Face Labelling and Place Recognition in Friends TV Show

Flavien Vidal
École Polytechnique
flavien.vidal@polytechnique.edu

October 10, 2022

Abstract

Labelling people in videos is essential for many video applications (organizing collections, automatic cast listing, browsing movies in Netflix, etc). Therefore, this report investigates generating statistics of scene presence of the famous characters in the Friends series.

1 Introduction

Labelling people in videos is essential for many video applications (organizing collections, automatic cast listing, browsing movies in Netflix, etc). In this report we focus our work on the first episode of season 3 of the Friends series. The goal is to provide statistics about the presence of the famous characters in the series along the different scenes inside the episode.

Section 2 discusses the method used for face detection, tracking and identification. Later Section 3 explains the scene detector used. Section 4 shows the results collected giving some insights to the reader and discusses the limitations of this method along faced problems before our conclusion that summarizes the report with future work extensions.

2 Tracking

For the tracking pipeline we divide the task into three main parts. The first part is face detection to locate the possible faces inside each frame. After that we follow our work with a face identification layer to assign a name to each face label detected. Finally, we perform the tracking using the SORT algorithm to keep following the characters along the scene. We explain in details the goal of our pipeline in the next following sections.

2.1 Face Detection

The overall pipeline of our approach [8] is shown in Fig. 1. Given an image, we initially resize it to different scales to build an image pyramid, which is the input of the following three-stage cascaded framework:

Stage 1: We exploit a fully convolutional network, called Proposal Network (P-Net), to obtain the candidate facial windows and their bounding box regression vectors. Then candidates are calibrated based on the estimated bounding box regression vectors. After that, we employ non-maximum suppression (NMS) to merge highly overlapped candidates.

Stage 2: All candidates are fed to another CNN, called Refine Network (R-Net), which further rejects a large number of false candidates, performs calibration with bounding box regression, and conducts NMS.

Stage 3: This stage is similar to the second stage, but in this stage we aim to identify face regions with more supervision. In particular, the network will output five facial landmarks' positions.

CNN Architectures: Multiple CNNs have been designed for face detection. However, we notice its performance might be limited by the following facts:

1. Some filters in convolution layers lack diversity that may limit their discriminative ability.

2. Compared to other multi-class objection detection and classification tasks, face detection is a challenging binary classification task, so it may need less numbers of filters per layer.

To this end, we reduce the number of filters and change the 5×5 filter to 3×3 filter to reduce the computing while increase the depth to get better performance. With these improvements we can get better performance with less runtime. Our CNN architectures are shown in Fig. 2. We apply PReLU as nonlinearity activation function after the convolution and fully connection layers (except output layers).

