

Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of computers

Video Face Clustering

Václav Rýdl

Supervisor: Ing. Vojtěch Franc, Ph.D.
January 2019



ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Rýdl** Jméno: **Václav** Osobní číslo: **420156**
 Fakulta/ústav: **Fakulta elektrotechnická**
 Zadávající katedra/ústav: **Katedra počítačů**
 Studijní program: **Otevřená informatika**
 Studijní obor: **Datové vědy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Shlukování tváří ve videu

Název diplomové práce anglicky:

Video Face Clustering

Pokyny pro vypracování:

The input is a set of image sequences obtained after processing video by a face tracker. Each sequence contains either facial images of a single identity or it contains corrupted sequences of false detections and/or multiple identities. The goal of the thesis is to design and implement an algorithm which merges sequences corresponding to a single identity and it detects the corrupted sequences. The algorithm should require minimal human intervention and it should exploit spatio-temporal constraints available from the face tracker. Efficiency of the designed method will be statistically evaluated on a set of real-life benchmark problems.

Seznam doporučené literatury:

- [1] Zhou et al. Video face clustering via Constrained Sparse Representation. In Proc of Inter. Conf. On Multimedia and Expo (ICME), 2014.
- [2] Cau et al. Constrained Multi-Video Face Clustering. IEEE Trans. On Image Processing, 2015.
- [3] Yang et al. Neural Aggregation Network for Video Recognition. In Proc of CVPR, 2017.
- [4] Wu et al. Constrained Clustering and Its Application to Face Clustering in Videos. In Proc. of CVPR, 2013.
- [5] Zhang et al. Joint Face Representation Adaptation and Clustering in Videos. In Proc. of ECCV, 2016.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Vojtěch Franc, Ph.D., Strojové učení FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **31.01.2018**

Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **30.09.2019**

Ing. Vojtěch Franc, Ph.D.
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

_____ Datum převzetí zadání

_____ Podpis studenta

Acknowledgements

I would like to thank my supervisor Ing. Vojtěch Franc PhD. for his guidance and help. I would also like to thank my wife for her support and patience, and my entire family for their contributions to this work.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, January 1, 2019

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 1 ledna 2019

Abstract

In this thesis we propose a simple but effective method for clustering faces in a video. The proposed method is composed of three main building blocks. Firstly, we extract feature descriptor for each face in the track by Convolutional Neural Network pre-trained on still images. Secondly, we describe each track by a single feature vector obtained by aggregating feature descriptors previously extracted from the faces of the track. Thirdly, we group the track descriptors by a clustering algorithm. We have systematically tested several different implementations of the three building blocks in order to find the best performing setup. Experiments on real-life data show that the resulting method outperforms most of the state-of-the-art approaches. The output of this thesis is an open source implementation of the proposed method which is modular to allow easy changes of the individual components.

Keywords: video face clustering, convolutional neural networks, face recognition, face attribute extraction

Supervisor: Ing. Vojtěch Franc, Ph.D.
Department of Cybernetics

Abstrakt

V této práci navrhujeme jednoduchou a přesto efektivní metodu pro shlukování tváří ve videu. Tato metoda se skládá ze tří hlavních bloků. Nejprve extrahujeme příznaky z obličejů detekovaných ve videu pomocí konvoluční neuronové sítě natrénované na statických obrázcích. Poté z těchto příznaků vytvoříme deskriptory pro jednotlivé tracky agregací deskriptorů extrahovaných pro jednotlivé obrázky. Na závěr tyto deskriptory spojíme do shluků dle identit. V této práci systematicky testujeme více různých implementací jednotlivých částí naší metody za účelem nalezení nejlepší kombinace extrakce příznaků, agregace deskriptorů a shlukovacího algoritmu. Experimenty ukazují, že námi navržená metoda je na testovaných datasech lepší než většina současných prací na toto téma. Výstup této práce je open source implementace navržené metody, která je modulární a umožňuje jednoduché nahrazení jednotlivých komponent.

Klíčová slova: shlukování obličejů ve videu, konvoluční neuronové sítě, rozpoznávání obličejů, predikce atributů z obličeje

Překlad názvu: Shlukování tváří ve videu

Contents

1 Introduction	1	Bibliography	45
2 Related work	3	A Contents of the Attached Medium	49
2.1 Video Face Clustering methods . .	3		
2.2 Evaluation metrics	4		
3 Databases and evaluation protocols	7		
3.1 Summary of databases	7		
3.1.1 Buffy database	8		
3.1.2 Accio database	8		
3.2 Evaluation protocols	9		
4 Method	13		
4.1 Framework	13		
4.2 Tracking	14		
4.3 Feature extraction	14		
4.3.1 VGG-Face	15		
4.3.2 RESNET50	15		
4.3.3 Combination of RESNET50 and VGG-Face	15		
4.4 Feature aggregation	15		
4.4.1 Template selection	16		
4.4.2 Median computation	16		
4.5 Clustering algorithms	16		
4.5.1 Hierarchical clustering	16		
4.5.2 K-Means clustering	16		
4.5.3 Spectral clustering	17		
4.6 Outlier detection	17		
5 Experiments	19		
5.1 Hyper-parameter tuning	19		
5.1.1 Bounding-box size	20		
5.1.2 Dimension of the track feature descriptor	21		
5.1.3 Neighborhood size for Spectral clustering	25		
5.2 Evaluation of the proposed method	26		
5.2.1 Comparison with other studies	29		
5.2.2 Comparison of F_1 score	30		
5.3 Visualization of clustering results using t-SNE method	32		
5.4 Error analysis	35		
5.5 Constraints	36		
5.6 Time requirements	39		
5.7 Outlier detection	39		
6 Conclusions	43		
6.1 Future work	44		

Figures

<p>3.1 Occurrence of identities in Buffy the Vampire Slayer databases. 8</p> <p>3.2 Histogram of tracks identities in Harry Potter 1 and 2 databases 9</p> <p>4.1 General framework of our method. The tracking algorithm is not part of our work, but is added for the purpose of clarity. 13</p> <p>4.2 Detail of VGG-Face network architecture 15</p> <p>5.1 Bounding box size comparison of the variance for extracted facial images. 20</p> <p>5.2 Examples of extracted images using different bounding box sizes. 21</p> <p>5.3 Impact of dimension reduction of features on the variance lost by the reduction. 21</p> <p>5.4 Mean squared error of PCA transformation given number of components. 22</p> <p>5.5 Visualization of F_1 score given number of components when using Spectral Clustering on KNN manifold with $K=7$ for VGG-Face and $K=6$ for RESNET50 and Combined feature extraction methods 23</p> <p>5.6 Visualization of F_1 score given number of components when using Hierarchical agglomerative clustering (average linkage with cosine metrics). 24</p> <p>5.7 Visualization of F_1 score given number of components when using K-Means clustering. 24</p> <p>5.8 Visualization of dependence of F_1 score on size of k-neighborhood in spectral clustering. 25</p> <p>5.9 Visualization of identities in BF5 using Combined features (Section 4.3.3). The labels (colors) are ground truth. 32</p>	<p>5.10 Visualization of Spectral clustering results with the track size based on the number of faces inside that track. The color coding corresponds to the legend in Figure 5.9. The top diagram shows the ground truth labelling, the middle diagram shows clustering result of our method and in the lowest diagram we show the misclassified tracks (in red). 33</p> <p>5.11 Results of all three clustering algorithms. The color coding corresponds to the legend in Figure 5.9. 34</p> <p>5.12 Histogram to visualize length distribution of the well classified, misclassified, and of all tracks. . . . 35</p> <p>5.13 All misclassified tracks on BF5 dataset. In columns A there are representatives of incorrectly clustered tracks, column B shows identities to whom our method clustered the tracks from column A. Column C shows the ground truth identities to whom they should have been clustered. 37</p> <p>5.14 Visualization of the impact of utilizing constraints on BF2 dataset. The color coding corresponds to the legend in Figure 5.9. On all three diagrams the lines depict the tracks that have been misclassified, but whose misclassification could have been avoided using the cannot-link constraints. The top diagram shows the ground truth labelling, the middle shows our clustering result, and the bottom diagram shows all the misclassified tracks in red. . . . 38</p>
---	--

5.15	Outlier detection using Logistic Regression. This visualization is created using PCA to reduce the features of the representatives to 31 dimensions, after that the t-SNE method is used to produce 2 dimensional coordinates for each track. The outliers are shown in red, the tracks containing faces are blue. The positions of the tracks are the same in both pictures, the result is expressed in color-coding of the tracks.....	39
5.16	Examples of facial images selected by Logistic Regression as faces, yet according to database annotation they are outliers.	40
5.17	Distribution of features extracted using RESNET50. The red line corresponds to features extracted from outlier, i.e. image not containing any face. The blue line corresponds to features extracted from a face image. Detailed zoom is shown in Figure 5.18.	41
5.18	Detail of graph in Figure 4.6. The red line corresponds to features extracted from outlier, i.e. image not containing any face. The blue line corresponds to features extracted from a face image. The most interesting here is that most of the feature coordinates belonging to the outlier are 0, that is, the corresponding neurons in the last layer were not excited.....	41

Tables

2.1	caption	5
3.1	Statistics of datasets. Variance is calculated for the number of tracks per identity.....	7
5.1	Summary of results of parameters optimization on Buffy the Vampire Slayer S05E05, where F_1 stands for F_1 score (3.2) and NC for number of components for PCA.	22
5.2	Results of k size optimization for k -nearest neighbor manifold in Spectral clustering. F_1 stands for F_1 score of the resulting clusters and K stands for the size of k inside the manifold.	25
5.3	caption	26
5.4	caption	27
5.5	caption	28
5.6	Rank of the first 10 algorithms in terms of their evaluation score with respect to F_1 , Accuracy and WCP.	29
5.7	Summary of the results on all tested datasets using our proposed method.	29
5.8	Comparison with the state-of-the-art in terms of Accuracy on BF2 dataset. The works written in blue use convolutional neural networks in their method.....	30
5.9	Comparison with state-of-the-art in terms of B-Cubed Precision, B-Cubed Recall and F_1 score on HP1 database.	30
5.10	Comparison of WCP on BF2 and BF5 dataset. NC stands for number of clusters.	31



Chapter 1

Introduction

Video Face Clustering concerns itself with grouping of faces occurring in a video to disjoint subsets according to their visual characteristics and/or their spatial-temporal constraints. The ultimate goal of Video Face Clustering is to be able to identify different identities in videos. The Video Face Clustering has many applications, for example, it is possible to do an interaction analysis of characters in a movie, compute identity occurrence statistics in videos from surveillance cameras, or it enables creation of large face datasets [35].

In this work we focus on face clustering in movies and TV series. In this scenario the Video Face Clustering method requires a lot of computing power due to the large amount of data that needs to be processed. Besides the computational demands, the face clustering is challenging due to substantial variation of a character appearance during the movie (or TV episode) caused by changes of light conditions, make-up, or even the change of the character's age (e.g. in Harry Potter movies [15] used as one of our benchmarks). In contrast to processing of still face images, the faces extracted from videos are often more affected by motion blur, occlusions and image compression making the Video Face Clustering even more challenging.

During the last years it has become clear that the Video Face Clustering methods using handcrafted features for face representation [36][35] are outperformed by recent works that employ deep learning for feature extraction [40, 30]. The deep learning has been popularized by the work of Krizhevsky et al. [17] whose Convolutional Neural Network (CNN) trained from millions of examples outperformed traditional computer vision approaches in the task of image classification by outstanding margin. Since then the methods based on deep learning have become the state-of-the-art in many other computer vision tasks including face recognition. The video clustering method proposed in this thesis also exploits deep CNNs trained from millions of examples.

This thesis attempts to develop a state-of-the-art video clustering method assembled using contemporary open source implementations of computer vision and machine learning algorithms. The input of our method is a set of face tracks, that is, sequences of faces extracted from a video by a common face tracker. The method outputs clusters of the face tracks each corresponding to a single identity. The proposed method itself is composed of three main building blocks. Firstly, we extract feature descriptor for each face in the track.

Secondly, we describe each track by a single feature obtained by aggregating feature descriptors previously extracted from individual faces of the track. Thirdly, the track feature descriptors are grouped by a clustering algorithm. We have systematically tested several different implementations of the three building blocks in order to find the best performing setup. In particular, we considered VGG-Face [26], RESNET50 [16] for face feature extraction, template selection and median representation for feature aggregation, and K-Means, Hierarchical clustering [23] and Spectral clustering algorithm [38, 34] to find clusters. We evaluated our method using evaluation protocols widely used in the community. Namely we used the B-Cubed Precision/Recall and classification Accuracy [3, 30] to measure the clustering precision obtained by the method on episodes of the TV series Buffy the Vampire Slayer [11] and the Accio dataset [15] composed of Harry Potter movies. The results showed that our method, which is a well tuned simple framework assembled from publicly available libraries, achieved state-of-the-art accuracy while able to process the videos in real-time on a standard PC.

The proposed method was evaluated in a commonly used scenario which assumed that the input video tracks did not contain outliers, i.e. tracks containing non-faces or faces of more than one identity. To get such input, one needs either a perfect tracker or a reliable detector of the outlier tracks. We propose a simple detector which uses the extracted feature descriptors and Logistic Regression. A preliminary empirical evaluation presented in this thesis shows promising results.

The other output of this thesis is an open source implementation of the proposed method and the evaluation framework written in Python. The code is well documented and modular in order to allow easy testing of different implementations of individual components of the method.

The thesis is divided into six chapters. Chapter 2 consists of a short description of the related works. Chapter 4 describes the proposed method and details of its individual components. In Chapter 3 we describe the datasets used as our benchmarks. Chapter 5 is devoted to experiments, evaluation against state-of-the-art and visualization of the results. Conclusions and future work are given in Chapter 6.

Chapter 2

Related work

2.1 Video Face Clustering methods

This chapter provides a brief overview of video clustering methods. The overview is not meant to be exhaustive, instead it concentrates only on the works most related to the proposed approach.

