# Face Labelling and Place Recognition in the Friends TV Show

Flavien Vidal
École Polytechnique
flavien.vidal@polytechnique.edu

**Abstract**

Face labelling and place recognition in videos are essential components of many applications and as the number of online videos keeps growing, an efficient and automatic indexing solution becomes a must. In this work, we propose a simple method for automatically labelling faces and locations in videos. To this end, we build on recent approaches, in particular those based on deep learning such as Multi-Task Cascaded Convolutional Neural Network, Simple Online Realtime Tracking and FaceNet. We first study the key steps to labelling faces: we start by detecting faces in video clips, which then allows us to build a reliable and robust face tracking system, and use it to accurately label the faces in the scenes. We further investigate the influence of character motion on the ability to first detect and then track faces. Finally, we study place recognition and examine here the influence of camera motion and background changes.

## 1 Introduction

Human face processing techniques for video, including face detection, tracking, and recognition, have attracted a lot of interest because of the recent success of deep learning (particularly in computer vision) and because they are essential for video-understanding and have value in multiple applications. The most obvious ones are certainly in the security domain, such as in airports through surveillance cameras, or on highways through self-driving cars. However, it is also used in many other less crucial areas among which the organization of collections or the structuring, indexing, summarizing and search of videos. The human face can also provide a lot of information for identifying the appearance of certain people of interest, for example government leaders in news videos or heroes in movies. Place recognition has also attracted a lot of interest. It consists in deducing a location from an image and has wide applications in robotics, consumer photography, social media and archives.

In this work we propose to study the face labelling and place recognition tasks. For this purpose, we use a dataset consisting of all the episodes of the Friends television show. These frames have several advantages: in most cases, the characters do not move much, and when they do it is quite slow, the camera is mostly static and there is little variation in the angle of the shots. In addition, the scenes alternate in a very limited number of locations. All these properties ease the task of multiple object tracking and place recognition. In this specific report, we decided to focus our attention on the first episode of season 3 of the famous show.

In section 2, we start by studying the face labelling task which we divide into three main steps. The first step concerns face detection: in order to locate the different possible faces in each frame, we decide to use the Multi-Task Cascaded Convolutional Neural Network (MTCNN) [9] which is one of the most popular and fast architecture for face localization. The second step is dedicated to face tracking: we use the Simple Online Realtime Tracking (SORT) [2] algorithm which allows us to track previously detected faces while significantly improving computation time and maintaining good performance. Finally, the last step concerns the face identification layer which allows to assign a name to each detected face. In order to obtain the different face embeddings, we rely on a mathod based on FaceNet [5]. Section 3 focuses on the two approaches we used for the scene detection and place recognition task. Finally, section 4 provides the reader with additional insights into the results obtained and discusses the problems encountered as well as the limitations of our methods. We conclude this report by highlighting the key points of our work and by proposing extensions for future work.

## 2 Tracking Pipeline

We decided to divide the tracking pipeline into three main parts. The first part concerns the detection of faces in each frame of the videos using an MTCNN [9]. Then, we chose to track the movement of these detected faces throughout the whole scene using the SORT [2] algorithm. Finally, we proceed with a face identification layer based on FaceNet [5] in order to assign a name to each of the previously detected face. These steps are detailed in the following sections.

### 2.1 Face Detection

In this section, we describe our approach to joint face detection and alignment. In order to train or use a face recognition and tracking system we first need to locate the faces in the different input frames and align them so that we can use only the region of interest and reduce the background noise. One of the popular and fast architectures for face localization is the Multi-task Cascaded Convolutional Neural Network (MTCNN), which, as mentioned previously, is the one we used in our project and whose process is illustrated in *Figures* 1 and 2. It consists of three CNNs which identify faces with high precision: the Proposal Network, the Refinement Network and the Output Network. Given an input frame, we start by resizing it to build an image pyramid, i.e. a collection containing the exact same frame at different scales. This pyramid is then used as the input of the cascade structure.