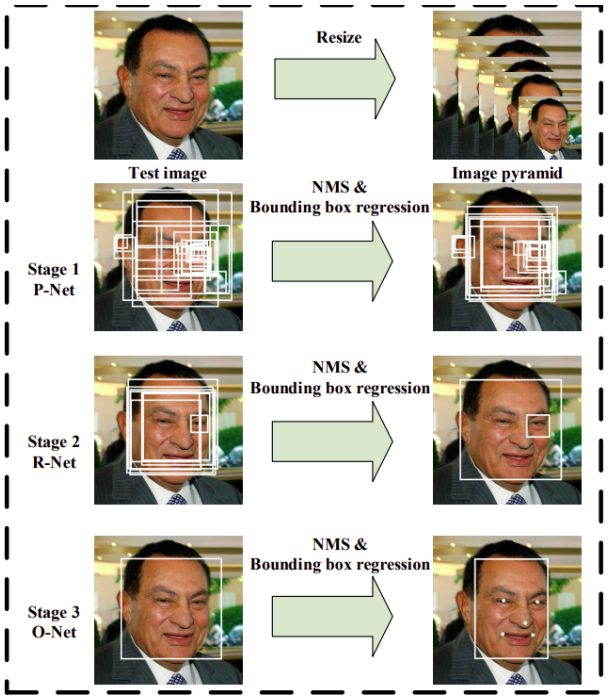

Figure 1: MTCNN pipeline

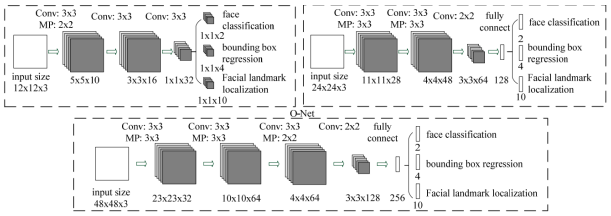

Figure 2: MTCNN Architecture

2.2 Face Identification

The approach used in this paper to do Face Identification is to find the encodings of the different faces of the main Friends character and save them for later comparison with detected faces along the different sampled frames. In order to get the encodings we rely on a model based on top of FaceNet [4].

Method: In contrast to previous approaches, FaceNet directly trains its output to be a compact 128-D embedding using a tripletbased loss function based on LMNN [6]. Our triplets consist of two matching face thumbnails and a non-matching face thumbnail and the loss aims to separate the positive pair from the negative by a distance margin. The thumbnails are tight crops of the face area, no 2D or 3D alignment, other than scale and translation is performed. Choosing which triplets to use turns out to be very important for achieving good performance and, inspired by curriculum learning [1], we present a novel online negative exemplar mining strategy which ensures consistently increasing difficulty of triplets as the network trains. To improve clustering accuracy, we also explore hard-positive mining techniques which encourage spherical clusters for the embeddings of a single person. As an illustration of the incredible variability that our method can handle see Fig.3. Shown are image pairs from PIE [5] that previously were considered to be very difficult for face verification systems.

Architecture: In all our experiments we train the CNN using Stochastic Gradient Descent (SGD) with standard backprop and AdaGrad. In most experiments we start with a learning rate of 0.05 which we lower to finalize the model. The models are initialized from random, and trained on a CPU cluster for 1,000 to 2,000 hours. The decrease in the loss (and increase in accuracy) slows down drastically after 500h of training, but additional training can still significantly improve performance. The margin α is set to 0.2. We used two types of architectures and explore their trade-offs in more detail in the experimental section. Their practical differences lie in the difference of parameters and FLOPS. The best model may be different depending on the application. E.g. a model running in a datacenter can have many param-