Traditionally, the face clustering techniques were aiming at still images, such as in photo albums, where it was used to order the images according to person appearance [42, 13]. To this end, it was essential to develop a function measuring similarity between two face images, so that the similarity of images of the same identity is high while the similarity of images of different identities is low. Having the similarity function defined, multitude of generic-purpose clustering algorithms can be used. Development of the face similarity functions has a long history. Originally, the face similarity was measured by Euclidean distance between vectors representing linear projection of pixel intensities of the compared face images [22]. The next era covering more than a decade of research was devoted to development hand-crafted feature descriptors tailored to the face images, e.g. [19, 5]. Finally, after the advent of deep learning the face similarity is typically implemented by Convolutional Neural Networks learned from examples [37].

The methods for face clustering in still images are not directly applicable to videos for several reasons. The faces extracted from video have often a lower resolution and they are affected by motion blur, occlusions or compression artifacts. In addition, in the video track clustering we have additional prior knowledge about the temporal and spatial information from the face tracks which should be incorporated into the clustering algorithm.

Analogically to clustering in the still images, the first works on video face clustering were using hand-crafted features for face representation. Among the historically first are the methods proposed in [12, 13]. For example, the work [13] proposes the Joint Manifold Distance (JMD) which measures the distance between two sub-spaces defined by pixel intensities extracted from the faces of tracks to be compared. The tracks are grouped by agglomerative clustering. The work [10] proposes an unsupervised method for learning face similarity function which was then applied in a hierarchical clustering algorithm to obtain the track clusters. The faces are represented by SIFT

features computed around a set of automatically detected landmarks. The positive example pairs needed to learn the similarity function are faces occurring at the same track, while the negative example pairs are selected from tracks that appear at the same time in the scene hence capturing different identity. The main advantage of the method is its ability to adapt the face similarity to the processed images without the need to annotate the tracks. The authors of [20] propose the Penalized Probabilistic Clustering algorithm which allows to incorporate soft as well as hard preference constraints on pairs of tracks that should be (or should not be) assigned to the same cluster. The work [35] proposes a generative clustering model based on the Hidden Markov Random Fields. Application of the sparse vector clustering algorithm to cluster faces extracted from video is proposed in [36]. This method simultaneously clusters faces into tracks and tracks into clusters while it allows to incorporate prior knowledge on which faces should appear in the same track and which tracks should not be assigned to the same cluster. Method of the same flavour have been proposed in [24, 8]. Both methods use RGB pixel intensities as image features.

The most recent works rely on features extracted by deep CNN. Namely, in [40] the authors formulate a joint face representation adaptation and clustering approach in a deep learning framework. The method allows face representation to gradually adapt from an external source domain to a target video domain without using any strong supervision. The authors of [39] explore the visual constraints among the face tracks to generate online training samples and then learn CNN with triplet-loss.

The closest work to the approach proposed in this thesis is the paper [30] which uses the same framework composed of three steps: feature extraction by CNN trained on still images, feature aggregation and clustering. In contrast to them, we use more sophisticated features (we use combination of VGG-Face and RESNET50 instead of only VGG-Face), different aggregation method (we use median aggregation instead of averaging) and different clustering algorithm (we use spectral clustering instead of K-means). We show experimentally that our implementation leads to a significant improvement of the clustering accuracy regardless which evaluation metric is used.

2.2 Evaluation metrics

It is interesting that until today there is not a strong consensus on which evaluation metrics should be used to evaluate the face clustering algorithms. Table 2.1 shows which evaluation metrics were applied in papers describing methods we use as baselines in this thesis.

When evaluating the clustering algorithms we usually have two agnostic criteria in mind i) cluster purity (i.e. each cluster should contain only one identity) and ii) a spread of the identities over different clusters (i.e. one identity should be assigned to one cluster only).

One of the most frequently used evaluation metrics is the B-Cubed metrics family [3] developed in 1998 which enables balancing the number of clusters

with their purity using two counter-posing metrics. Firstly, the B-Cubed Precision penalizes classification of two different identities as one but does not take into consideration the number of clusters. Secondly, the B-Cubed Recall which penalizes for not putting two tracks of the same identity together in one cluster. In order to express these two metrics by a single number, their harmonic mean, so called F_1 score, is usually used. The F_1 score has been also recommended already in 1974 by Van Rijsbergen [28] as a standard way of combining two different metrics.

The other metric is the classification Accuracy which is computed from the confusion matrix between the predicted cluster labels and the ground truth labels under most optimistic permutation. A disadvantage of the Accuracy metric is that it requires the tested method to return the correct number of clusters. Despite its disadvantages this metric is most frequently used in literature.

Other metrics are the Cluster Purity, also referred to as the Weighted Cluster Purity (WCP), and the Cluster Inverse Purity [21, 31]. The two metrics have similar meaning as the B-Cubed Precision and Recall, however, they have less favourable theoretical properties [3].

We believe that the B-Cubed metrics will be used much more widely on the expense of the others because they were chosen as the main evaluation criteria for clustering algorithms in the recent NIST Challenge [14].

Metrics / Article	DFR [30]	JFAC [40]	CSR [41]	VFSC [24]	WBSLRR [36]	HMRF [35]	ULDML [10]
B-cubed precision	NO	YES	NO	NO	NO	NO	NO
B-cubed recall	NO	YES	NO	NO	NO	NO	NO
F_1 score	NO	YES	NO	NO	NO	NO	NO
Accuracy	YES	YES	YES	YES	YES	YES	YES
WCP	YES	NO	NO	NO	NO	NO	NO

Table 2.1: Table of metrics used in state-of-the-art articles on Face Clustering in Videos which we used as baselines in our empirical evaluation. The referred methods are Discriminant Metric Learning (ULDML) [10], Hidden Markov Random Field based Clustering (HMRF) [35], Weighted Block-Sparse Low Rank Representation clustering (WBSLRR), Very Fast Sparse Clustering [24], Joint Face Adaptation Representation clustering method (JFAC) [40], Deep Face Representation clustering (DFR) [30] and Constrained Sparse Clustering (CSR) [41]

Chapter 3

Databases and evaluation protocols

3.1 Summary of databases

In our work we use two different databases with four videos in total. In all the videos, there is a huge difference between the minimum and maximum number of tracks per identity, resulting in problems with using various clustering techniques relying on at least comparably large resulting clusters. Both databases were originally created for a slightly different purpose yet they have become the standard benchmark of video face clustering used by the computer vision community. The much older Buffy the Vampire Slayer [11] (2006) dataset was created for the demonstration of the benefit gained by using face and cloth matching to the use of automatically aligned subtitles and script text for identity labelling [11]. The Harry Potter (Accio) database [15] was created for the investigation of automatic face matching of the identity across age.

In the databases, a track is defined as a sequence of positions of bounding boxes and their frame numbers.

Database	Number of tracks	Number of identities	Avg. number of tracks per identity	Maximum tracks per identity	Minimum tracks per identity	Variance
Harry Potter 1	5249	36	145	2006	2	362
Harry Potter 2	5335	42	127	1874	2	318
Buffy S05E02	2088	19	109	982	6	212
Buffy S05E05	1532	16	96	655	2	160

Table 3.1: Statistics of datasets. Variance is calculated for the number of tracks per identity.

3.1.1 Buffy database

Buffy the Vampire Slayer database [11] consists of two annotated episodes, 2 and 5 from season 5. In our work, we use the episode 5 as our training set, therefore all parameter setting and/or tuning is done on this dataset. Later in this work, we will refer to Buffy the Vampire Slayer S05E02 as to BF2 and to Buffy the Vampire Slayer S05E05 as to BF5.

The face detection for this database was done using Boosted Cascade of Simple Features detector [33]. Then, the tracks were created using spatial-temporal locations of the detected faces. The labelling of the tracks was done manually. The labeled tracks fall into three categories: a name of one of the main characters, false-positive for tracks not containing any facial image, and "other" for any tracks containing facial images with unknown person (i.e. someone other than the main character). The histogram of identity occurrence in this database is shown in Figure 3.1.

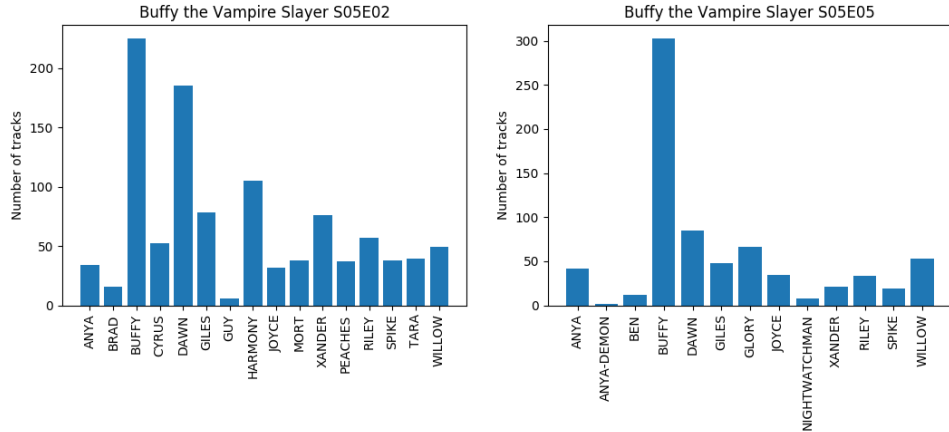


Figure 3.1: Occurrence of identities in Buffy the Vampire Slayer databases.

3.1.2 Accio database

Accio [15] database consists of 8 annotated movies, Harry Potter 1-8, from whose we use only 1 and 2 for evaluation purposes as only the two episodes have been used in the relevant literature. This dataset is much more challenging than the Buffy dataset [11] due to its higher number of tracks, identities, and a lot of dark scenes throughout the movies. Later in this work, we will refer to the Harry Potter databases as to HP1 and HP2.

Detection and tracking used for the creation of this database was a Particle filter tracker method proposed in [4], which also indicated initial labelling. The final labelling was done manually and consist of two main options: a name of one of the known characters and a false-positive for the tracks with images containing everything else but the known identities. The histogram of identity occurrence for this database is shown in Figure 3.2

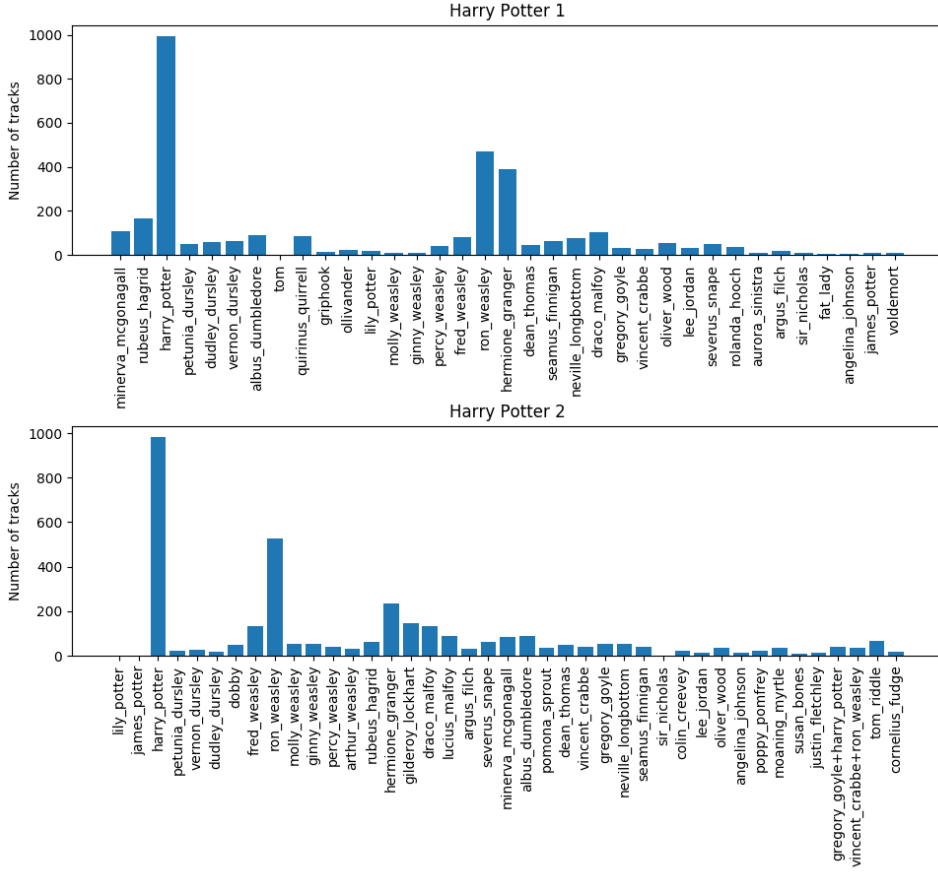


Figure 3.2: Histogram of tracks identities in Harry Potter 1 and 2 databases

3.2 Evaluation protocols

Let $y \in \mathcal{Y} = \{1, \dots, Y\}$ denote the identity of a person captured in the track described by a feature $x \in \mathcal{X}$. Let $h: \mathcal{X} \rightarrow \mathcal{C} = \{1, \dots, C\}$ be a clustering strategy that assigns tracks to clusters. The clusters correspond to the identities.

Let $\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} \mid i = 1, \dots, m\}$ be a test set containing pairs of tracks along with their true y . Let further denote by $\mathcal{I}_y = \{i \in \{1, \dots, m\} \mid y^i = y\}$ a set containing indices of test tracks with the identity y . Given a strategy h , we can assign each test track x^i to a cluster $h(x^i)$. Let us represent the cluster assignment by sets $\hat{\mathcal{I}}_c = \{i \in \{1, \dots, m\} \mid h(x^i) = c\}$, $c \in \mathcal{C}$, containing indices of test tracks assigned by h to the respective cluster.

In most experiments (except when comparing WCP score in Table 5.10) we assume that the correct number of clusters C is known a priori. This assumption has been used in all the related works to which we compare our results, namely, in [10, 20, 35, 24, 40, 30].