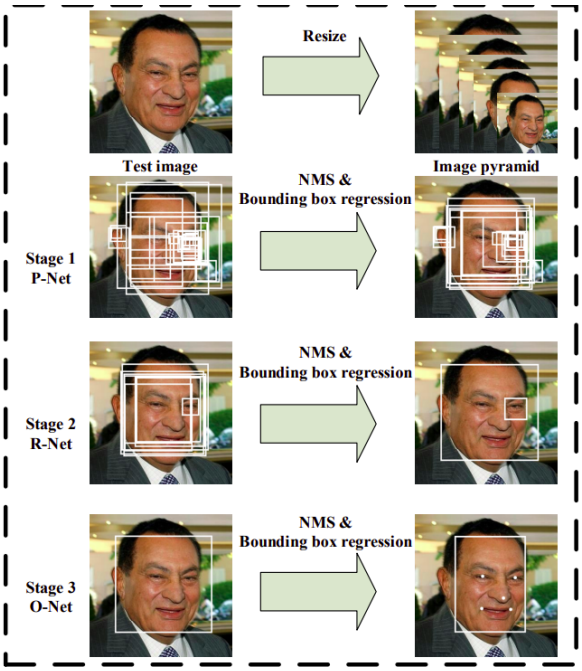


Figure 1: MTCNN pipeline [9]

**Stage 1**: For each frame in the pyramid, we use a $12 \times 12$ kernel that goes through every part of the frame, from the top left corner to the bottom right. All these portions of frames are given as input to a fully convolutional network, called Proposal Network (P-Net), which is scanning for faces and which eventually returns the coordinates of the bounding boxes and their confidence level if any faces are found. Since this work is performed in a short period of time and in order to produce relatively accurate bounding boxes, we chose to use a pre-trained Proposal Network with known weights and biases. Once the bounding box regression vectors are obtained, those with lower confidence are then removed. Since, most often, there are still a lot of bounding boxes left, and a lot of them overlap, we perform Non-Maximum Suppression (NMS) to merge highly overlapped candidates.

**Stage 2**: All candidates are then fed to another network, called Refinement Network (R-Net), which also rejects a large number of false candidates. It takes as input all the faces that survived stage 1 and in a similar way, outputs the new, more accurate bounding boxes, and their confidence level. Then, it performs calibration with bounding box regression, and conducts Non-Maximum Suppression (NMS).

**Stage 3**: Finally, the resulting bounding boxes are transmitted to the Output network, which performs a similar job, but instead of producing only the bounding boxes and the corresponding confidence level, identifies the face regions with more supervision. In particular, it further provides the coordinates of the five facial landmarks.

Regarding the architecture of the used networks, several CNNs have already been developed and studied for face detection. However, the authors of [9] observed that their performance could be limited by the following facts:

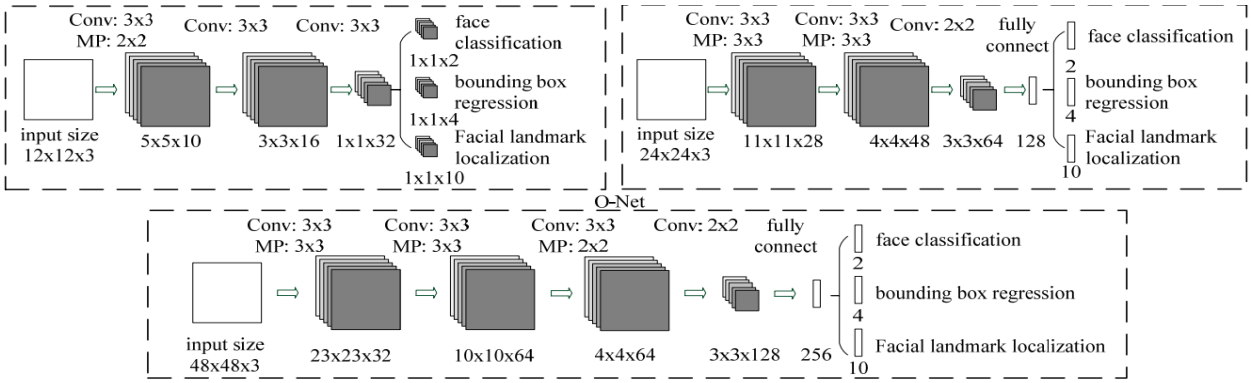


Figure 2: MTCNN Architecture [9]

1. Some filters in the convolution layers lack diversity, which can limit their ability to discriminate.
2. Compared to other multi-class object detection tasks, face detection is binary and may therefore require less but discriminative filters per layer.

To this end, the authors reduced the number of filters and changed the 5×5 filter to a 3×3 filter to reduce the computation while increasing the depth to achieve higher performance. With these improvements, they achieve higher performance with less computation time. Theses results motivated our choice of an MTCNN for face detection. The three CNN architectures used are illustrated in *Figure* 2.

