Face recognition in movies

Yevhen Kuzmovych
ČVUT - FIT
kuzmoyev@fit.cvut.cz

May 2, 2018

1 Introduction

This work explores one of the modern face recognition techniques that is used in the *face_recognition*[2] library for Python. The output of this work will be the description of used methods and the application with the command-line interface that can analyze users video and produce the copy of the video with highlighted and named faces, statistics in CSV format and/or visualization of actors' appearances throughout the movie.

2 Input data

2.1 Retrieving

Movies casts' photos are needed for analysis. Actors photos were retrieved mainly from the *IMDB-WIKI dataset*[7] that has 500k+ images of the 20k+ celebrities. This dataset also provides metadata about photos like year photo was taken, face score, etc.

Photos of actors that are not found in this dataset are retrieved from the *TMDB API*[6]. The TMDB API is also used for movie search and cast specification (as movie title is set by an application parameter).

Additional photos with names can be added to analysis as application parameters.

2.2 Selection

After the cast is specified, photos of the actors are sought in the IMDB dataset. First photos are filtered by the year they were taken, taken only those from range $(y-3, y+3)$, where y is a release year of the movie. Then for each actor 3 photos(at most) with the highest face score are taken. If no such photos are found, they are downloaded from the TMDB API (only one photo per actor is provided).

The cast is limited to top 20 billed actors.

3 Methods

In this section methods used in the *face_recognition* library and their application in this work will be described.

3.1 Preprocessing

Photo preprocessing consists of:

- Converting to gradient domain (fig. 1)

Converting image to gradient domain eliminates dependency on the lighting in the scene. The same image with different brightness would have very different color values but the same gradient values which makes the task a lot easier.

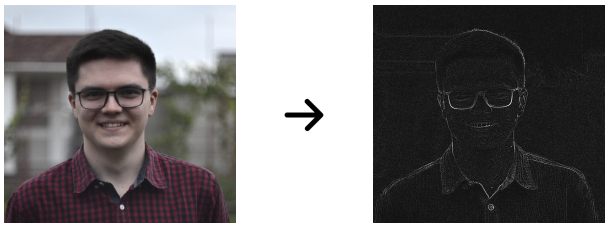

Figure 1: Gradient domain.

- Converting gradient to the histogram of oriented gradients (HOG)(fig. 2)

The gradient of the whole image gives too much detail and does not show actual facial features. That is why HOG representation of the image is used.

The algorithm for extracting HOGs counts occurrences of edge orientations in a local neighborhood of an image. In our case, the image is first divided into small connected regions, called cells, and for each cell a histogram of edge orientations is computed. The histogram channels are evenly spread over 0–180 or 0–360, depending on whether the gradient is 'unsigned' or 'signed'. The histogram counts are normalized to compensate for illumination.[1]

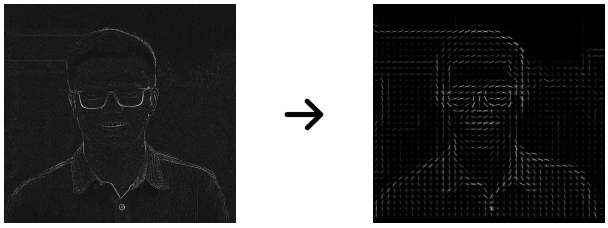

Figure 2: Histogram of oriented gradients(HOG).

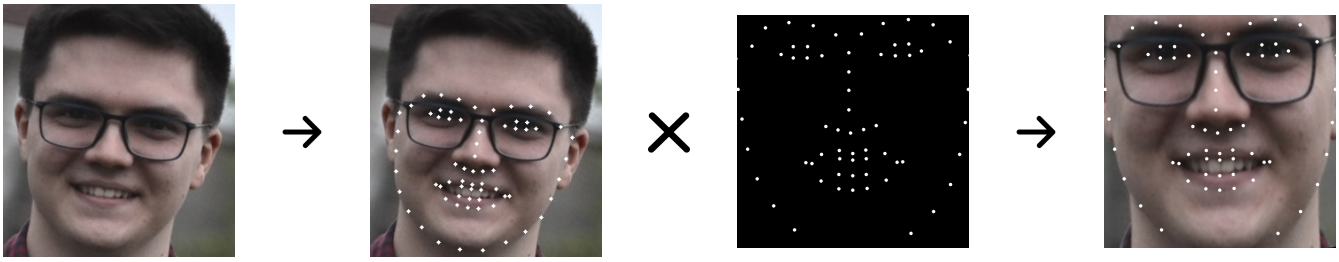

Figure 3: Aligned face landmarks.