■ B-cubed Precision

B-cubed precision is the average fraction of the track pairs assigned to a cluster with matching class labels. The B-cubed precision is defined as:

$$\text{Precision}(h) = \frac{1}{m} \sum_{i=1}^m \frac{1}{|\hat{\mathcal{I}}_{h(x^i)}|} \sum_{j \in \hat{\mathcal{I}}_{h(x^i)}} \llbracket y^i = y^j \rrbracket. \quad (3.1)$$

For example, the B-cubed precision will be 1 if we put every track into its own cluster. In case that in all clusters there are 2 different identities (labels) the precision will 0.5 and so on.

■ B-cubed recall

B-cubed recall is an average fraction of the track pairs belonging to the same class which assigned to the same cluster. It says how many clusters there are per label on average. The B-cubed recall is defined as

$$\text{Recall}(h) = \frac{1}{m} \sum_{i=1}^m \frac{1}{|\mathcal{I}_{y^i}|} \sum_{j \in \hat{\mathcal{I}}_{y^i}} \llbracket h(x^i) = h(x^j) \rrbracket. \quad (3.2)$$

For example, if all identities are distributed over 4 clusters (each identity in 4 different clusters) than the recall will be 0.25. The recall will be 1 if we put all tracks in one cluster.

■ F₁ score

F₁ score is defined as the harmonic mean of the B-cubed recall and the B-cubed precision:

$$F_1(h) = 2 \frac{\text{Precision}(h)\text{Recall}(h)}{\text{Precision}(h) + \text{Recall}(h)} \quad (3.3)$$

■ Accuracy

The classification Accuracy (as defined in [30]) is calculated from the confusion matrix between predicted and ground truth classes for tracks and is given in percentage of how many predictions were correct. The accuracy is measured on track-level and it requires the knowledge of the number of true classes (clusters).

For the calculation of accuracy it is necessary to find an optimal matching between the predicted labels and the true labels by solving a linear assignment problem. In our work, this is done using an Scipy [25] implementation of the Hungarian Algorithm [18].

The Accuracy is defined as

$$\text{Accuracy}(h) = 100 \min_{\pi \in \Pi} \left(\frac{1}{m} \sum_{i=1}^m \llbracket y^i \neq \pi(\hat{y}^i) \rrbracket \right) \quad (3.4)$$

where Π stands for all possible permutations between the assigned and true labels.

Cluster Purity and Weighted Cluster Purity

Weighted Cluster Purity (WCP) and Cluster Purity are mathematically equivalent. Their value is analogical to the B-Cubed Precision, because it covers the cases where tracks with different labels are put into one cluster. The Cluster Purity (and Weighted Cluster Purity) is defined as

$$\text{Purity}(h) = \sum_{c \in \mathcal{C}} \frac{|\hat{\mathcal{I}}_c|}{m} \left(\max_{y \in \mathcal{Y}} \frac{|\hat{\mathcal{I}}_c \cap \mathcal{I}_y|}{|\hat{\mathcal{I}}_c|} \right) = \frac{1}{m} \sum_{c \in \mathcal{C}} \max_{y \in \mathcal{Y}} |\hat{\mathcal{I}}_c \cap \mathcal{I}_y|. \quad (3.5)$$

The purity penalizes clusters with more identities but it ignores if tracks with the same label are put in different clusters. The purity is maximal, if there is no cluster containing 2 or more different identities.

Cluster Inverse-Purity

We do not report any results using Cluster Inverse-Purity in our work, but it is deemed necessary to mention it, because it is the counter-part of Cluster Purity. The Cluster Inverse-Purity is analogical to the B-Cubed Recall, because it deals with tracks of one label putted to multiple clusters. The Cluster Purity and Cluster Inverse-Purity metrics are dominant in works related to face clustering in still images, but they are seldom used in face clustering in videos.

The Cluster inverse-purity is defined as

$$\text{InvPurity}(h) = \sum_{y \in \mathcal{Y}} \frac{|\mathcal{I}_y|}{m} \left(\max_{c \in \mathcal{C}} \frac{|\hat{\mathcal{I}}_c \cap \mathcal{I}_y|}{|\mathcal{I}_y|} \right) = \frac{1}{m} \sum_{y \in \mathcal{Y}} \max_{c \in \mathcal{C}} |\hat{\mathcal{I}}_c \cap \mathcal{I}_y|. \quad (3.6)$$

The inverse-purity rewards if faces with the same labels are put to the same cluster but it ignores if the clusters contain more labels. The Inverse Purity is maximal, if there is no label in two or more clusters.

Chapter 4

Method

4.1 Framework

Our method takes as input a set of face tracks, which are sequences of faces extracted from a video by a common tracker. The output are clusters of the tracks, where each cluster corresponds to an identity. The method consists of three main parts: feature extraction from single face images using pre-trained CNN, feature aggregation for the track representation by a single feature, and clustering. We used a CNN trained on still images because the face recognition in still images is more developed - the CNNs for face recognition in still images are trained on databases whose size is at least a magnitude larger than the databases of annotated videos.

In our work we do not propose any method or approach for the face tracking. Our clustering method assumes that the tracks are provided and that they are correct, i.e. they contain only tracks that correspond to a single identity. However, at the end of this chapter, we propose a simple detector of outlier tracks.

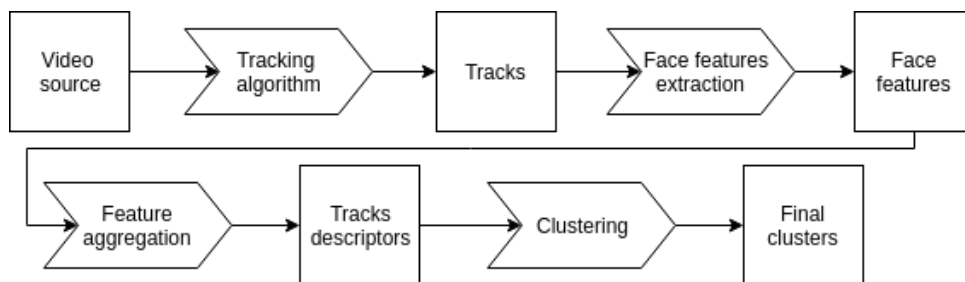


Figure 4.1: General framework of our method. The tracking algorithm is not part of our work, but is added for the purpose of clarity.

Figure 4.1 illustrates the main building blocks of the proposed method (including the tracking part). At the beginning there is the input video in which the tracking algorithm finds faces and merges them into tracks.

Afterwards, we extract features from every face image using one of the methods described in Section 4.3. We represent every track by a single feature vector, later referred to as track representative, by one of the two

aggregation methods described in Section 4.4. The last part of our framework is clustering into disjoint identities groups using one of the methods described in Section 4.5.

For evaluation of our method we used videos which already come with the face tracks [11] [15], i.e. sequences of bounding boxes. In Section 4.6, we propose a simple method for outlier detection.

4.2 Tracking

The input of tracking part is a video file and the output is a set of tracks, each consisting of any number of images (at least one image per track). As mentioned before, for our experiments we used annotated video sources. Our first dataset, Buffy the Vampire Slayer episodes 2 and 5 from the 5th series, was created using Boosted cascade of simple features detector [33] used for the detection. The organizing into tracks was done with the spatial-temporal location of the faces and the annotation was done manually.

The second dataset we use, Accio [15], was created by performing tracking-by-detection using a particle filter [4] with the use of multi-pose detectors for the covering of the entire range of *pan* and *roll* face angles.

4.3 Feature extraction

In our work we used several different approaches for feature extraction. The basis for all of them was a convolutional neural network - for our first two methods we used VGG-Face [26], for the third approach we used residual convolutional neural network RESNET50 [16] and for the fourth one we used combination of VGG-Face and RESNET50. We employed both of these networks using Keras-VGGFace [27], running on Keras [9] on Tensorflow backend [1]. The weights for these networks were the ones published by the authors of [26] and [16] respectively, only imported for the use in Keras framework. The CNNs used for the feature extraction required a fixed-size image on the input. For the best result, the face-area shown in the image had to be as close as possible to the data, on which the CNN was trained. Due to that, we had to resize the the bounding boxes of the detected faces to fit the requirements of the CNN. Details are provided in Experiments in Section 5.1.1.

4.3.1 VGG-Face

From this network we used responses of two different layers for feature extraction. In the first approach we used the response of the last layer (*fc8*) of VGG-Face network [26], resulting in 2,622 features per image. In the second approach we used the response of the *fc7* layer (it is two layers above the last layer). We used the *fc7* layer in order to be able to compare the results of this work with the work of the authors of [30]. This setting resulted in 4,096 features per image. The detailed architecture of the network is shown in Figure 4.2.

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	-	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	-	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	-	3	-	64	-	-	64	-	128	-	-	128	-	256	-	256	-	-	256
num flts	-	64	-	64	-	-	128	-	128	-	-	256	-	256	-	256	-	-	512
stride	-	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	1	2
pad	-	1	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	conv6	relu6	conv7	relu7	conv8	softmax
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	-	512	-	512	-	-	512	-	512	-	512	-	-	512	-	4096	-	4096	-
num flts	-	512	-	512	-	-	512	-	512	-	512	-	-	4096	-	4096	-	2622	-
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Figure 4.2: Detail of VGG-Face network architecture

4.3.2 RESNET50

In this approach we used the residual CNN RESNET50 [16] pretrained on the VGG-Face2 dataset [7]. The network model and weights were taken from the same paper [7]. The architecture of this model allowed it to be much deeper than the VGG-Face [26] without losing accuracy. It was expected that this network would perform better than VGG-Face, because it is the most recent work of the same authors and it is even trained on a superior dataset [7] compared to the one VGG-Face was trained on. This setting results in 2,048 features per image.

4.3.3 Combination of RESNET50 and VGG-Face

We planed on using this combination at the beginning, the idea to make it was born while we were doing experiments on the feature extraction approaches. We combined the features obtained by RESNET50 and VGG-Face by simply appending VGG-Face (responses from *fc7* layer) features to RESNET50, which resulted in total of 6,144 features per image.

4.4 Feature aggregation

By feature aggregation is meant the aggregation of the sequence of face features extracted for all images in one track into a single feature describing the whole track. The resulting feature set (later referred to as track representative) has the same number of features as a single picture. This can be achieved using

various techniques, in our work we tested the Template selection and the Median computation.

As shown in Section 5.1, it is desirable to perform dimension reduction on the set of the track representatives. For this task we used standard Principal Component Analysis (PCA) to project the features to space with significantly lower dimension. Details are provided in Section 5.1.

■ 4.4.1 Template selection

Template selection means selecting one of the face images in the the track as a track representative. We evaluated the median value m (coordinate-wise) for the features of images inside one track. After that, we selected (as the track representative) the L2 norm of the feature having the smallest euclidean distance to the median value m .

■ 4.4.2 Median computation

Here we computed L2 norm of all features of images contained in one track and then computed the coordinate-wise median value m for the features of images inside that track. This median value m was then the representative of the track.

■ 4.5 Clustering algorithms

The input of this clustering part are the track representatives reduced to a feature space with lower dimension by PCA. The output are clusters of tracks, each cluster corresponding to one identity. In our work, we tested three different clustering algorithms - Hierarchical clustering using average linkage, K-Means and Spectral clustering on k -nearest neighbors (KNN) manifold.

■ 4.5.1 Hierarchical clustering

We used agglomerative (bottom-up) approach. We used average linkage clustering algorithm [23] with cosine metrics - the distance between two faces is defined as the cosine value of the angle between them from the origin of the coordinates. The distance between two clusters, 1 and 2, is defined as the average pairwise distance between tracks in cluster 1 and tracks in cluster 2. At the beginning, the algorithm marks all tracks as clusters containing a single track. At every step, it takes two clusters having the smallest distance between each other and puts them together in one cluster. Therefore, after each step the number of clusters is decreased by one. It stops when it reaches the desired number of clusters.

■ 4.5.2 K-Means clustering

K-Means [2] object is to partition the examples into k clusters. The initialization of the algorithm may vary depending on implementation, in our work

we seeded k random points having the same dimension as our feature space, referred to as centroids. After that, the initial assignment of every track to a cluster represented by its nearest centroid is made. Each step has two parts: in the first one the coordinates of every centroid are recalculated, so that the within-cluster variance is minimized (the centroids are put in the center of the cluster) and in the second one the tracks whose centroid has changed are put to the cluster belonging to their closest centroid. The stopping criterion is when the coordinates of the centroids are not changed (or the change is below given threshold) in the first part of the step. The main problem of this algorithm is the necessity of multiple runs as it is very prone to stopping at local optimum. The objective function is to minimize the within-cluster sum of squares, formally the objective is:

$$\arg \min_{\mu_1, \dots, \mu_k} \sum_{i=1}^m \min_{j=1, \dots, k} \|x_i - \mu_j\|^2 \quad (4.1)$$

where μ_j is the centroid of cluster j .

■ 4.5.3 Spectral clustering

The spectral clustering algorithm [38][34] is capable of exploiting more information (than the Hierarchical clustering and K-Means), because the clustering itself is done not in the original space, but in a transformed space, a kernel. In our work we used k -nearest neighbor kernel. The run of the algorithm consists of three main parts: firstly, creating a similarity matrix - we used k -nearest neighbor method, the size of k was a tuning parameter for us (the k selection is in Section 5.1). The second step is the computation of the first k eigenvectors of the Laplacian matrix to define a feature vector for every track, and the last step is to run K-Means on those features.

After the creation of similarity matrix, we detected if the resulting graph had more than one connected component. If it did, we increased the k by one and repeat the similarity matrix creation. (This detection step was omitted on experiments on our training dataset, because for this dataset we used precisely tuned k (see Figure 5.8))

■ 4.6 Outlier detection

In this section we used our track representation to perform outlier detection. For this purpose, we used the Logistic Regression to separate the tracks to outliers and to tracks composed of images containing only faces. This means, our detector does not deal with tracks containing multiple identities, only the ones without any facial images, i.e. track containing one or more. The Logistic Regression model was fitted on BF5, our training database.