## 2.2 Face tracking

After obtaining the resulting MTCNN bounding boxes for an entire video, we need to associate these detections between the different frames of the sequence. To do so, we chose to use the Simple Online Realtime Tracking (SORT) [2] algorithm. It is a simple implementation of a framework for visual tracking of multiple 2D objects in video sequences.

Traditionally, the multiple object tracking 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 various 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 as shown in *Figure* 3: 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. Moreover, their combinatorial complexity is exponential with respect to the number of tracked objects making them ineffective in many applications such as tracking vehicles on highways or tracking players on a soccer field.

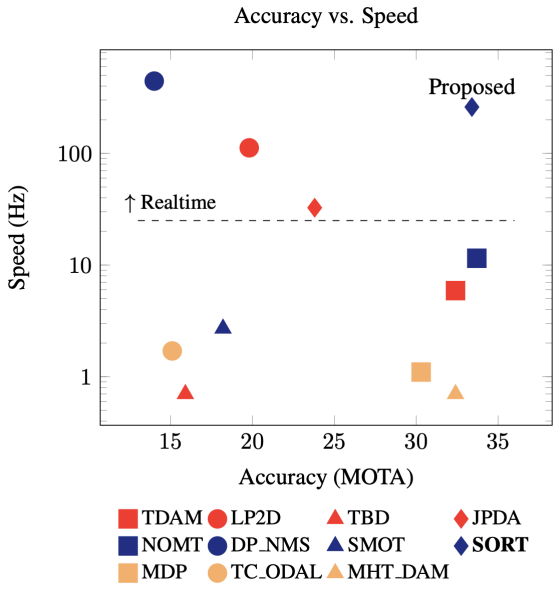


Figure 3: Benchmark performance of SORT in relation to several baseline trackers [2]

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 ap-

plications 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, but due to the flexibility of CNN-based detectors, it can be generalized to other object classes. Face detection and tracking in TV series is one such application. However, the quality of detection is identified as one of the key factors influencing tracking performance. Their work showed changing the detector could further 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. It takes advantage of advances in CNN-based detection, notably by using Faster R-CNN [4] framework. Features are first extracted and regions are proposed, then the object is classified in the proposed region. 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 a given number of frames. This avoids unlimited growth in the number of trackers and localization errors caused by predictions over long durations without corrections from the detector.

## 2.3 Face Identification

The approach we use to many other face identification consists in first determining the face embeddings of the main characters and saving them for later comparison with the newly detected faces in the different frames. In order to obtain these embeddings, we rely on a model based on FaceNet [5]. This is a deep neural network that allows us, from an input image of a person's face, to extract its characteristics in the form of a 128-dimensional byte vector.

Ideally, the embeddings of similar faces are also similar, which makes it then possible to easily recognize an unknown person on a new image. We therefore only need to compute the embedding of the face of this unknown person, and then determine its distance to the already known embeddings of the main characters. If the new embedding is close enough to one of the known embeddings, say Chandler's, then the newly detected face is probably Chandler's.

Regarding the method we chose to use, and contrary to previous learning approaches, FaceNet directly trains its output to be a compact 128-dimensional embedding using a triplet-based loss function based on Large Margin Nearest Neighbor (LMNN) [7]. The triplets we use consist of two matching face thumbnails (the anchor example and the positive example) and one non-matching face thumbnail (the negative example). These thumbnails are tight crops of the face area. The employed loss aims to separate the positive pair from the negative example by a distance margin. Choosing which triplets to use turns out to be very important for achieving good performance and, inspired by curriculum learning [1], FaceNet proposes a novel online negative exemplar mining strategy which ensures consistently increasing difficulty of triplets as the network trains. To improve clustering accuracy, the authors also explore hard-positive mining techniques which encourage spherical clusters for the embeddings of a single person. *Figure* 4 illustrates the incredible variability that this method can handle. It shows pairs of images from the Pose Illumination and Expression (PIE) database [6] that were previously considered very difficult for face verification systems.