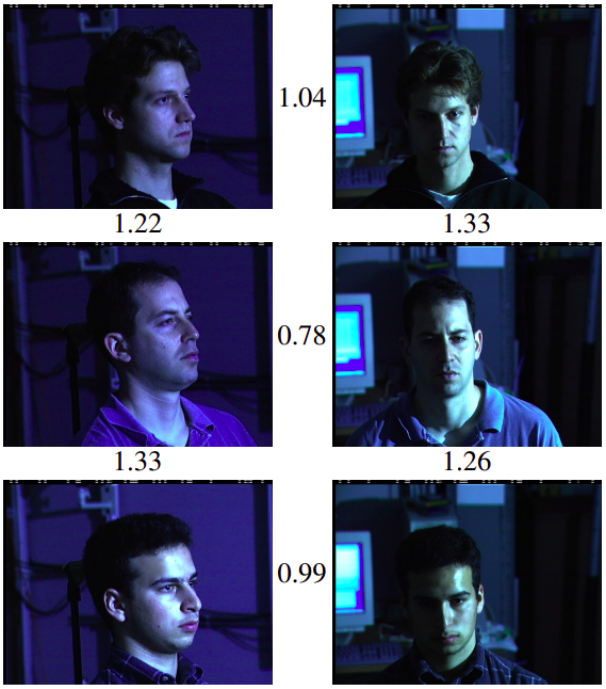

Figure 3: Variability

eters and require a large number of FLOPS, whereas a model running on a mobile phone needs to have few parameters, so that it can fit into memory. All our models use rectified linear units as the non-linear activation function. The first category, shown in Fig.4, adds 1x1xd convolutional layers between the standard convolutional layers of the Zeiler&Fergus [7] architecture and results in a model 22 layers deep. It has a total of 140 million parameters and requires around 1.6 billion FLOPS per image. The second category we use is based on GoogLeNet style Inception models. These models have 20× fewer parameters (around 6.6M-7.5M) and up to 5× fewer FLOPS (between 500M-1.6B). Some of these models are dramatically reduced in size (both depth and number of filters), so that they can be run on a mobile phone. One, NNS1, has 26M parameters and only requires 220M FLOPS per image. The other, NNS2, has 4.3M parameters and 20M FLOPS. Table 2 describes NN2 our largest network in detail. NN3 is identical in architecture but has a reduced input size of 160x160. NN4 has an input size of only 96x96, thereby drastically reducing the CPU requirements (285M FLOPS vs 1.6B for NN2). In addition to the reduced input size it does not use 5x5 convolutions in the higher layers as the receptive field is already too small by then. Generally we found that the 5x5 convolutions can be removed throughout with only a minor drop in accuracy. Fig.5 compares all our models

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Figure 4: NN1

2.3 SORT tracking

SORT[2] stands for "simple online realtime tracking" algorithm. It is a simple implementation of a framework for visual tracking of multiple 2D objects in video sequences.

Traditionally, this multiple object tracking (MOT) problem has long been considered as a data association problem where the objective was to associate detections between different frames of a video sequence. To facilitate this association process, trackers use var-

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

Figure 5: NN2

ious methods to model both the motion and the appearance of detected objects in the scene. Nevertheless, these traditional methods present a significant trade-off between accuracy and speed: most accurate trackers have a speed that is considered too slow for real-time tracking applications where only past and current image detections are available (as shown in figure 6).

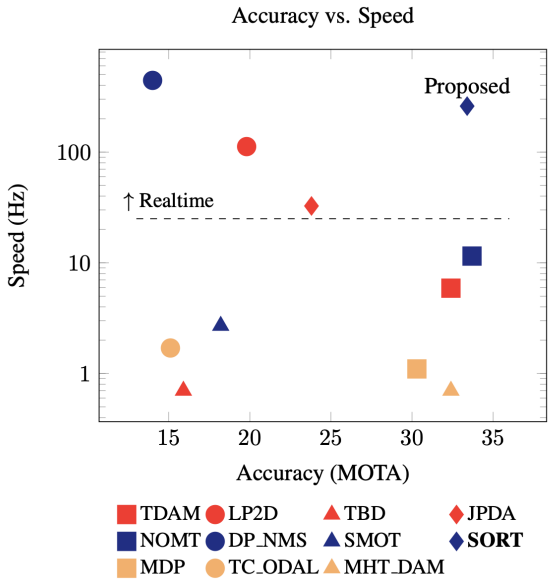

Figure 6: Benchmark performance of SORT in relation to several baseline trackers [2]

Unlike many batch tracking approaches, the main focus of SORT is precisely to associate the different objects in an efficient way for online and real-time applications and then to produce object identifications on the fly. Thus, the main focus is on efficiency to facilitate real-time tracking and promote the adoption of applications such as pedestrian tracking for autonomous vehicles. The work of Alex Bewley et al. mainly focuses on the latter application case, but due to the flexibility of CNN-based detectors, it can be generalized to other object classes. The case of face detection and tracking in a TV series is one such application.

For this reason, the quality of detection is identified as one of the key factors influencing tracking performance. The work of Alex Bewley et al. has shown that changing the detector could lead to an improvement in tracking of up to 18.9%.

In order to respect its objectives, especially regarding speed, SORT's architecture remains quite minimalist. For example, it does not manage occlusion or object return in a scene because their processing introduces an undesirable complexity that could limit its use in real-time applications. Only the position and size of the bounding box are used for motion estimation and data association, appearance features other than the detection component are all ignored. Its operation is based on rudimentary data association and state estimation techniques such as the Kalman filter and Hungarian algorithm.

Finally, SORT achieves results comparable to other state-of-the-art online trackers. On the other hand, due to its simplicity, it updates at a rate of 260 Hz, which is more than 20 times faster than other state-of-the-art trackers.

SORT algorithm takes advantage of advances in CNN-based detection, notably by using Faster Region CNN (FrRCNN) framework. Features are first extracted and regions are proposed, then the object is classified in the proposed region. The study by Alex Bewley et al. showed that the quality of the detection produced had a significant impact on the tracking performance.

