algorithm					Other clustering algorithms					
	Test set	**Rank-1 Count (automatic threshold)**	**Rank-1 Count**	L2	Template Adaptation [4]	Rank-Order Distance [45]	Rank-Order Distance based Clustering [45]	Affinity Propagation [11]	DBSCAN [8]	Spectral Clustering [33]	Birch [42]	MiniBatch KMeans [30]
Video	BBT s01 [2]	**.7728**	**.7828**	.7365	**.7612**	.6692	.6634	.1916	.2936	.6319	.2326	.1945
	Buffy s05 [2]	**.5661**	**.6299**	.3931	**.5845**	.2990	.5439	.1601	.1409	.5351	.1214	.1143
	Hannah [24]	**.6436**	**.6813**	.2581	.3620	**.4123**	.3955	.1886	.1230	.3344	.1240	.1052
Image	LFW [14]	**.8532**	**.8943**	**.8498**	.3735	.5989	.5812	.3197	.0117	.2538	.4520	.3133

blue pairs are not clustered pairs from different individuals. The upper left portion of the matrix represents false positives with no corresponding annotation. The green pairs in this region correspond to any false positive matching with any valid detection. The lower right portion of the matrix corresponds to the false negatives. The red pairs in this region correspond to any missed clustered pairs resulting from these missed detections. The ideal result would contain blue and white pairs, with no green, red, cyan, or magenta.

The unified pairwise precision (UPP) is the fraction of pairs, $\{d_i, a_j\}$ within all clusters with matching identities, *i.e.*, the number of white pairs divided by the number of white, cyan, and green pairs. UPP decreases if: two matched detections in a cluster do not correspond to the same individual; if a matched detection is clustered with a false positive; for each false positive regardless of its clustering; and for false positives clustered with valid detections. Similarly, the unified pairwise recall (UPR) is the fraction of pairs within all identities that have been properly clustered, *i.e.*, the number of white pairs divided by number of white, magenta, and red pairs. UPR decreases if: two matched detections of the same identity are not clustered; a matched detection should be matched but there is no corresponding detection; for each false negative; and for false negative pairs that should be detected and clustered. The only way to achieve perfect UPP and UPR is to detect every face with no false positives and cluster all faces correctly. At a glance, our visualization in Fig. 5 shows that our detection produces few false negatives, many more false positives, and is less aggressive in clustering. Using this unified metric, others can tune their own detection, tracking, and clustering algorithms to optimize the unified performance metrics. Note that for image matching without any detection failures, the UPP and UPR reduce to standard pairwise precision and pairwise recall.

The UPP and UPR can be summarized with a single F-measure (the weighted harmonic mean) providing a single, unified performance measure for the entire process. It can be α-weighted to alter the relative value of precision and recall performance:

$$F_\alpha = \frac{1}{\frac{\alpha}{UPP} + \frac{1-\alpha}{UPR}} \quad (2)$$

where $\alpha \in [0, 1]$. $\alpha = 0.5$ denotes a balanced F-measure.