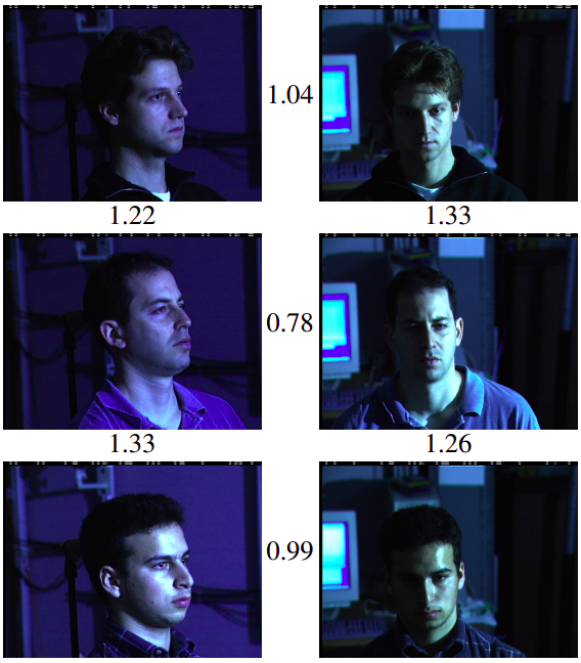


Figure 4: Variability

In all experiments conducted by the authors, the CNNs were trained using Stochastic Gradient Descent (SGD) with standard backpropation and AdaGrad. In most experiments they start with a learning rate of 0.05 which they 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 $\alpha$ is set to 0.2. They used two types of architectures and explore their trade-offs. Their practical differences lie in the difference of parameters and FLOPS. The best model may be different depending on the application. For example, a model running in a datacenter can have many parameters 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 the models use ReLU as the non-linear activation function.

The first category, shown in *Figure* 5, adds $1\times1\times d$ convolutional layers between the standard convolutional layers of the Zeiler&Fergus [8] 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 is based on GoogLeNet style Inception models. These models have twenty times fewer parameters (around 6.6M-7.5M) and up to five times 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. *Figure* 6 describes NN2 largest network in detail.

| 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 5: NN1 (FaceNet)

| 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 6: NN2 (FaceNet)

# 3 Scene Detection

In this section, we study the problem of scene detection and place recognition within the episodes of the show. The end goal is to develop a flexible approach to collect statistics based on each scene and characters. We first propose an approach based on pre-collected shots and then adopt a second approach that does not require these shots as input. We then compare the results and show that in both approaches, the same issue remains.

## 3.1 Places365-CNNs Approach

As mentioned above this first approach relies on the pre-collected shots of the Friends TV show. A shot is a range of frames representing a single camera roll or capture during the filming procedure. We assume a scene starts at a certain shot end which limits our searching space to a smaller number of frames pr shot we have access to. For power and time computation reasons, we chose to sample a subset of $k$ frames of a shot that will be used to predict the expected scene in this specific shot. For this, we rely on CNNs that have been trained on the Places365 dataset to provide probabilities on a set of predefined places.

### 3.1.1 The Places Dataset

The Places365 dataset is a subset of the Places Database [10]. It is designed following principles of human visual cognition with the aim 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. The dataset, illustrated in *Figure* 7, includes categories such as rooms, streets, gardens, stores, and many others. Since there are different subcategories in which the analyses can be quite different, the dataset embeds many categories of rooms, streets, gardens, ... Indeed, 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. Overall, Places contains more than 10 million images comprising more than 400 unique scene categories. The dataset features 5000 to 30,000 training images per class, consistent with real-world frequencies of occurrence. For all these reasons, we believe that this dataset can provide flexible and adaptable solutions for many situations, which motivates our choice to use it.

### 3.1.2 Places356-CNNs

The Places dataset presented above, enables convolutional neural networks to learn deep scene features for various scene recognition tasks, with the aim of establishing new state-of-the-art performance on scene-centric benchmarks. To illustrate this, *Figure* 8 presents the training of state-of-the-art CNNs like VGG, Resnet, AlexNet and GoogLeNet [10] on the Places dataset with the associated top-1 and top-5 accuracies.

| | 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 8: Benchmark performance of places365 CNNs [10]

After testing CNNs trained on Places360, we observe that this first approach suffers from various problems.