Chapter 5

Experiments

We have structured this chapter into seven main parts: hyper-parameter tuning in Section 5.1, evaluation of the proposed method in Section 5.2, visualization of the obtained results in Section 5.3, error analysis for the detection of the weak part of our method in Section 5.4, proposition of the cannot-link constraints incorporation and corresponding visualizations in Section 5.5, time requirement analysis in Section 5.6, and proposition of outlier detection based on our track representation in Section 4.6.

In Section 5.1, devoted to hyper-parameter tuning, we show that the reduction of the features dimension significantly improves the clustering results. We then find the optimal settings for all combinations of our feature extraction techniques (see Section 4.3) with all selected clustering algorithms (see Section 4.5).

In Section 5.2, with all the parameters set, in the evaluation part we test the different feature extraction - clustering algorithm combinations on 4 datasets, BF2 and BF5 [11] and on HP1 and HP2 from the Accio dataset [15]. We then compare our results to the state-of-the-art on Buffy the Vampire Slayer S05E02 dataset in terms of the accuracy and F_1 score.

In the visualization part, we employ t-SNE [32] method for dimension reduction to 2. Then we plot the results of our method compared to ground truth and visualize the differences between the individual clustering algorithms.

In Error Analysis section we show a simple track-length based misclassification analysis. The constraints section shows the possible benefits emerging from the incorporation of the cannot-link constraints and we show the related visualizations. In Time requirements section we describe the computational demands of our method. In the last section we propose a new, very successful approach for outlier detection with the use of our track description.

5.1 Hyper-parameter tuning

Our method has three hyper-parameters, tuning of which is presented in this section: the bounding box size for the input of the CNN, the number of components for the PCA dimension reduction and the size k of the k -nearest neighborhood kernel used in the Spectral clustering algorithm. The hyper-parameters were tuned on the BF5 selected for training.

5.1.1 Bounding-box size

Given the general framework, at first it was necessary to extract facial images from the videos. Frame number and position, where the face was located, was given by the databases we have used in this work - for Buffy the Vampire Slayer it was [11] and for Harry Potter it was [15]. For the actual image extraction, we have used methods implemented in OpenCV library [6].

Both databases were created using different face-tracking algorithm (as mentioned in Section 4.2), their bounding box sizes (area selected to contain the desired face) had different settings. In order to unify that we extracted faces in 10 different sizes, beginning from 10% larger than the original settings and going up by 10% up to 2 times the original size (as shown in Figure 5.2) - this has been done for a large part of the BF5 dataset. After that we extracted features from all images using the VGG-Face [26] and median computation (Section 4.4.2) for feature aggregation. Given these feature-vectors, we calculated the variance between vectors belonging to the same identity (further referred to as Inner variance), then medians of the vectors for every identity and afterwards the variance between these medians (further referred to as Outer variance). Our goal was to minimize the Inner/Outer variance ratio, as this shows how good the selected features are in describing the facial characteristics of individual identities. In other words this means fine-tuning the bounding box size for the VGG-Face [26] neural network.

In Figure 5.1 we show the results of this test. In our work we used the 150% resizing ratio.

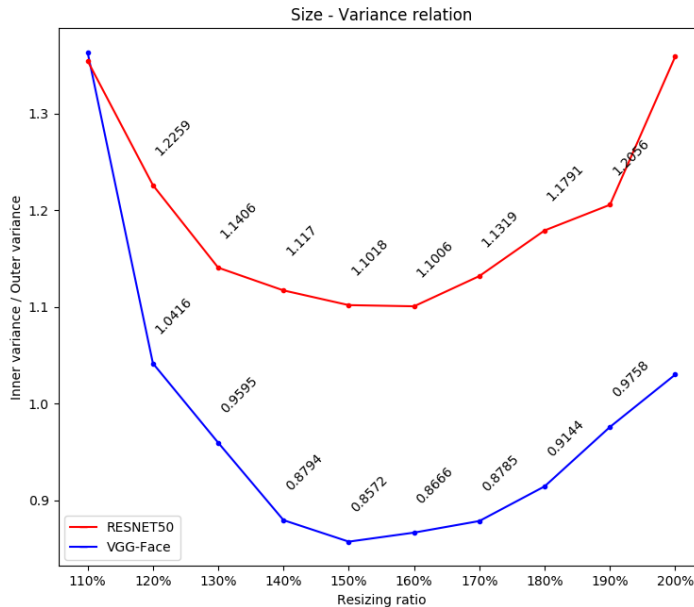


Figure 5.1: Bounding box size comparison of the variance for extracted facial images.



Figure 5.2: Examples of extracted images using different bounding box sizes.

5.1.2 Dimension of the track feature descriptor

As shown in Figure 5.3 and in Figure 5.4, it is unnecessary to use all dimensions of the feature space. We propose selecting only certain number of dimensions by the means of PCA using our training dataset BF5 to maximize the class separability in terms of the F_1 (Section 3.2) score using Hierarchical clustering, K-Means and Spectral clustering on features extracted by VGG-Face (Section 4.3.1), RESNET50 (Section 4.3.2) and Combined (4.3.3) networks. The results of this optimization are shown in Figure 5.6, Figure 5.7 and in Figure 5.5.

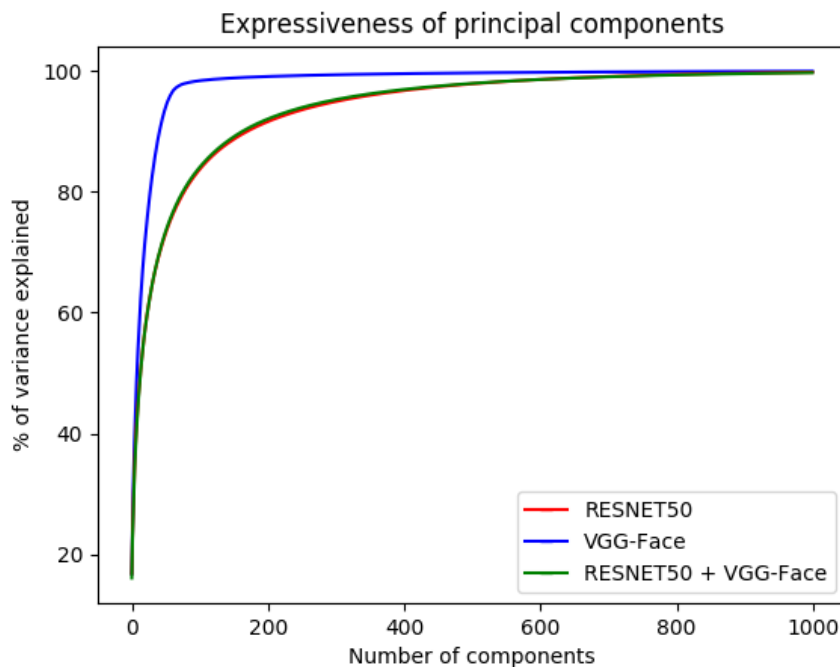


Figure 5.3: Impact of dimension reduction of features on the variance lost by the reduction.

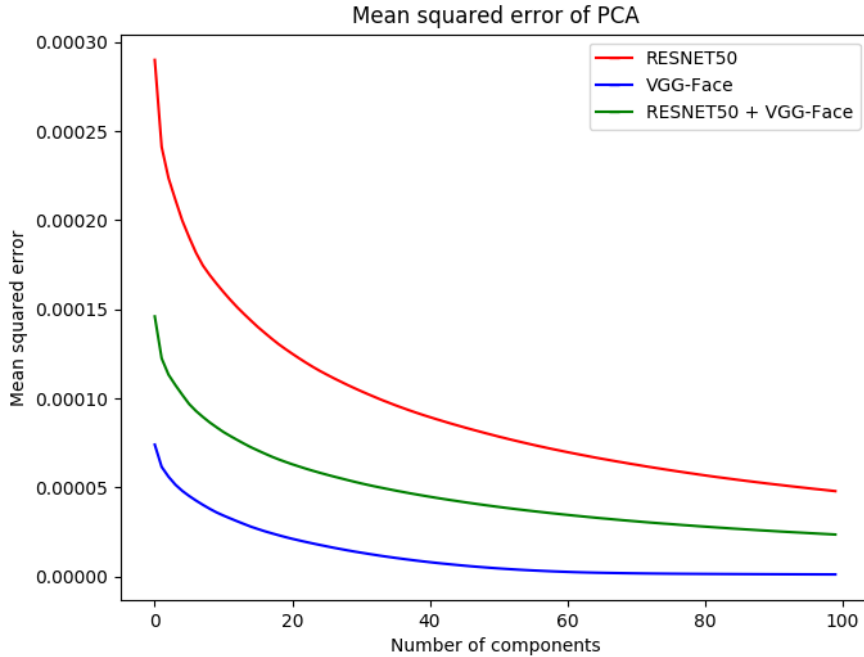


Figure 5.4: Mean squared error of PCA transformation given number of components.

Table 5.1 summarizes the best results for all clustering algorithms - feature extraction combinations from experiments in Figure 5.6, Figure 5.7 and from Figure 5.5. It is clear that the Spectral clustering performs superior to the other clustering techniques regardless of feature extraction method used. It is also probable that the results on the training dataset may suffer from over-fitting, as can be seen mainly from Figure 5.6, but partially also from Figure 5.5 due to the oscillation in the results.

Algorithm	VGG-Face		RESNET50		Combined	
	F ₁	NC	F ₁	NC	F ₁	NC
Hierarchical clustering	0.77	131	0.85	201	0.80	71
K-Means	0.81	32	0.77	32	0.83	26
Spectral clustering	0.90	21	0.95	35	0.95	31

Table 5.1: Summary of results of parameters optimization on Buffy the Vampire Slayer S05E05, where F₁ stands for F₁ score (3.2) and NC for number of components for PCA.

The results showed very significant changes in the optimal number of components for different feature extraction methods. The most substantial changes were for the Hierarchical clustering, where the optimal dimension was 131 for VGG-Face, 201 for RESNET50 and 71 for the Combined extraction method. Most interestingly, the use of combination of the features from VGG-Face and RESNET50 gave the best results using significantly less dimensions

than any of VGG-Face and RESNET50 alone. The same applies for the K-Means clustering algorithm, where both the VGG-Face and RESNET50 performed optimally on 32 dimensions, while their combination reached the best score on 26 dimensions. This observation did not apply to Spectral clustering, because Spectral clustering performed best on 21 dimensions for VGG-Face, 35 for RESNET50 and 31 for the Combined extraction method.

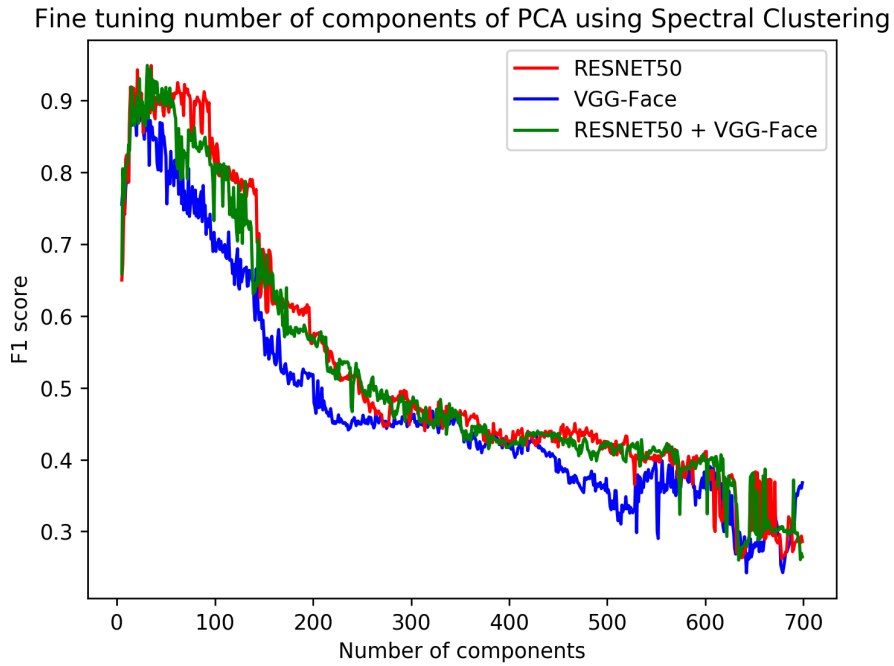


Figure 5.5: Visualization of F_1 score given number of components when using Spectral Clustering on KNN manifold with $K=7$ for VGG-Face and $K=6$ for RESNET50 and Combined feature extraction methods

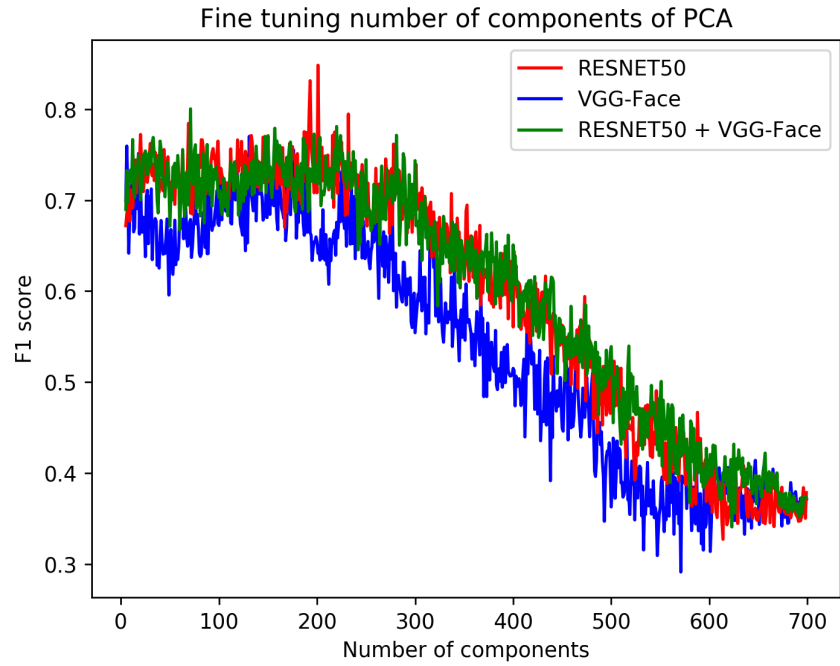


Figure 5.6: Visualization of F_1 score given number of components when using Hierarchical agglomerative clustering (average linkage with cosine metrics).