To propagate a target's identity into the next frame, the inter-frame displacements of each object are approximated using a constant velocity linear model.

The locations in the current frame of the targets'

bounding boxes are predicted and then the assignment cost matrix is computed. It is defined as the intersection-over-union (IOU) distance between these predicted bounding boxes and each detection. The assignment is then solved using the Hungarian algorithm and a minimum IOU is used to reject assignments with too little overlap.

As objects enter and exit the video, unique identities must be created or discarded accordingly. The tracker is initialized using the bounding box geometry and tracks are terminated if not detected for TLost frames. This avoids unlimited growth in the number of trackers and localization errors caused by predictions over long durations without corrections from the detector.

3 Scene Detection

In this section we investigate the problem of scene detection inside the episode of the series in order to collect statistics based on each scene and character. We try one approach that relies on the pre-collected shots and show the problems faced in this approach. Secondly we move to an approach that does not require the shots as input and we compare results and show that the same problem persists.

3.1 Approach I

As discussed the idea of this approach is to rely on the pre-collected shots of our series. A shot is a range of frames representing a single camera roll or capture during the filming procedure. We then assume that a scene starts at a certain shot end (or in other words the start of the shot after it) and this limits our searching space to the number of shots we have if we are able to represent the scene information of a shot. We define our representation of scene information inside a shot by sampling k frames we call them scene frames that will be used to give the expected scene at this shot. We then use Places365-CNNs to provide a softmax probabilities over a set of predefined scene places.

3.1.1 Places

The Places dataset [9] is designed following principles of human visual cognition. Our goal is to build a core of visual knowledge that can be used to train artificial systems for high-level visual understanding tasks, such as scene context, object recognition, action and event prediction, and theory-of-mind inference. The semantic categories of Places are defined by their function: the labels represent the entry-level of an environment. To illustrate, the dataset has different categories of bedrooms, or streets, etc, as one does not act the same way, and does not make the same predictions of what can happen next, in a home bedroom, an hotel bedroom or a nursery.

In total, Places contains more than 10 million images comprising 400+ unique scene categories. The dataset features 5000 to 30,000 training images per class, consistent with real-world frequencies of occurrence.

3.1.2 Places356 CNNs

Using convolutional neural networks (CNN), Places dataset allows learning of deep scene features for various scene recognition tasks, with the goal to establish new state-of-the-art performances on scene-centric benchmarks. We present in next figure (Fig.7) the training of state of the art CNNs like VGG, Resnet, AlexNet etc.. [9] on the Places dataset with a top 1 and top 5 accuracy.

This approach suffered from problems related to the assumption made on the given shots ranges. We discuss in our results that the tests showed that these scene detectors are sensitive to the big rotation of the camera even in the same scene which tricks the algorithm to understand that this is the same scene due to high (even visual) changes in the background. Another observed problem is that the given shots (calculated in an automated way) also contain errors that prevent us from assuming that a shot lives strictly in a single scene. Which on other hands contradicts our assumption that a scene ends at an end (or start) of a shot. **Appendix A** illustrates this problem and highlights the discovered problem found in the shots data in a more visual way.

	Validation Set of Places365		Test Set of Places365	
	Top-1 acc.	Top-5 acc.	Top-1 acc.	Top-5 acc.
Places365-AlexNet	53.17%	82.89%	53.31%	82.75%
Places365-GoogLeNet	53.63%	83.88%	53.59%	84.01%
Places365-VGG	**55.24%**	84.91%	**55.19%**	85.01%
Places365-ResNet	54.74%	**85.08%**	54.65%	**85.07%**

Figure 7: Benchmark performance of places365 CNNs [9]

3.2 Approach II

In this second trial, we adopted a method that does not rely on the given shots information. Instead it takes the original mp4 video to extract the scene directly from the video.