A first limitation of our approach concerns the predicted probability distribution. We note that the probabilities associated with the different classes are most often very low, which in some cases makes the prediction of one location rather than another very sensitive. This is due to the fact that the number of classes we are trying to discriminate is very high (400) which compresses the probability distribution and makes the distinction of two categories sometimes very small. A second reason comes from the fact that the target categories are often close to each other: restaurants, restaurant patios, food courts, coffee shops, et. The different places on which we evaluate our model are also very close to each other and contain common elements, e.g. Chandler and Joey's salon, Monica and Rachelle's salon, Ross's salon, the coffee lounge, etc. We considered using fewer categories or grouping similar categories into a larger one. This part is left for our next work. We also considered training only on places of the Friends show


Figure 7: Image samples from various categories of the Places Database [10]

but we would like to achieve the most flexible solution and the Places dataset ensures this. Nevertheless, further work on this axis should be conducted in our next work. In particular we would like to work with a sliding window on the video frames in order to adjust and unify our results.

To illustrate the second issue we chose to study a limited number of frames as an example: frames 2059 to 2062 presented in *Figure* 9. Between frame 2060 and frame 2061, we observe a rotation of the camera angle and an obvious change of the background elements. However, these two images belong to the same scene since it is the same room. As such, they should be classified as one and the same place. However, as shown by the *Figure* 11, our model fails to group these two images into the same category. The scenes predicted by the model change from *coffee shop* to *nursing home* which are not so much related.

One important point, however, is that we observe a convergence in the predictions of our model. The top two predictions associated with each frame, from 2059 to 2062, all contain the same primary category, which can be described as "a place to eat": *coffee shop*, *restaurant patio*, *food court*, and *restaurant*.

Another positive feature of our approach is that it was able to detect the true boundaries of the scene, for example between frames 3190 to 3200, as illustrated in *Figures* 10 and 12 (we show only 4 frames for simplicity). We observe that the model detects a perturbation of the scene with many fluctuations due to the gradient transition between the two scenes. This is particularly visible in frame 3200. Furthermore, we note that this scene ending contradicts the assumptions that a scene begins and ends respectively at the beginning and end of a shot.

Figure 9: Highlight of the change in shot and background in frames 2060 to 2061 (from top to bottom)