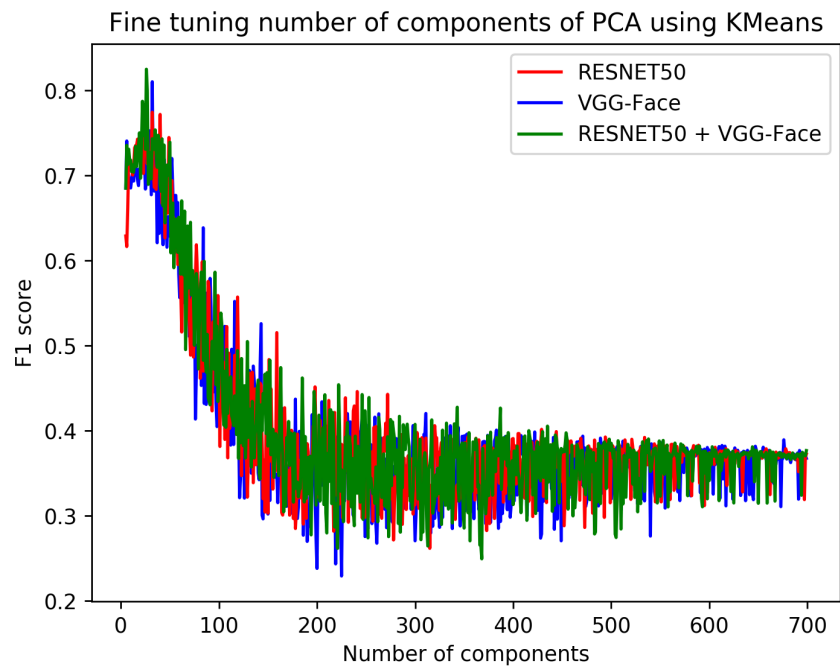


Figure 5.7: Visualization of F_1 score given number of components when using K-Means clustering.

5.1.3 Neighborhood size for Spectral clustering

Spectral clustering algorithm is the only algorithm - from the three we are using - requiring additional parameters then just the number of clusters: due to the use of k -nearest neighbors kernel (manifold), we need to select appropriate size of k for every feature extraction method. However, with selection of the desirable size of k arises the need to perform dimension reduction with the newly selected k . We use a block-wise ascent approach towards this problem. As step a), we fix the k (we have started with k equal to 30 for all feature extraction methods) and we find the optimal number of components for PCA. Then, as step b), we fix the number of components and select optimal k , then repeat from step a). The dimension optimization results shown in Figure 5.5 and the k size neighborhood optimization results are the last step a) and b), when both the number of components and size of k already converged to the results reported in Table 5.2 and in Table 5.1 on the last row, respectively.

Algorithm	VGG-Face		RESNET50		Combined	
	F ₁	K	F ₁	K	F ₁	K
Spectral clustering	0.90	7	0.95	6	0.95	6

Table 5.2: Results of k size optimization for k -nearest neighbor manifold in Spectral clustering. F₁ stands for F₁ score of the resulting clusters and K stands for the size of k inside the manifold.

The results show the k does not differ much with different feature extraction methods. For VGG-Face, the optimal size of k is 7 and for both the RESNET50 and the Combined extraction method the optimal size of k is 6.

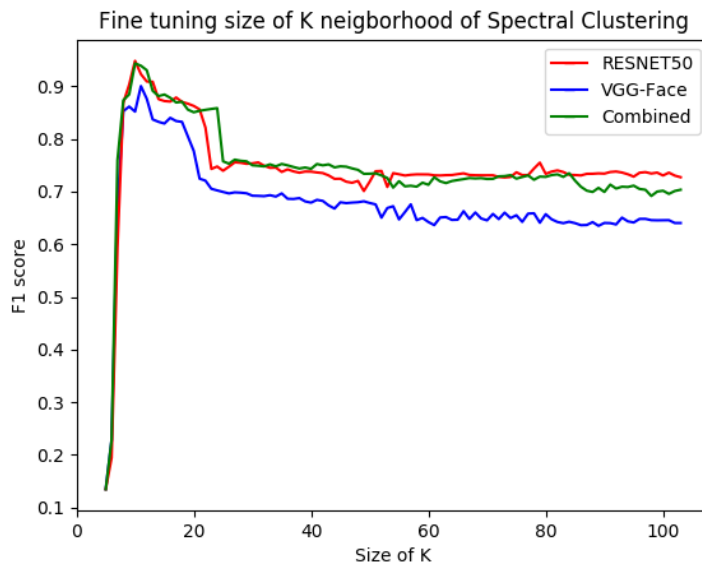


Figure 5.8: Visualization of dependence of F₁ score on size of k -neighborhood in spectral clustering.

5.2 Evaluation of the proposed method

Tables 5.3, 5.4 and 5.5 show the results for all combinations of clustering algorithm (Section 4.5), feature aggregation (Section 4.3) and feature aggregation (Section 4.4) sorted from the most successful one to the least successful one. For every evaluation metrics (F_1 score, Accuracy and WCP) there is one table and in all the tables the testing dataset is printed in red. All results were generated using the proposed setting unique to each clustering algorithm - feature extraction combination shown in Table 5.1.

Algorithm combination	Dataset				Rank
	BF2	BF5	HP1	HP2	
Spectral clustering-combined-median	0.86	0.95	0.79	0.71	1.
Spectral clustering-resnet50-median	0.82	0.95	0.77	0.66	2.
Spectral clustering-combined-template	0.81	0.87	0.75	0.65	3.
Hierarchical-resnet50-median	0.79	0.85	0.63	0.68	4.
Spectral clustering-resnet50-template	0.77	0.86	0.74	0.65	5.
Spectral clustering-vgg <i>fc7</i> -median	0.84	0.9	0.71	0.56	6.
Spectral clustering-vgg <i>fc7</i> -template	0.74	0.85	0.61	0.66	7.
Hierarchical-combined-median	0.73	0.8	0.63	0.67	8.
Spectral clustering-vgg ll-median	0.69	0.84	0.72	0.65	9.
K-Means-combined-template	0.76	0.75	0.57	0.61	10.
K-Means-combined-median	0.76	0.78	0.55	0.61	11.
Hierarchical-vgg <i>fc7</i> -median	0.71	0.77	0.58	0.59	12.
Hierarchical-combined-template	0.68	0.71	0.61	0.65	13.
Hierarchical-resnet50-template	0.7	0.69	0.61	0.64	14.
Spectral clustering-vgg ll-template	0.61	0.84	0.66	0.56	15.
K-Means-resnet50-median	0.75	0.76	0.52	0.58	16.
K-Means-vgg <i>fc7</i> -median	0.75	0.75	0.56	0.55	17.
Hierarchical-vgg <i>fc7</i> -template	0.67	0.68	0.55	0.63	18.
K-Means-vgg <i>fc7</i> -template	0.67	0.77	0.54	0.58	19.
Hierarchical-vgg ll-median	0.65	0.66	0.58	0.61	20.
K-Means-resnet50-template	0.73	0.75	0.52	0.56	21.
Hierarchical-vgg ll-template	0.63	0.66	0.51	0.56	22.
K-Means-vgg ll-median	0.62	0.66	0.5	0.53	23.
K-Means-vgg ll-template	0.53	0.67	0.49	0.51	24.

Table 5.3: Algorithms comparison in terms of F_1 score. The ordering is based on the average rank computed over the datasets. The results on the training dataset are printed in red.

Algorithm combination	Dataset				Rank
	BF2	BF5	HP1	HP2	
Spectral clustering-combined-median	89.4	96.7	81.2	68.8	1.
Spectral clustering-resnet50-median	86.8	96.7	79.4	60.6	2.
Spectral clustering-combined-template	83.9	87.9	77.8	62.2	3.
Spectral clustering-resnet50-template	80.4	86.2	76.3	65.3	4.
Hierarchical-resnet50-median	83.1	88.2	61.3	64.4	5.
Spectral clustering-vgg ll-median	72.7	86.9	75.3	65.6	6.
Spectral clustering-vgg fc7-median	88	93.9	73.2	46.5	7.
Hierarchical-combined-median	77.4	82.8	58.2	61.1	8.
Spectral clustering-vgg fc7-template	76.5	88.9	62.3	52.7	9.
Hierarchical-vgg fc7-median	76.4	81.3	52.4	62.8	10.
Spectral clustering-vgg ll-template	60	88.4	71	54.8	11.
K-Means-combined-median	81.6	79.5	49.9	53.5	12.
K-Means-combined-template	79.8	72.8	53.7	55.8	13.
Hierarchical-resnet50-template	73.9	66	63.5	57.9	14.
Hierarchical-combined-template	73.3	66.9	58.3	58.5	15.
Hierarchical-vgg fc7-template	74.3	65.1	52.7	61.2	16.
K-Means-vgg fc7-median	79.1	78	48.9	53	17.
Hierarchical-vgg ll-median	70.5	61.8	54.9	55.8	18.
K-Means-resnet50-median	80.1	71.8	46	48.8	19.
K-Means-resnet50-template	77.3	68.4	47.5	49.4	20.
K-Means-vgg fc7-template	69.2	77.4	48.2	52.2	21.
Hierarchical-vgg ll-template	69.6	63.7	46.3	51.3	22.
K-Means-vgg ll-median	66.3	60.8	46.4	47.8	23.
K-Means-vgg ll-template	54.6	63.4	45.5	45.4	24.

Table 5.4: Algorithms comparison in terms of Accuracy. The ordering is based on the average rank computed over the datasets. The results on the training dataset are printed in red.

Algorithm combination	Dataset				Rank
	BF2	BF5	HP1	HP2	
Spectral clustering-combined-median	0.93	0.98	0.9	0.88	1.
Spectral clustering-resnet50-median	0.91	0.98	0.88	0.88	2.
Spectral clustering-combined-template	0.88	0.92	0.87	0.87	3.
Hierarchical-resnet50-median	0.87	0.92	0.87	0.87	4.
Spectral clustering-vgg <i>fc7</i> -median	0.9	0.95	0.88	0.83	5.
K-Means-combined-median	0.87	0.92	0.85	0.86	6.
K-Means-combined-template	0.86	0.91	0.86	0.85	7.
K-Means-vgg <i>fc7</i> -median	0.87	0.91	0.85	0.85	8.
Spectral clustering-resnet50-template	0.85	0.91	0.86	0.85	9.
Spectral clustering-vgg <i>fc7</i> -template	0.83	0.93	0.84	0.85	10.
Hierarchical-combined-median	0.85	0.91	0.84	0.86	11.
K-Means-resnet50-median	0.85	0.93	0.83	0.84	12.
K-Means-resnet50-template	0.83	0.94	0.81	0.83	13.
K-Means-vgg <i>fc7</i> -template	0.81	0.9	0.84	0.84	14.
Spectral clustering-vgg ll-median	0.79	0.9	0.85	0.83	15.
Hierarchical-resnet50-template	0.8	0.88	0.84	0.85	16.
Hierarchical-combined-template	0.78	0.91	0.83	0.84	17.
Hierarchical-vgg <i>fc7</i> -median	0.8	0.89	0.83	0.84	18.
Spectral clustering-vgg ll-template	0.73	0.92	0.83	0.79	19.
Hierarchical-vgg ll-median	0.8	0.88	0.81	0.82	20.
Hierarchical-vgg <i>fc7</i> -template	0.81	0.88	0.78	0.73	21.
K-Means-vgg ll-median	0.78	0.88	0.8	0.8	22.
K-Means-vgg ll-template	0.7	0.89	0.79	0.78	23.
Hierarchical-vgg ll-template	0.77	0.86	0.77	0.79	24.

Table 5.5: Algorithms comparison in terms of WCP. The ordering is based on the average rank computed over the datasets. The results on the training dataset are printed in red.

The final algorithm ranking (based on the results for the F_1 score, Accuracy and WCP, shown in Table 5.3, Table 5.4 and Table 5.5) is shown in Table 5.6. Based on the obtained results we can draw the following conclusions:

- The Spectral clustering algorithm is consistently superior to the other clustering algorithms regardless of the feature extraction and aggregation method.
- The best feature extraction method is the Combined method regardless of the clustering algorithm and aggregation method.
- The best aggregation method is the median computation (regardless of the feature extraction method and clustering algorithm).
- According to the results in Table 5.6, it can be seen that the three metrics, i.e. F_1 , WCP and Accuracy, are consistent in the sense that they lead to the same best (up to the third) configuration.

Algorithm combination / Rank	F ₁	Accuracy	WCP
Spectral clustering-combined-median	1	1	1
Spectral clustering-resnet50-median	2	2	2
Spectral clustering-combined-template	3	3	3
Hierarchical-resnet50-median	4	5	4
Spectral clustering-vgg <i>fc7</i> -median	6	7	5
Spectral clustering-resnet50-template	5	4	9
Spectral clustering-vgg <i>fc7</i> -template	7	9	10
Hierarchical-combined-median	8	8	11
K-Means-combined-median	11	12	6
Spectral clustering-vgg last layer-median	9	6	15

Table 5.6: Rank of the first 10 algorithms in terms of their evaluation score with respect to F₁, Accuracy and WCP.

We used the metrics evaluated on training dataset (see Tables 5.3, 5.4 and 5.5) to select the best configuration of the three components (note that the same best configuration was selected regardless the used metric):

- Face feature descriptor: Combination of RESNET50 and VGG-Face
- Aggregation method: Median
- Clustering algorithm: Spectral clustering

The performance of the best configuration evaluated on the test datasets is summarized in Table 5.7. As mentioned in Section 5.1, our method slightly over-fits to the training dataset, but the extremely high scores on the training dataset may be partially caused by the fact that training dataset BF5 is the simplest one concerning the number of tracks and light conditions.