3.2.1 PySceneDetect

PySceneDetect [3] is a command-line application and a Python library for detecting shot changes in videos, and automatically splitting the video into separate clips. Not only is it free and open-source software (FOSS), but there are several detection methods available, from simple threshold-based fade in/out detection, to advanced content aware fast-cut detection of each shot. PySceneDetect can be used on its own as a stand-alone executable, with other applications as part of a video processing pipeline, or integrated directly into other programs/scripts via the Python API. PySceneDetect is written in Python, and requires the OpenCV and Numpy software libraries. For our case we use the content-aware command for more accurate results. We show in **Appendix B** that this method also suffers from the same problem of large camera rotation inside the same scene.

4 Results and Discussion

4.1 Basic Statistics

We showed in the previous section and in both Appendix A, B that the scene detector is not reliable (along some other problems related to face positions and rotation which can prevent the face identification or tracking). Nonetheless we provide few statistic over the complete first episode of season 3 of friend. The machine used is a single core of an Intel i7 2.3GHz machine with 16GB of memory. We set the sampling rate k to two frames per shot sampling and we run our algorithm for around 6 min.

The number of scenes detected in the first episode is 77 and they are represented as follow: *stage indoor, pub indoor, downtown, fountain, discotheque, elevator door, dressing room, office cubicles, gift shop, classroom, basement, music studio, hot spring, bar, beauty salon, park, ice cream parlor, legislative chamber, nursing home, art school, biology laboratory, archive, pond, coffee shop, museum indoor, jewelry shop, art studio, butchers shop, ice skating rink outdoor, catacomb, fire escape, beer garden, restaurant, flea market indoor, restaurant kitchen, living room, kitchen, chemistry lab, utility room, office, hospital room, pizzeria, slum, general store indoor, conference center, pharmacy, movie theater indoor, airplane cabin, auditorium, dining hall, bookstore, food court, nursery, recreation room, physics laboratory, waiting room, veterinarians office, art gallery, jail cell, bank vault, pantry, science museum, television studio, reception, bedroom, hotel room, dorm room, operating room, storage room, home theater, lobby, shoe shop, berth, beer hall, boxing ring, sky, water tower*

Also we present in Fig.8 the scene presence per character For example it appears in this bar plot that Ross is the one that appeared the most in the scenes. Also in Fig.9 we present the opposite, which is most frequent scene inside the episode (we only show top 5 of the 77 scenes found). This graph shows that the stage indoor is the place where all character were present. This kind of studies can provide insight on the importance of scenes by analysing the presence of characters inside them.

4.2 SORT tracking

Our study mainly focuses on static camera sequences. Among the different video recordings of " Friends " TV show, we choose to study frames 2400 to 2650

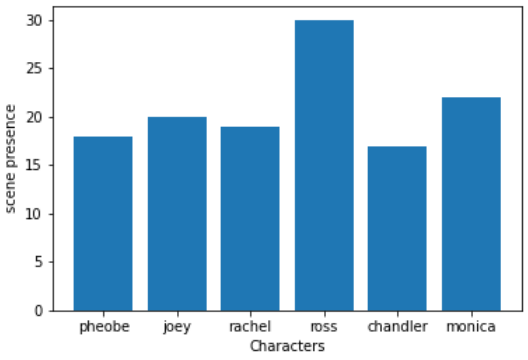

Figure 8: Scene count per character ep1-s3

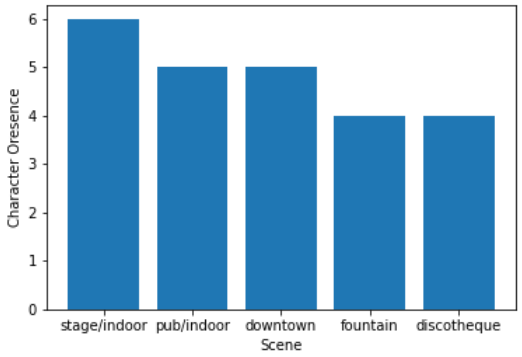

Figure 9: Character presence per scene ep1-s3

of the first episode. They allow us to identify different challenges related to the multiple object tracking task. For example, they include occlusion phases, character entering and leaving phases, the presence of face-like objects, ... The objects are detected in each frame and represented by bounding boxes.

In our experiments, we found that the quality of the detection had indeed a significant impact on the tracking. When some elements are recognized and detected as faces, SORT keeps track of these elements over several consecutive frames, even though the element is for example only a clock or an abstract shape. This problem is mainly due to the way SORT algorithm works and the principles on which it is implemented.