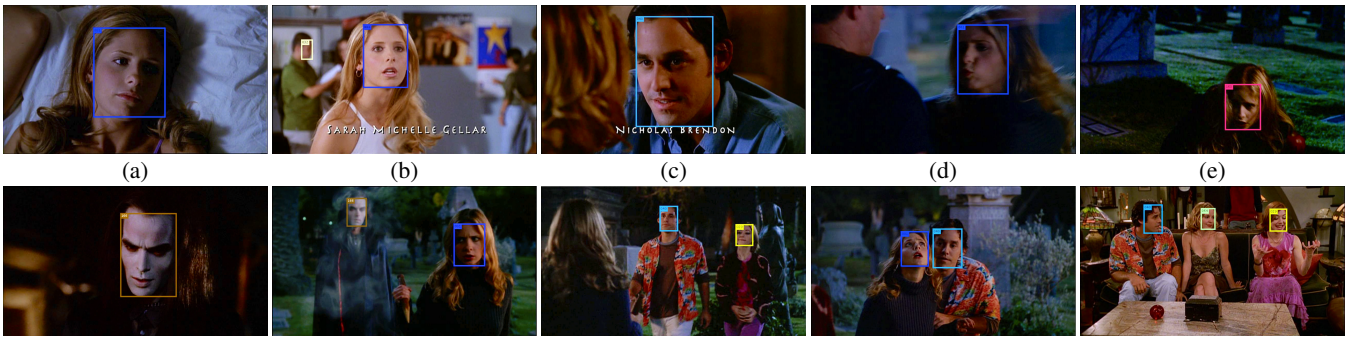
5.1. Threshold for rank-1 counts

The leftmost column in Table 2 shows our clustering results when the threshold is set automatically by the validation set. We used LFW as a validation set for BBT, Buffy and Hannah while Hannah was used for LFW. Note that the proposed method is very competitive even when the threshold is automatically set.

5.2. Comparisons

We can divide other clustering algorithms into two broad categories–link-based clustering algorithms (like ours) that use a different similarity function and clustering algorithms that are not link-based (such as spectral clustering [33]). Table 2 shows the comparisons to various distance functions [4, 23, 45] with our link-based clustering algorithm. L2 shows competitive performance in LFW while the performance drops dramatically when a test set has large pose variations. We also compare against a recent so-called "template adaptation" method [4] which also requires a reference set. It takes 2nd and 3rd place on Buffy and BBT. In addition, we compare to the rank-order method [45] in two different ways: link-based clustering algorithms using their rank-order distance, and rank-order distance based clustering.

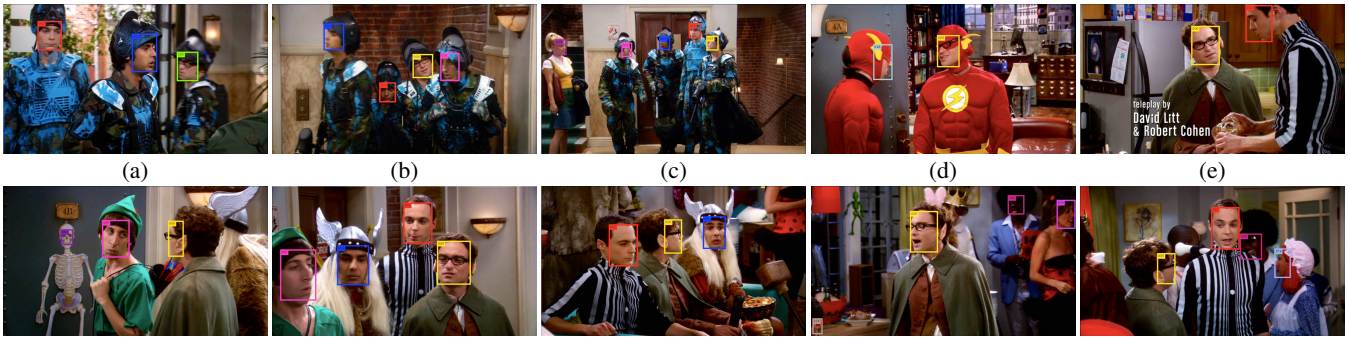
In addition, we compare against several generic clustering algorithms (Affinity Propagation [11], DBSCAN [8],



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

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

Figure 6: **Clustering results from *Buffy the Vampire Slayer***. A failure example can be seen in frame (e), in which the main character Buffy (otherwise in a purple box) in shown in a pink box.



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

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

Figure 7: **Clustering results from *the Big Bang Theory***. A failure example can be seen in frame (d), in which the main character Howard (otherwise in a magenta box) in shown in a gray box.