Dataset	Precision	Recall	F ₁ score	Accuracy	WCP
BF2	0.87	0.85	0.86	89.4	0.93
BF5	0.95	0.95	0.95	96.7	0.98
HP1	0.86	0.73	0.79	81.2	0.90
HP2	0.84	0.67	0.71	68.8	0.88

Table 5.7: Summary of the results on all tested datasets using our proposed method.

■ 5.2.1 Comparison with other studies

In this section we compare our work to the state of the art in terms of Accuracy and WCP using BF2[11] dataset and in terms of B-Cubed precision, B-Cubed recall and their F₁ score using HP1 dataset [15]. In Table 5.8 we show the results of our method compared to the Unsupervised Logistic Discriminant Metric Learning (ULDML) [10], Penalized Probabilistic Clustering (PPC) [20], Hidden Markov Random Field based Clustering (HMRF) [35], Weighted Block-Sparse Low Rank Representation clustering (WBSLRR), Very Fast

Sparse Clustering [24], Joint Face Adaptation Representation clustering method (JFAC) [40], and Deep Face Representation clustering (DFR) [30]. It is important to mention that these studies tuned their methods using the BF2 dataset, but we used it only as a testing dataset.

The results for PPC, ULDMML and HMRF, shown in Table 5.8, are reported in [24] - in this paper the VFSC method was presented. In Table 5.8 all the results (except possibly for JFAC and DFR, because the authors did not report details of the evaluation metrics) are for the Accuracy measured per track, i.e. percentage of the misclassified tracks (as defined in Section 3.2). This setting was in [24] named Acc. 2, whereas the Accuracy measured per image was named Acc. 1. For completeness, the result of our method measured using the Accuracy per image (Acc. 1 in [24]) was 95.69%.

■ Comparison of clustering accuracy

Method	Year	Accuracy
PPC [20]	2007	40.11 \pm 4.03
ULDML [10]	2011	48.28 \pm 0.00
HMRF [35]	2013	48.52 \pm 3.49
WBSLRR [36]	2014	62.76 \pm 1.10
VFSC [24]	2016	66.37 \pm 0.00
JFAC [40]	2016	92.13 \pm 0.90
DFR [30]	2017	87.46 \pm 0.00
Our method	2018	89.40 \pm 0.00

Table 5.8: Comparison with the state-of-the-art in terms of Accuracy on BF2 dataset. The works written in blue use convolutional neural networks in their method.

■ 5.2.2 Comparison of F_1 score

Algorithm	Precision	Recall	F_1 score
JFAC [40], 2016	0.69	0.35	0.46
Our method	0.86	0.73	0.79

Table 5.9: Comparison with state-of-the-art in terms of B-Cubed Precision, B-Cubed Recall and F_1 score on HP1 database.

In Table 5.9 we show the comparison of our method with the state-of-the-art Joint Face Representation Adaptation (JFAC) [40] method in terms of the F_1 score on the HP1 from the Accio dataset. This method is the number one in terms of Accuracy on Buffy the Vampire Slayer dataset, but on Accio dataset our method is significantly superior. We outperform the JFAC method by 33% in F_1 score, which is a really big improvement considering that the JFAC is - to our best knowledge - the most successful method for clustering in videos so far.

■ Comparison of weighted cluster purity

Interestingly, none of the works devoted to this topic reported WCP for the desired number of clusters (identities). Only two [31] [30] reported this score, but for a very high number of clusters.

Method	BF2		BF5	
	NC	WCP	NC	WCP
DFR [30]	688	0.965	575	0.946
HAC-Neg [31]	688	1.000	575	0.999
Ours	688	1.000	575	0.997
DFR [30]	598	0.950	507	0.946
SLC [31]	598	1.000	507	0.998
Ours	598	0.999	507	0.997

Table 5.10: Comparison of WCP on BF2 and BF5 dataset. NC stands for number of clusters.

In Table 5.10 we compare our Hierarchical agglomerative approach (on VGG-Face *fc7*) with the SIFT based Fisher encoding using Hierarchical Agglomerative Clustering with Negative Constraints (HAC-Neg) [31], Scene-level clustering (SLC) [31] and the corresponding results from [30], both of which use the same method, only with different number of clusters (NC). The number of clusters for each episode was set in [31] for the purpose of demonstration of their method. The authors of [30] only used this number of clusters for comparability to [31], as we did. It is important to mention, that the number of clusters here was irrationally high, because the total number of tracks containing faces was 1067 for BF2 and 727 for BF5, while the number of identities was 16 for BF2 and 13 for BF5.

5.3 Visualization of clustering results using t-SNE method

Figure 5.9 shows the visualization of BF5, the colors correspond to identities in the legend of the visualization. The x and y locations of the tracks were obtained via two steps from features extracted using the combined settings (Section 4.3.3), the first one was PCA used to select 31 components with the largest variance, the second was t-SNE [32] method, reducing it effectively from 31 to 2. The number 31 for the number of components selected by PCA was chosen because we were using the same data obtained from PCA for clustering by Spectral clustering in Figure 5.10, where we show the results. The clustering accuracy of the depicted result was 96.7%, meaning that only 24 tracks were misclassified. We used the track length-based size of the tracks to point out that our method misclassified mainly tracks containing one or two faces - the track-length based distributions are shown later, in Figure 5.12 in Section 5.4.

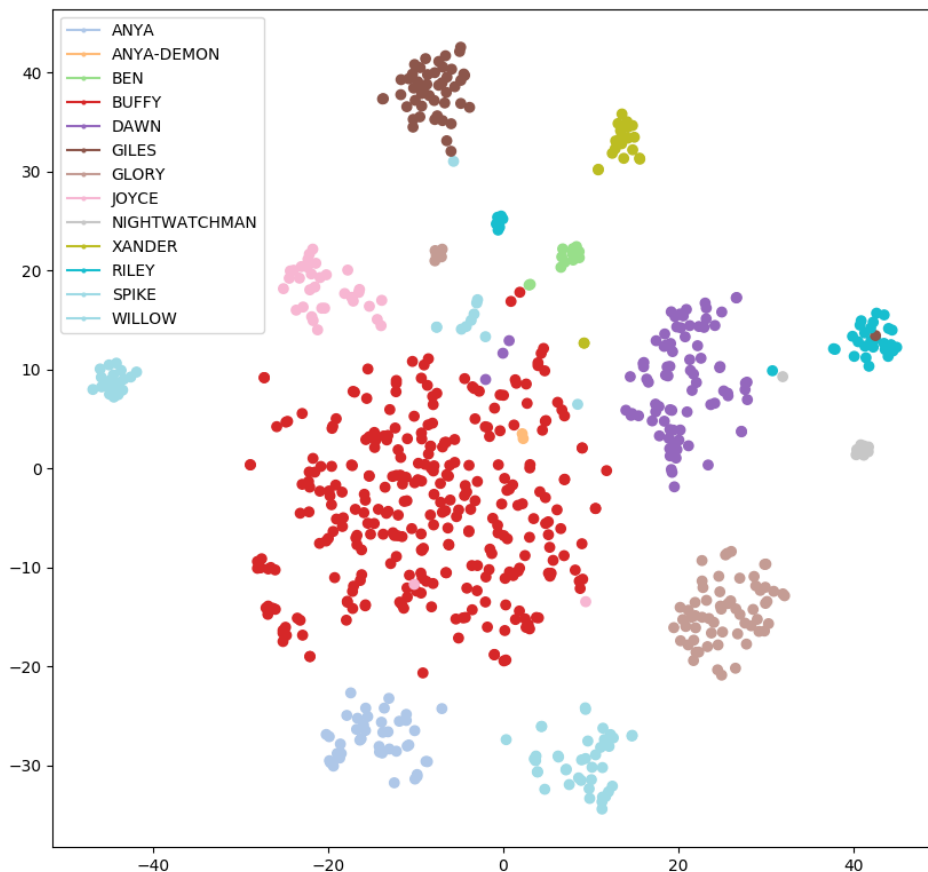


Figure 5.9: Visualization of identities in BF5 using Combined features (Section 4.3.3). The labels (colors) are ground truth.

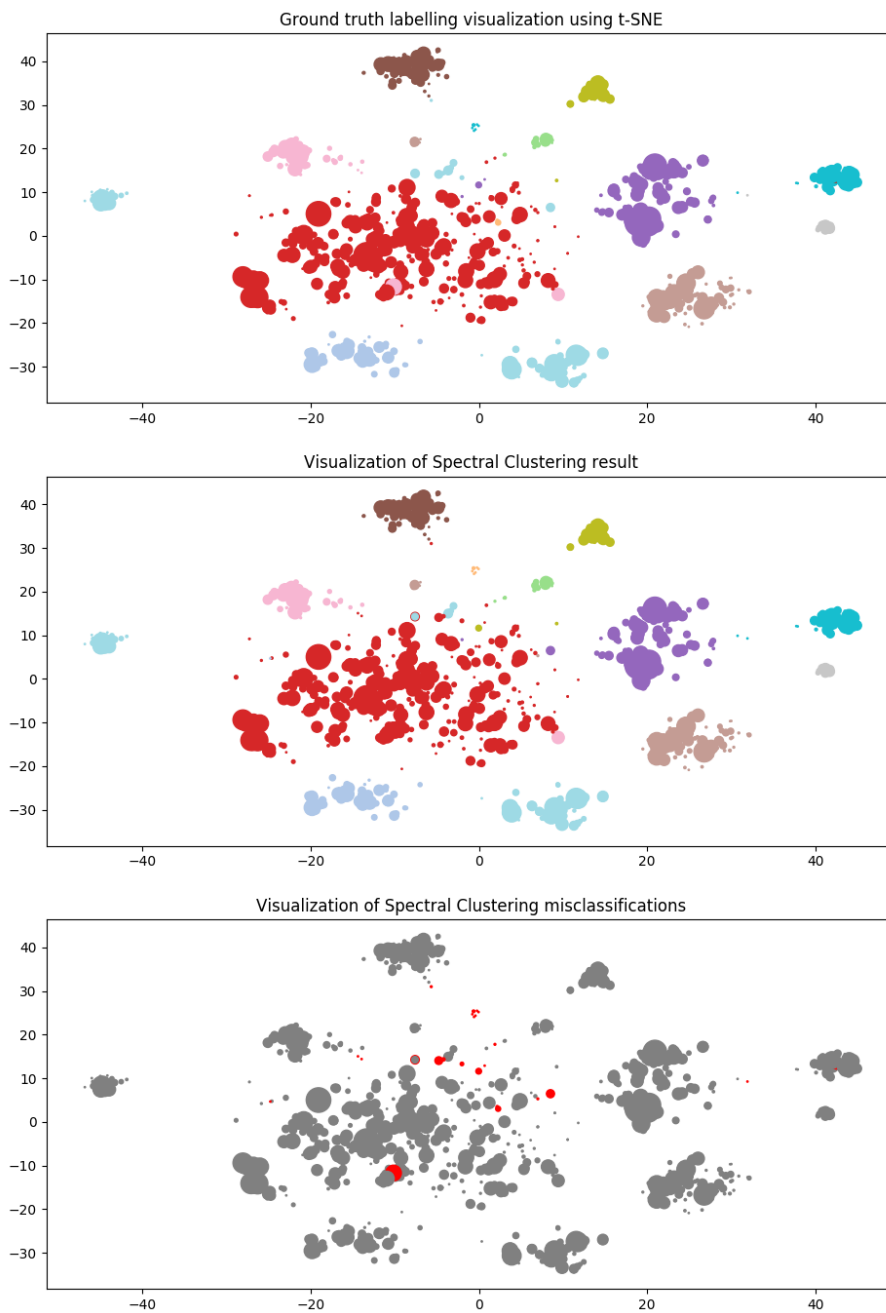


Figure 5.10: Visualization of Spectral clustering results with the track size based on the number of faces inside that track. The color coding corresponds to the legend in Figure 5.9. The top diagram shows the ground truth labelling, the middle diagram shows clustering result of our method and in the lowest diagram we show the misclassified tracks (in red).

In Figure 5.11 we show the differences between the Hierarchical clustering, K-Means and Spectral clustering, while using the same track positions (layout) as in Figure 5.10. This was done for all these graphs in order to be visually comparable to each other. The clustering accuracy was obviously much lower

in the first two approaches, namely for the Hierarchical clustering the accuracy was 70.7% and for K-Means it was 60%. From Table 5.4 it is obvious that the accuracy for both of them could be slightly higher, but we ran them on the same number (31) of components as the Spectral clustering.