Since the SORT method focuses on frame-to-frame associations to grow tracklets over time, it should have a low number of lost targets compared to other traditional methods. Indeed, in our experiments we observe that targets are lost only when the detections disappear, for example when a character suddenly turns his back and only his hair is visible.

Furthermore, we observed that the IOU distance of the bounding boxes implicitly handles occlusion but only in the very short term. When a target is covered by an occluding object, only the occluder is detected, since the IOU distance appropriately favors similarly scaled detections. This allows both the occluder target to be corrected with detection, while the covered target is unaffected because no assignment is made.

The majority of multiple object tracking solutions aim for higher accuracy, often at the detriment of runtime performance. While the methods that achieve the best accuracy tend to be the slowest and the fastest methods tend to be less accurate, the SORT method combines these two properties, speed and accuracy. Although in our case a slower execution could have been tolerated, running SORT allowed us to run at about 393.54FPS without displaying results (and 800.18FPS with displaying results) on the single core of an Intel i7 2.3GHz machine with 8GB of memory.

5 Conclusion

As discussed in our report, we have investigated a pipeline of tracking, identification of faces along the frames of the first episode of season 3 of the friends series. We also investigated using scene detector with and without prior knowledge about the shots in order to find the scene boundaries. We then analyzed and ran some statistics on the frequency and character presence along the scenes of the episode. Also We presented the method used for face detection and identification We have. We then detailed the operation of SORT in the context of tracking identified faces. We showed that the scene detector is not always reliable, and we discussed in particular the problems related to the position and rotation of faces that can prevent identification or tracking. Regarding face

Figure 10: a) on leaving and returning, Ross is assigned a new ID, b) objects detected as faces are tracked, c) occlusion of Pheobe's face when looking at Ross results in the assignment of a new ID, d) Rachel's face does not disappear and is never occluded and is therefore tracked correctly from start to end.

tracking via the SORT algorithm, the results were shown to be very dependent on the quality of the detection algorithm. Although accurate and very fast, the algorithm does not always allow perfect tracking because of the way it is implemented. Finally, Appendix A, B showed the limitations and problems of our approach that is relying on the two approaches discussed for scene detection.

Among the tasks to be improved, we first distinguish the use of a better detection algorithm that would at the same time improve tracking significantly, the management of long-term occlusions of faces during tracking as well as the problems of assigning new identifiers when characters enter and exit the video. Also more face rotation and position can be captured and saved for the the ability to identify face in hard rotations.

Code: Our code can be found open sour on this GitHub LINK.

6 Appendix A

In this Appendix we illustrate the problem of the change of the distribution probability along the scene classes we have in the pre-trained Places360 CNN model. We choose as example the frames between 2059 and 2062 presented below (only 4 to make the example simple). We can see clearly that there is a big change in the background elements and camera rotation between the frames 2059-2060 and 2061-2062. On the other hand both of these frames should be considered as in the same scene because it is the same room. We illustrate the problem of Places360 CNN in Fig.11 that shows that between the frames 2060 and 2061 the top 2 predicted scenes change completely from *coffee shot, food court* to *nursing home, restaurant* although it is the same scene. Another problem noticed here is that the softmax values of probabilities is very low that makes the prediction almost uniform without clear meaning. This could be a problems related to the complex structure of elements present in the room of these frames. Interestingly, while looking at the detected scene boundaries using this approach, the algorithm we developed was able to detect true scene boundaries (between frames 3190-3200) as illustrated in Fig.13 (only 4 frames for simplicity). In Fig.12 we can see that the model detects a scene perturbation with many fluctuations due to the gradient transition between the two scenes. This can be seen in frame 3200 in Fig.14. On the other we notice that this scene end contradicts the assumed notion that a scene starts/ends at a start/end of shot because line 67 in the episode I shots shows that the frames **003155 - 03416** lie in the same shot although we have a scene change around frame 3200.

```
Frame: 2059
Scenes: ['coffee_shop', 'restaurant_patio']
Prob.: [0.10816515982151031, 0.09364070743322372]

Frame: 2060
Scenes: ['coffee_shop', 'food_court']
Prob.: [0.10258258134126663, 0.09118860214948654]

Frame: 2061
Scenes: ['nursing_home', 'restaurant']
Prob.: [0.07286517322063446, 0.06767337769269943]

Frame: 2062
Scenes: ['restaurant', 'nursing_home']
Prob.: [0.07185082882642746, 0.0700455829501152]
```