Spectral Clustering [33], Birch [42], KMeans [30]), where L2 distance is used as pairwise metric. For algorithms that can take as input the similarity matrix (Affinity Propagation, DBSCAN, Spectral Clustering), do-not-link constraints are applied by setting the distance between the corresponding pairs to ∞. Note that this is just an approximation, and in general does not guarantee the constraints in the final clustering result (*e.g.* for single-linkage agglomerative clustering, a modified update rule is also needed in Section 4.5).

Note that all other settings (feature encoding, tracklet generation) are common for all methods. In Table 2, except for the leftmost column, we report the best $F_{0.5}$ scores using optimal (oracle-supplied) thresholds for (number of clusters, distance). The link-based clustering algorithm with rank-1 counts outperforms the state-of-the-art on all four data sets in $F_{0.5}$ score. Figures 6 and 7 show some clustering results on Buffy and BBT.

6. Discussion

We have presented a system for doing end-to-end clustering in full length videos and movies. In addition to a careful combination of detection and tracking, and a new end-to-end evaluation metric, we have introduced a novel approach to link-based clustering that we call Erdős-Rényi clustering. We demonstrated a method for automatically estimating a good decision threshold for a verification method based on rank-1 counts by estimating the underlying portion of the rank-1 counts distribution due to mismatched pairs.

This decision threshold was shown to result in good recall at a low false-positive operating point. Such operating points are critical for large clustering problems, since the vast majority of pairs are from different clusters, and false positive links that incorrectly join clusters can have a large negative effect on clustering performance.

There are several things that could disrupt our algorithm: a) if a high percentage of different pairs are highly similar (e.g. family members), b) if only a small percentage of pairs are different (e.g., one cluster contains 90% of the images), and if same pairs lack lots of matching features (e.g., every cluster is a pair of images of the same person under extremely different conditions). Nevertheless, we showed excellent results on 3 popular video data sets. Not only do we dominate other methods when thresholds are optimized for clustering, but we outperform other methods even when our thresholds are picked automatically.

References

[1] H. Barlow. Cerebral cortex as model builder. In *Matters of Intelligence*, pages 395–406. Springer, 1987. 4

[2] M. Bauml, M. Tapaswi, and R. Stiefelhagen. Semi-supervised learning with constraints for person identification in multimedia data. In *Proc. CVPR*, 2013. 2, 6, 7

[3] R. G. Cinbis, J. Verbeek, and C. Schmid. Unsupervised metric learning for face identification in TV video. In *Proc. ICCV*, 2011. 6

[4] N. Crosswhite, J. Byrne, C. Stauffer, O. M. Parkhi, Q. Cao, and A. Zisserman. Template adaptation for face verification and identification. In *Face and Gesture*, 2017. 2, 5, 7

[5] S. J. Davey, M. G. Rutten, and B. Cheung. A comparison of detection performance for several track-before-detect algorithms. *EURASIP Journal on Advances in Signal Processing*, 2008:41, 2008. 3

[6] B. DeCann and A. Ross. Modeling errors in a biometric re-identification system. *IET Biometrics*, 4(4):209–219, 2015. 2

[7] P. Erdős and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960. 3

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 96(34):226–231, 1996. 7

[9] M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is... Buffy" Automatic naming of characters in TV video. In *Proc. BMVC*, 2006. 1, 2

[10] G. D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. 3

[11] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007. 7

[12] A. Gyaourova and A. Ross. Index codes for multibiometric pattern retrieval. *IEEE Transactions on Information Forensics and Security (TIFS)*, 7(2):518–529, April 2012. 2

[13] M.-L. Haurilet, M. Tapaswi, Z. Al-Halah, and R. Stiefelhagen. Naming TV characters by watching and analyzing dialogs. In *Proc. CVPR*, 2016. 2

[14] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *The Workshop on Faces in Real-Life Images at ECCV*, 2008. 4, 5, 6, 7

[15] H. Jiang and E. Learned-Miller. Face detection with the Faster R-CNN. In *Face and Gesture*, 2017. 1, 2

[16] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 3

[17] Z. Li, J. Liu, and X. Tang. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proc. ICML*, 2008. 6