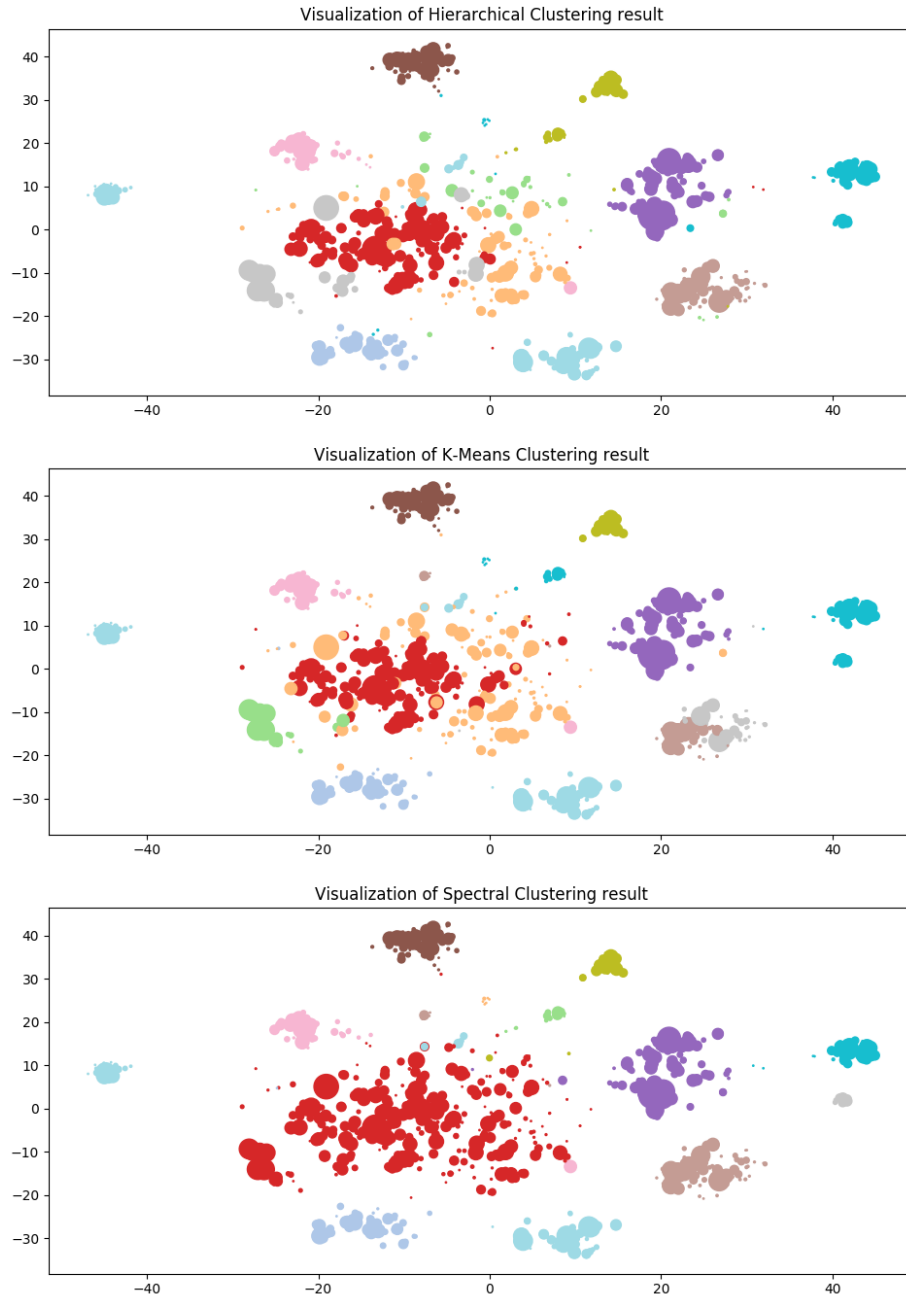


Figure 5.11: Results of all three clustering algorithms. The color coding corresponds to the legend in Figure 5.9.

5.4 Error analysis

We performed a series of tests to analyze our misclassification errors and to propose improvements to our method that would eliminate significant percentage of the errors in the future. The implementation of these improvements is beyond the scope of our work. In Figure 5.12 we show the track-length distributions of errors, correctly classified, and all tracks for the BF2 movie. On the x-axis is the track length and on the y-axis is the number of occurrences. It is obvious that most errors are made on the single face tracks but even without normalization it is clear that all the distributions are almost the same, so it is not possible to conclude any results based only on the track length.

Nevertheless, the probability of misclassification of the track is smaller when the number of facial images in the track is bigger, so for our method it would be desirable to set the tracker to prefer longer tracks. That naturally increases the probability of having outliers inside the face tracks, but this is solved by our feature aggregation method.

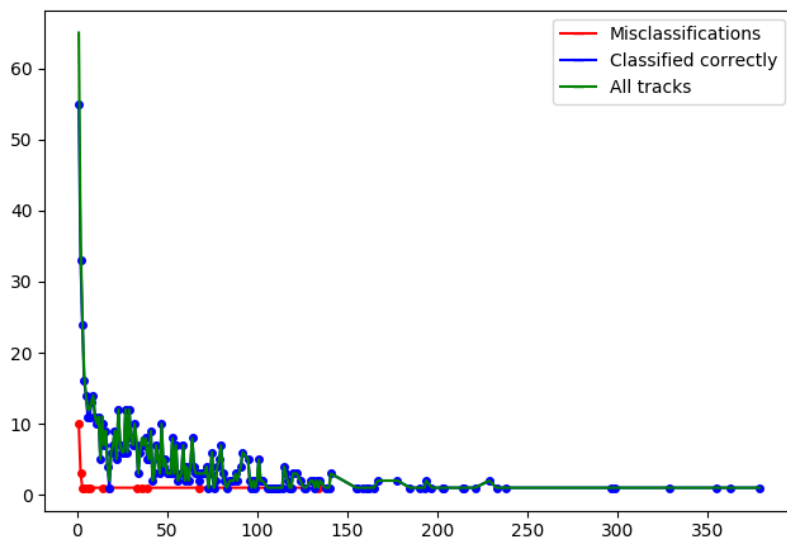


Figure 5.12: Histogram to visualize length distribution of the well classified, misclassified, and of all tracks.

In Figure 5.13 we show all errors, i.e. incorrect classifications of tracks, made by our method on BF5 dataset. In the columns A there are the representatives of the incorrectly clustered tracks, column B shows identities into which our method clustered the tracks from column A, column C shows the ground truth identities whom they should have been clustered to. On the left side there are 6 incorrect classifications of the same person (a guy (Ben) classified as a woman with lot of makeup (Anya-demon)). This happened,

because Anya-demon identity has only six tracks in the entire dataset and the features representing her are closer to the features extracted for Buffy identity (a girl shown in four middle pictures on the left side, one picture from bottom) than these six tracks of Ben are to the features belonging to any other identity, including his own. This can be seen right beneath these six incorrect classifications, where the tracks of Anya-demon are merged by our algorithm together with Buffy. On the right side of the Figure 5.13 it is obvious that our method has troubles correctly recognizing blurred, low-resolution images.

In summary there are two types of errors: i) low quality images caused by low resolution, bad light conditions and/or extreme head-pose and ii) errors is the database annotation.

■ 5.5 Constraints

In our work we did not use the spatial-temporal constraints because we opted for general-purpose clustering algorithms. These constraints are obtained from overlapping tracks in frames, i.e. the tracks occur at the same time, as used e.g. in [35] [41] [8]. In this section we estimate the potential drop in the performance due to neglecting these constraints by our method.

In BF2 there are total of 1282 pairwise constraints (the cannot-link constraints are symmetrical - if there is a constraint between track A and track B then there is a constraint between track B and track A, therefore the total number of one-way constraints is 2564). In Figure 5.14 we show only the constraints relevant to this clustering. By relevant we mean constraints between track A and track B, where our method erroneously clustered the track A and B into one cluster, but due to the existing constraints this merging should not have been permitted. If we were to incorporate these constraints into algorithm, the optimal improvement would be near 16% decrease in errors, meaning 1.9% improvement in the Accuracy score. The 16% is the upper bound, because it means we place all the incorrectly clustered (and now divided) tracks to their true cluster. The cannot-link constraints are most useful in movies where most of the time there are more main actors on scene simultaneously, such as in many sitcom series.



Figure 5.13: All misclassified tracks on BF5 dataset. In columns A there are representatives of incorrectly clustered tracks, column B shows identities to whom our method clustered the tracks from column A. Column C shows the ground truth identities to whom they should have been clustered.

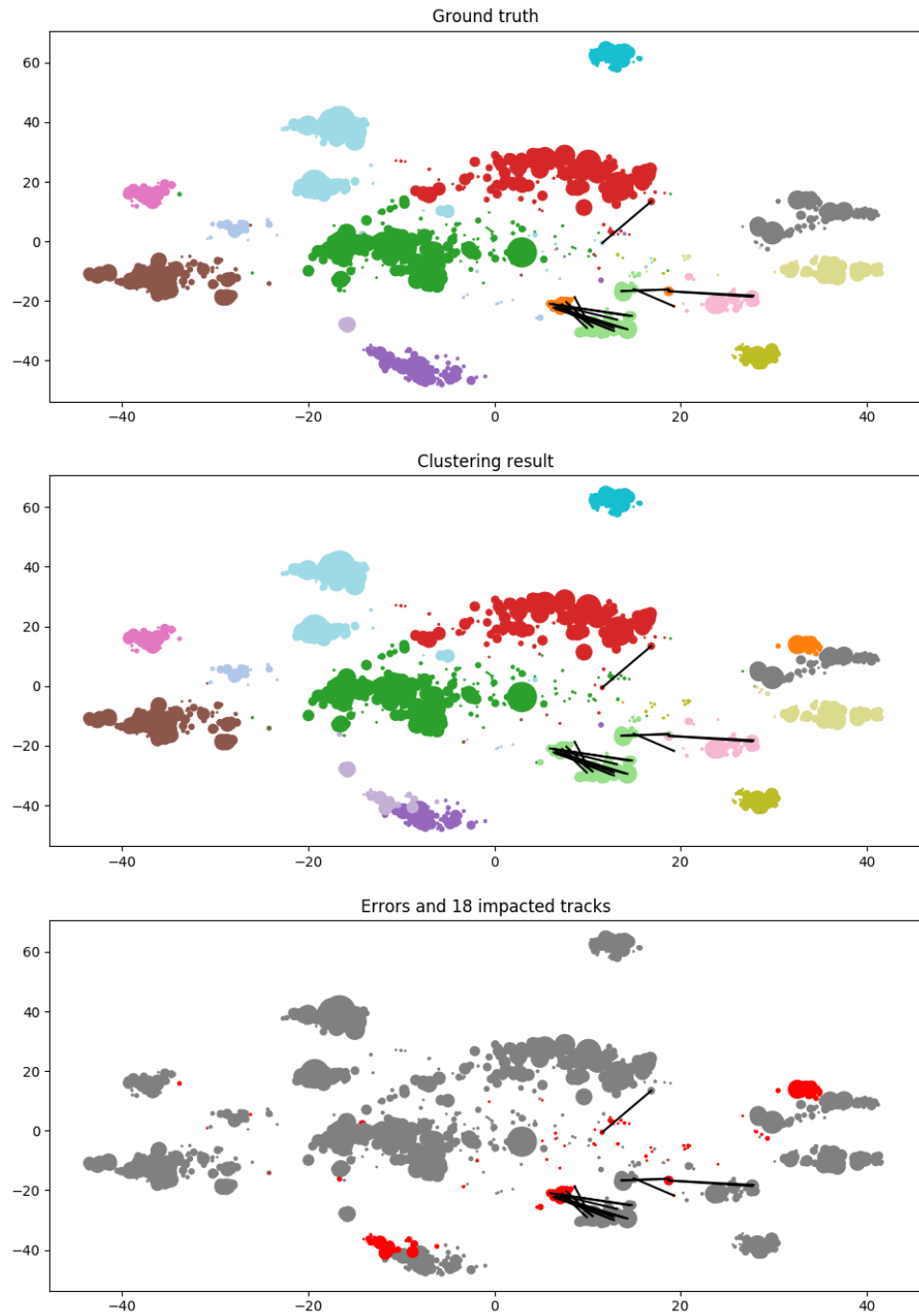


Figure 5.14: Visualization of the impact of utilizing constraints on BF2 dataset. The color coding corresponds to the legend in Figure 5.9. On all three diagrams the lines depict the tracks that have been misclassified, but whose misclassification could have been avoided using the cannot-link constraints. The top diagram shows the ground truth labelling, the middle shows our clustering result, and the bottom diagram shows all the misclassified tracks in red.

5.6 Time requirements

If a video is encoded in .mp4 format, such as a movie, our method takes approximately 42 minutes for 45 minutes of video footage on medium range graphic card - we used NVidia Quadro M4000 with 8GB dedicated memory. Video decompression and extraction of face images takes approximately 30 minutes. Feature extraction from the images takes 12 minutes for the same movie with the same card. The final feature reduction and clustering takes generally 1.5 seconds on Intel Core i7 7700 processor, which is a negligible timespan.

As stated above most of the time is used for image and feature extraction. These can be parallelized to shorten the total amount of running time. The clustering itself takes only fraction of the time, 1.5 seconds for BF5 dataset.

Based on the above, we imply that our method runs approximately the same time as is the length of the video it is run on.

5.7 Outlier detection

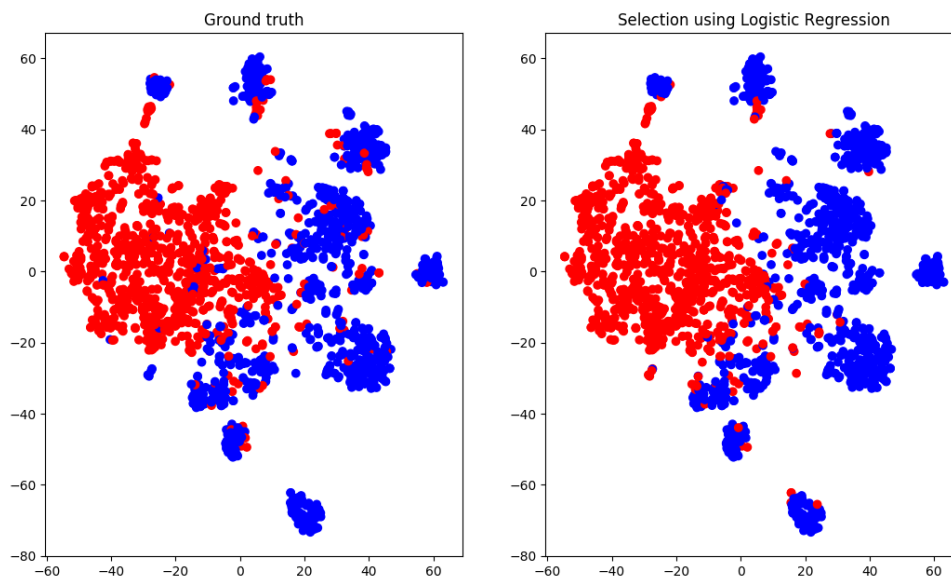


Figure 5.15: Outlier detection using Logistic Regression. This visualization is created using PCA to reduce the features of the representatives to 31 dimensions, after that the t-SNE method is used to produce 2 dimensional coordinates for each track. The outliers are shown in red, the tracks containing faces are blue. The positions of the tracks are the same in both pictures, the result is expressed in color-coding of the tracks.