Figure 10: Highlight of the fade detection in frames 3190, 3196, 3200 and 3203 (from top to bottom)

```
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

### 3.2 PySceneDetect Approach

In this second approach, we adopt a method that does not rely on the information on the given shots but instead takes the original video as input to extract the scene directly from it. It involves automatically dividing an episode of the show into basic time segments by detecting transitions between shots in a video. In particular we rely on PySceneDetect [3].

### 3.2.1 PySceneDetect

PySceneDetect is a command line application and Python library for detecting shot changes in videos, and automatically splitting the video into separate clips. This free and open-source software has several detection methods ranging from simple threshold-based fade-in/fade-out detection, to advanced content-based fast cut-off detection of each shot which, given our previous observations, is an important consideration in our study. To obtain more accurate results we choose to use the content-aware command.

However, this method also has some limitations. In particular, it also suffers from the difficulty of dealing with large camera rotations within the same scene.

*Figure* 13 illustrates this problem of fluctuating predictions when the camera is rotated. The images considered to obtain these results are identical to those studied previously. We observe that the algorithm detects twenty position changes while only two are genuine.

[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 13: Highlight of the different scenes detected by PySceneDetect

# 4 Results and Discussion

Regarding face labelling, our models have produced interesting results. First of all, let us specify that our study mainly focuses on static camera sequences. Among the different video recordings of the "Friends" TV show, we chose to further study frames 2400 to 2650 of the first episode. They allow us to identify different challenges related to the multiple object tracking task. For example, they include occlusion phases, characters entering and leaving phases, the presence of face-like objects, ... The objects are detected in each frame and represented by bounding boxes.

The first observation we can make concerns the difficulties related to the position and rotation of the faces. These movements are likely to interfere with the detection phase and increase the computations required to label them. More precisely, if a detected character turns his head to check the time on the clock for example, the MTCNN is no longer able to detect his face for a certain period of time before detecting it again. This implies an additional step since for each newly detected face, a new tracker is initialized to which it is necessary to associate a name via FaceNet.

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 as shown in *Figure* 14. 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-based model we used 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.

Regarding place recognition, the different scene detectors do not seem to be reliable enough to be satisfied. In particular, since they try to discriminate a large number of possible locations, they produce very compressed probability distributions making the prediction of one location rather than another sometimes very delicate. Moreover, they are often not able to handle the rotation of the camera angle and the change of background elements. However, they have also led to positive results. First, they resulted in a convergence of the top predictions with a mean average precision at 3 close to 0.9. For each input frame, the top 3 predictions generally contain the target category. Second, they are able to detect the true limits of scenes and this despite faded images.

In addition, we provide some statistics on the entire first episode of season 3 of the Friends TV show. We set the sampling rate $k$ to two frames per shot and ran our algorithm for almost 15 minutes. 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*

We also present in *Figure* **??** the presence of the characters in the scenes. For example, we observe that Ross is the character who appears the most often in this episode. Figure 9 presents the scenes most frequented by the characters (only the first 5 of the 77 scenes found are presented). This graph shows for example that the indoor scene is one of the place where all the characters are present. This type of study can give an overview of the importance of the scenes by analyzing the presence of the characters in them.

# 5 Conclusion

As discussed in this report, we started by studying and designing a face detection, tracking and identification pipeline. For this, we focused on the first episode of season three of the Friends TV show. We then studied the use of a scene detector with and without prior knowledge of the shots in order to determine the boundaries of the scene as well as to identify the locations where they took place.

Regarding face recognition and tracking, we started by outlining the three steps of our approach. We chose to focus on camera sequences that contained multiple challenges such as occlusion phases, character entry and exit phases or presence of face-like objects. Our experiments demonstrated the challenges associated with the position and rotation of faces that are likely to interfere with the detection phase and then increase the computations required to label them. Although SORT proved to be particularly fast, we saw that the quality of detection also had a significant impact on face tracking.

Among the tasks to be improved, we first distinguish the use of a better detection algorithm which would at the same time improve the tracking significantly. Next we further identify the handling of long-term occlusions of faces during tracking as well as the problems of assigning new identifiers when characters enter and exit the video. In addition, a larger set of facial rotations and positions can be captured and saved to improve the face identification capability. Finally the camera sequences we studied were static, therefore we leave the study on dynamic camera sequences for our next work.

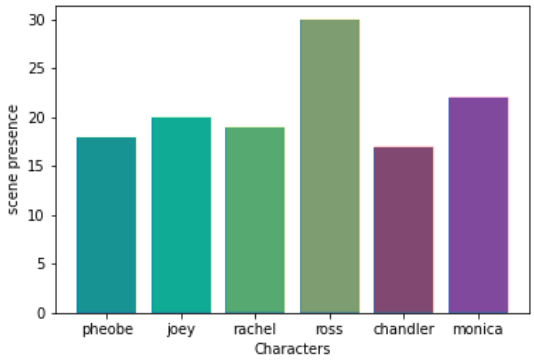


Figure 15: Scene count per character ep1-s3

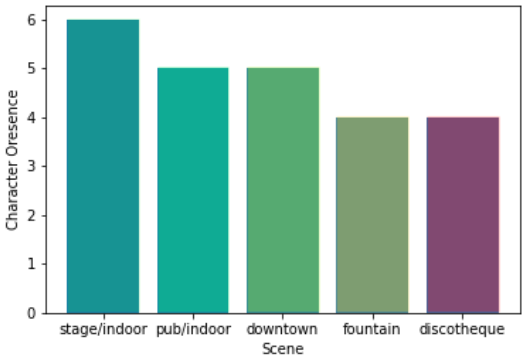


Figure 16: Character presence per scene ep1-s3

Regarding place recognition and scene detection, we used a pre-trained CNN and unfortunately found that the performance was quite poor, especially due to the variations of camera views and the changes of background within a same scene. The total number of classes we were aiming to discriminate had an impact on the overall performance, and this number being very large, the distinction between two classes was sometimes too fine. We considered using fewer classes or grouping similar classes into one larger one. We


Figure 14: 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.

also considered training our model only on the subset of the Friends show locations, but wanted to obtain the most generalizable and flexible solution possible, and therefore left this direction aside. Nevertheless, further work on these axes should be conducted in a future work.

Regarding the two tasks, face labelling and place recognition, the work has focused on the available frames but does not exploit additional information such as video sounds. However, this represents an essential element to help the identification of a character (through the voice) or a place (through the noise). It is therefore an important parameter that we will study in our next report.

# References


[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML '09*, 2009. 4

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking, 2017. 1, 2, 3

[3] BreakThrough. Pyscenedetect. `https://github.com/Breakthrough/PySceneDetect`, 2018. 8

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 1:91–99, 2015. 4

[5] 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. 1, 2, 4

[6] 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. 4

[7] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In *In NIPS*. MIT Press, 2006. 4

[8] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013. 5

[9] 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, 2, 3

[10] 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. 6, 7