[18] G. Lisanti, I. Masi, A. D. Bagdanov, and A. D. Bimbo. Person re-identification by iterative re-weighted sparse ranking. *TPAMI*, 37(8):1629–1642, August 2015. 2

[19] Z. Lu and T. K. Leen. Penalized probabilistic clustering. *Neural Computation*, 19(6):1528–1567, 2007. 6

[20] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831. 6

[21] S. Miyamoto and A. Terami. Semi-supervised agglomerative hierarchical clustering algorithms with pairwise constraints. In *Fuzzy Systems (FUZZ)*, pages 1–6. IEEE, 2010. 6

[22] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012. 6

[23] C. Otto, D. Wang, and A. K. Jain. Clustering millions of faces by identity. *TPAMI*, Mar. 2017. 7

[24] A. Ozerov, J.-R. Vigouroux, L. Chevallier, and P. Pérez. On evaluating face tracks in movies. In *Proc. ICIP*, 2013. 2, 6, 7

[25] A. Ozerov, J.-R. Vigouroux, L. Chevallier, and P. Pérez. On evaluating face tracks in movies. In *Proc. ICIP*. IEEE, 2013. 6

[26] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *bmvc*, 2015. 4

[27] A. Patron-Perez, M. Marszaek, A. Zisserman, and I. D. Reid. High five: Recognising human interactions in tv shows. In *Proc. BMVC*, 2010. 4

[28] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proc. NIPS*, 2015. 2

[29] M. Roth, M. Bauml, R. Nevatia, and R. Stiefelhagen. Robust multi-pose face tracking by multi-stage tracklet association. In *Proc. ICPR*, 2012. 2

[30] D. Sculley. Web-scale k-means clustering. In *Proc. WWW*, pages 1177–1178. ACM, 2010. 7, 8

[31] L. Sevilla-Lara and E. Learned-Miller. Distribution fields for tracking. In *Proc. CVPR*, 2012. 1, 3

[32] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. In *Proc. NIPS*, 2004. 6

[33] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 22(8):888–905, 2000. 7, 8

[34] R. Sibson. SLINK: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973. 6

[35] M. Tapaswi, M. Bauml, and R. Stiefelhagen. "Knock! Knock! Who is it?" Probabilistic person identification in TV series. In *Proc. CVPR*, 2012. 2

[36] M. Tapaswi, C. C. Corez, M. Bauml, H. K. Ekenel, and R. Stiefelhagen. Cleaning up after a face tracker: False positive removal. In *Proc. ICIP*, 2014. 2

[37] M. Tapaswi, O. M. Parkhi, E. Rahtu, E. Sommerlade, R. Stiefelhagen, and A. Zisserman. Total cluster: A person agnostic clustering method for broadcast videos. In *ICVGIP*, 2014. 2, 6

[38] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *Proc. ICML*, 2001. 6

[39] B. Wu, S. Lyu, B.-G. Hu, and Q. Ji. Simultaneous clustering and tracklet linking for multi-face tracking in videos. In *Proc. ICCV*, 2013. 2, 6

[40] B. Wu, Y. Zhang, B.-G. Hu, and Q. Ji. Constrained clustering and its application to face clustering in videos. In *Proc. CVPR*, 2013. 2, 6

[41] S. Yang, P. Luo, C. C. Loy, and X. Tang. Wider face: A face detection benchmark. In *CVPR*, 2016. 2

[42] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *SIGMOD*. ACM, 1996. 7, 8

[43] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Joint face representation adaptation and clustering in videos. In *Proc. ECCV*, 2016. 6

[44] L. Zheng, Y. Yang, and A. G. Hauptman. Person re-identification: Past, present and future. *arXiv*, Oct. 2016. 2

[45] C. Zhu, F. Wen, and J. Sun. A rank-order distance based clustering algorithm for face tagging. In *Proc. CVPR*, 2011. 5, 6, 7