The outlier detection was tested on BF2 dataset. The Logistic Regression model was fitted on our training dataset, BF5. The graphs in Figure 5.15 show the result of this method - the position of individual tracks was computed

using t-SNE method from our track representatives reduced by PCA to 31 dimensions. The color coding depicts the outlier tracks in red and tracks with face images in blue. The left diagram is labelled using the ground truth, the right one using the Logistic Regression model.

Using this method, a very high score was achieved, although there was an issue in the database itself, as it contained multiple tracks annotated as outliers, where in fact these tracks contained easily recognizable facial images, as shown in Figure 5.16. According to the Buffy the Vampire Slayer database annotations, tracks containing images in Figure 5.16 are outliers, or specifically the annotation says "false positive", meaning the hit of the used face tracker on these was wrong. Based on this observation, our method might perform in fact much better than the results written below, because faces on these images are clearly recognizable. Therefore on large number of the tracks our method was technically not wrong, even if the database annotation said so.



Figure 5.16: Examples of facial images selected by Logistic Regression as faces, yet according to database annotation they are outliers.

In Figure 5.17 we show the distribution of the coordinates in features for a track representatives of two tracks: an outlier track (shown red) and a track containing face images (shown blue), both extracted using RESNET50. As better seen in the zoomed-in image in Figure 5.18, most of the coordinates of the outlier feature representation were 0 meaning that the corresponding neurons in the layer (whose response we used) were not excited. It seems the distributions for the outlier track and for the face track may be distinguishable by some statistical approach based on the frequency of not-excited neurons in the layer whose responses we observed. For the features of the outlier track even applied that the median value across all coordinates was 0, while for a face track it was not. In this work we do not propose such method, but we believe it to be worthy of exploration in the future.

We fitted the Logistic Regression model on our training dataset. The result on the testing database, BF2, was 93.07% accuracy, i.e. less than 7% of track were incorrectly classified as being outliers when in fact they contained face images (or being classified as face containing tracks when being outliers), which is a very high score given the annotation issues mentioned above.

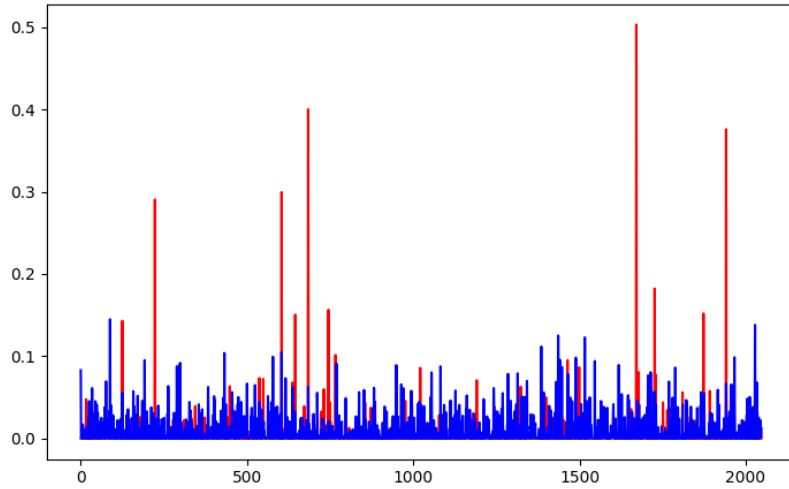


Figure 5.17: Distribution of features extracted using RESNET50. The red line corresponds to features extracted from outlier, i.e. image not containing any face. The blue line corresponds to features extracted from a face image. Detailed zoom is shown in Figure 5.18.

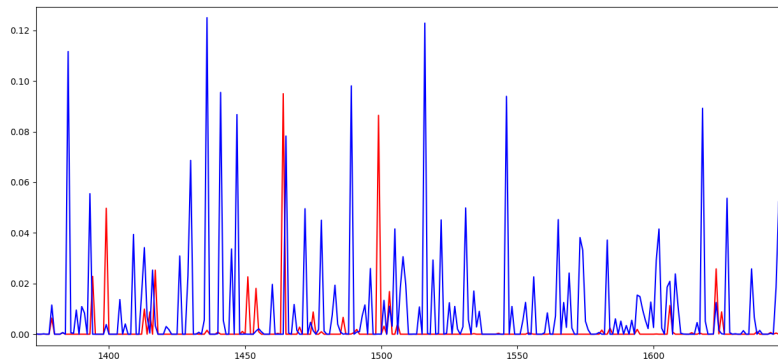



Figure 5.18: Detail of graph in Figure 4.6. The red line corresponds to features extracted from outlier, i.e. image not containing any face. The blue line corresponds to features extracted from a face image. The most interesting here is that most of the feature coordinates belonging to the outlier are 0, that is, the corresponding neurons in the last layer were not excited.



Chapter 6

Conclusions

In this thesis we have proposed a method for clustering identities in videos based on information extracted solely from facial images. In thorough experimental evaluation we have found that our method performs nearly as good as the current state-of-the-art video clustering algorithm of [40] while it is significantly simpler to implement.

The input to our method are faces extracted from a video by an ordinary face tracker. The output are clusters of face tracks corresponding to individual identities. Our method is composed of three main steps the optimal implementation of which was tuned experimentally. In the first step, we extract features from each detected face independently. The best face feature extractor turned out to be a combination of the VGG-Face [26] and RESNET50 [16] which are both CNNs pre-trained offline on millions of still images. In the second step, we aggregate the face features to a track descriptors. The coordinate-wise median was found to be the most efficient method for the feature aggregation. We also found that the performance can be significantly boosted by reducing dimension of the track descriptors to only 31 components via PCA. Finally, in the third step we group the track descriptors by the Spectral Clustering algorithm [38] that uses ($k = 6$)-nearest neighborhood kernel.

Our method is assembled from Open Source components and implemented in Python [29]. We use OpenCV [6] for basic manipulation with images and videos. The CNN face feature extractors, VGG-Face [26] and RESNET50 [16], were implemented in KERAS [9]. The tested clustering algorithms were imported from the Scipy [25] library.

Following the evaluation protocol used by the community, we used Buffy the Vampire Slayer [11] and Accio [15] dataset in our experiments. The hyper-parameters were tuned on the S05E05 episode of Buffy the Vampire Slayer while all the other episodes/movies were used solely for testing. Our method scored second to the state-of-the-art on Buffy the Vampire Slayer S05E02 dataset in terms of classification Accuracy, achieving 89.53% compared to the best 92.13% [40]. Our method took the first place by a large margin when compared to the state-of-the-art on Accio [15] database in terms of F_1 score, where our method got the score of 0.79 compared to the score 0.46 of the second best method [40]. Interestingly, the second method on Accio [15] was the first one on Buffy the Vampire Slayer S05E02 database.

We have also implemented a simple detector of outlier tracks, i.e. those composed of non-faces or containing multiple identities. The detector is based on the Logistic Regression and on the same track features which we used for clustering. A preliminary empirical evaluation show promising results as the detector is able to distinguish correct tracks from outliers with 92.07% accuracy.

6.1 Future work

In this section we outline three directions of a potential future work:

- The most straightforward improvement would involve using the spatial-temporal constraints as discussed in Section 5.5. A naive incorporation of the constraints would change the optimization task solved by the clustering algorithm into NP hard problem. Hence we recommend incorporation of the constraints into Spectral clustering as it was done e.g. in [41] who proposed so called Constrained Spectral Clustering.
- We also believe that there is a large room for improvements in the used method for feature aggregation. Currently, we use coordinate-wise median of L2-normalized face features to represent the track by a single feature vector. However, we hypothesize that a significant improvement might be achieved by replacing the single vector track descriptor with a more flexible representation of the faces in the track, e.g. using a set of descriptors.
- The proposed outlier detection method is preliminary. The main reason for putting relatively less energy to outlier detection is that the established evaluation protocol currently used for video clustering does not require us to deal with the outliers at all. However, development of a better track outlier detector as well as defining an evaluation protocol taking existence of outliers into account needs to be done in the future.



Bibliography

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical Report 2006-13, Stanford InfoLab, June 2006.
- [3] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proc. of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1, ACL '98/COLING '98*, pages 79–85, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.
- [4] M. Bauml, M. Tapaswi, and R. Stiefelhagen. Semi-supervised learning with constraints for person identification in multimedia data. In *Proc. of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13*, pages 3602–3609, Washington, DC, USA, 2013. IEEE Computer Society.
- [5] T. Berg and P. N. Belhumeur. Tom-vs-pete classifiers and identity-preserving alignment for face verification. In *British Machine Vision Conference*, 2012.
- [6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [7] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face and Gesture Recognition*, 2018.

- [8] X. Cao, C. Zhang, C. Zhou, H. Fu, and H. Foroosh. Constrained multi-view video face clustering. *IEEE Transactions on Image Processing*, 24(11):4381–4393, 2015.
- [9] F. Chollet et al. Keras. <https://keras.io>, 2015.
- [10] R. G. Cinbis, J. Verbeek, and C. Schmid. Unsupervised metric learning for face identification in tv video. In *Proc. of the 2011 International Conference on Computer Vision, ICCV '11*, pages 1559–1566, Washington, DC, USA, 2011. IEEE Computer Society.
- [11] M. Everingham, J. Sivic, and A. Zisserman. “Hello! My name is... Buffy” – automatic naming of characters in TV video. In *Proc. of the British Machine Vision Conference*, 2006.
- [12] A. W. Fitzgibbon and A. Zisserman. On affine invariant clustering and automatic cast listing in movies. In *Proc. of the 7th European Conference on Computer Vision-Part III, ECCV '02*, pages 304–320, Berlin, Heidelberg, 2002. Springer-Verlag.
- [13] A.W. Fitzgibbon and A. Zisserman. Joint manifold distance: a new approach to appearance based clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003*. IEEE Comput. Soc, 2003.
- [14] P. A. Flanagan. Face challenges, Dec 2018.
- [15] E. Ghaleb, M. Tapaswi, Z. Al-Halah, H. K. Ekenel, and R. Stiefelhagen. Accio: A data set for face track retrieval in movies across age. In *Proc. of the 5th ACM on International Conference on Multimedia Retrieval, ICMR '15*, pages 455–458, New York, NY, USA, 2015. ACM.
- [16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.
- [17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12*, pages 1097–1105, USA, 2012. Curran Associates Inc.
- [18] H. W. Kuhn and B. Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.
- [19] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Describable visual attributes for face verification and image search. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 33, pages 1962–1977, October 2011.
- [20] Z. Lu and T. K. Leen. Penalized probabilistic clustering. *Neural Computation*, 19(6):1528–1567, jun 2007.

- [21] M. Meilă. Comparing clusterings: An axiomatic view. In *Proc. of the 22Nd International Conference on Machine Learning, ICML '05*, pages 577–584, New York, NY, USA, 2005. ACM.
- [22] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Conference on Computer Vision and Pattern Recognition*, 1991.
- [23] D. Müllner. Modern hierarchical, agglomerative clustering algorithms. *CoRR*, abs/1109.2378, 2011.
- [24] D. L. Nguyen and M. T. Tran. VFSC: A very fast sparse clustering to cluster faces from videos. In *Computer Vision – ACCV 2016 Workshops*, pages 417–433. Springer International Publishing, 2017.
- [25] T. E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.
- [26] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.
- [27] S. Ramírez R. C. Malli, A. Suri. Vggface implementation with keras framework. <https://github.com/rcmalli/keras-vggface>, 2018.
- [28] C.J. VAN RIJSBERGEN. FOUNDATION OF EVALUATION. *Journal of Documentation*, 30(4):365–373, apr 1974.
- [29] G. Rossum. Python reference manual. Technical report, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, The Netherlands, The Netherlands, 1995.
- [30] Vivek Sharma, M Saquib Sarfraz, and Rainer Stiefelhagen. A simple and effective technique for face clustering in tv series. In *CVPR: Brave New Motion Representations Workshop*. IEEE, 2017.
- [31] 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 *Proc. of the 2014 Indian Conference on Computer Vision Graphics and Image Processing, ICVGIP '14*, pages 7:1–7:8, New York, NY, USA, 2014. ACM.
- [32] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
- [33] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. IEEE Comput. Soc, 2001.
- [34] Ulrike von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586, apr 2008.

- [35] Baoyuan Wu, Yifan Zhang, Bao-Gang Hu, and Qiang Ji. Constrained clustering and its application to face clustering in videos. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
- [36] S. Xiao, M. Tan, and D. Xu. Weighted block-sparse low rank representation for face clustering in videos. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 123–138, Cham, 2014. Springer International Publishing.
- [37] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition*, 2014.
- [38] S. X. Yu and J. Shi. Multiclass spectral clustering. In *Proc. of the Ninth IEEE International Conference on Computer Vision - Volume 2, ICCV '03*, pages 313–, Washington, DC, USA, 2003. IEEE Computer Society.
- [39] S. Zhang and J. Wang. Deep metric learning with improved triplet loss for face clustering in videos. In *Pacific Rim Conference on Multimedia*, 2016.
- [40] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Joint face representation adaptation and clustering in videos. In *Computer Vision – ECCV 2016*, pages 236–251. Springer International Publishing, 2016.
- [41] C. Zhou, C. Zhang, X. Li, G. Shi, and X. Cao. Video face clustering via constrained sparse representation. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, jul 2014.
- [42] C. Zhu, F. W., and J. Sun. A rank-order distance based clustering algorithm for face tagging. In *CVPR 2011*. IEEE, jun 2011.



Appendix A

Contents of the Attached Medium

- thesis.pdf
- source_thesis.zip: L^AT_EX source code of the text of this thesis
- source_code.zip: developed scripts in form of compressed git repository