Figure 11: Stats - Frames 2059 - 2062

```
Frame: 3197
Scenes: ['beer_garden', 'market/outdoor']
Prob.: [0.10517697781324387, 0.06423811614513397]

Frame: 3198
Scenes: ['flea_market/indoor', 'jewelry_shop']
Prob.: [0.07503433525562286, 0.07052400708198547]

Frame: 3199
Scenes: ['jewelry_shop', 'gift_shop']
Prob.: [0.20711961388587952, 0.09818024933338165]

Frame: 3200
Scenes: ['jewelry_shop', 'gift_shop']
Prob.: [0.45633479952812195, 0.13539573550224304]

Frame: 3201
Scenes: ['jewelry_shop', 'gift_shop']
Prob.: [0.3951789438724518, 0.1693039834499359]

Frame: 3202
Scenes: ['art_studio', 'jewelry_shop']
Prob.: [0.13079434633255005, 0.11911459267139435]

Frame: 3203
Scenes: ['art_studio', 'museum/indoor']
Prob.: [0.20823179185390472, 0.19204042851924896]

Frame: 3204
Scenes: ['art_studio', 'art_gallery']
Prob.: [0.36324194073677063, 0.2032979279756546]
```

Figure 12: Stats - Frames 3197 - 3204

7 Appendix B

In this Appendix we illustrate the same problem about scene fluctuation in case of camera rotation by using the PySceneDetect package and we present on the same set of frames used in Appendix A the detected scene boundaries in Fig.15. The table shows that the algorithm detected 20 different scene changes were we initially only have 1.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML '09*, 2009. 2

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking, 2017. 3, 4

[3] BreakThrough. Pyscenedetect. https://github.com/Breakthrough/PySceneDetect, 2018. 6

[4] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[5] Terence Sim, Simon Baker, and Maan Bsat. The cmu pose, illumination, and expression (pie) database. *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 53–58, 2002. 2

[6] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In *In NIPS*. MIT Press, 2006. 2

[7] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013. 3

[8] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016. 1

[9] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 5, 6

0.5 

Figure 13: Frame 2059

0.5 

Figure 14: Frame 2060

0.5 

Figure 15: Frame 2061

0.5 

Figure 16: Frame 2062

0.5 

Figure 18: Frame 3190

0.5 

Figure 19: Frame 3196

0.5 

Figure 20: Frame 3200

10

0.5 

Figure 21: Frame 3203

[PySceneDetect] Scene List:

Scene #	Start Frame	Start Time	End Frame	End Time
1	0	00:00:00.000	25	00:00:01.000
2	25	00:00:01.000	61	00:00:02.440
3	61	00:00:02.440	104	00:00:04.160
4	104	00:00:04.160	202	00:00:08.080
5	202	00:00:08.080	229	00:00:09.160
6	229	00:00:09.160	282	00:00:11.280
7	282	00:00:11.280	314	00:00:12.560
8	314	00:00:12.560	668	00:00:26.720
9	668	00:00:26.720	934	00:00:37.360
10	934	00:00:37.360	988	00:00:39.520
11	988	00:00:39.520	1155	00:00:46.200
12	1155	00:00:46.200	1417	00:00:56.680
13	1417	00:00:56.680	1505	00:01:00.200
14	1505	00:01:00.200	1545	00:01:01.800
15	1545	00:01:01.800	1587	00:01:03.480
16	1587	00:01:03.480	1663	00:01:06.520
17	1663	00:01:06.520	1720	00:01:08.800
18	1720	00:01:08.800	1841	00:01:13.640
19	1841	00:01:13.640	1965	00:01:18.600
20	1965	00:01:18.600	2000	00:01:20.000

Figure 23: Stats - PySceneDetect