Now to find a face on the image, we need to find the region that is similar to learned faces HOG (fig. 4). The example of learned HOG is taken from the *dlib*[5].

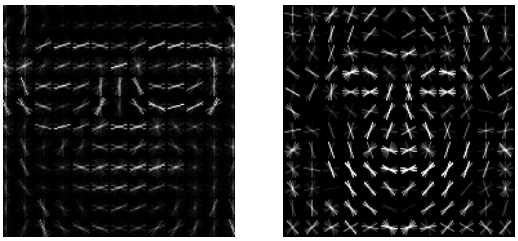

Figure 4: Comparison to learned HOG detector.

- Alignment of face landmarks (fig. 3)

The last problem that preprocessing solves is the face alignment. Algorithm for the face alignment used in the *face_recognition* was introduced by Vahid Kazemi and Josephine Sullivan in the paper *One Millisecond Face Alignment with an Ensemble of Regression Trees*[4]. The main idea of the algorithm is to find 68 specific points-landmarks on the face (fig. 3, 2nd image). Then using affine transformations align them with centered landmarks (fig. 3, 3rd image).

3.2 Encoding

The problem of image processing is that it is genuinely slow, so comparing images directly would not be efficient enough. That is why it is necessary to encode images into lower dimensionality. Those encodings would be some facial features that distinguish the face of the one person from the face of another.

Deep neural networks are used for specifying of those features. Trained DNN will receive the image of the face as an input and return 128 feature values as the output.

3.3 DNN training

DNN receives 3 images: 2 images of the same person and 3rd of the other and adjusts itself in such a way that difference of the first 2 images' encodings is minimal and difference of 1st and 3rd image is maximal. This DNN will encode faces into 128 numbers that describe most distinguishable features of the faces.

3.4 Faces classification

After preprocessing and encoding faces can be simply classified with any classification algorithm. The *face_recognition*s approach is to calculate euclidean distance between all known faces and the given face and classify it as one(or many) of the known faces if their distance is less than some constant tolerance.

Another approach for face classification in movies, where the number of known faces (cast) is limited and for each actor, there are more available photos, would be using KNN algorithm to get rid of classification as "Unknown" and to improve accuracy. Unfortunately, this approach had not been tested as it came to mind of the author at the time of writing this report. So it will be tested later.

4 Outputs

The result of the video processing by implemented application are:

- Copy of the vidoe with highligted faces as shown on fig. 7
- Statistics of the actors' faces apperiences and their locations throughout the movie
- Visualisation of the actors' apperiences throughout the movie as shown on fig. 5 and fig. 6

Five whole movies were analysed in the frameworks of this porject: *Pulp Fiction*, *Kill Bill: Vol. 1*, *Kill Bill: Vol. 2*, *Inglourious Basterds* and *Reservoir Dogs*. Statistics files can be found in *src/visualization/data/* folder.

5 Possible improvements

One of the big issues of the trained model is that it is has bad accuracy on non european faces. *Since the face recognition model was trained using public datasets built pictures scraped from websites. <...> Those public datasets are not evenly distributed amongst all individuals from all countries.*[3]. This problem is well visible on fig. 5 as Samuel L. Jacksons line is much rarer and containes more misclassifications than John Travoltas even in the common scenes. Another example of the same issue is fig. 8.

The solution would be to train model on images with bigger variety of races.

Another problem is in the misclassifications ("Unknown"s and wrong classification). As was mentioned in section 3.4 one of the solutions could be to use KNN for classification. This problem could also be solved by the noise removing in the output data. As most misclassifications don't last more than couple of frames, they can be detected and corrected.

6 Conclusion

The aim of this work was to understand modern face recognition techniques and to implement command-line interface to analyze videos. The goal is reached at a sufficient level and the work can be extended in functionality and classification quality.

References

[1] O. Déniz, G. Bueno, J. Salido, and F. De la Torre. Face recognition using histograms of oriented gradients. *Pattern Recognition Letters*, 32(12):1598 – 1603, 2011.

[2] Adam Geitgey. Face recognition. `https://github.com/ageitgey/face_recognition`, 2018.

[3] Adam Geitgey. Face recognition accuracy problems. `https://github.com/ageitgey/face_recognition/wiki/Face-Recognition-Accuracy-Problems`, 2018.

[4] Vahid Kazemi and Sullivan Josephine. One millisecond face alignment with an ensemble of regression trees. In *27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, United States, 23 June 2014 through 28 June 2014*, pages 1867–1874. IEEE Computer Society, 2014.

[5] Reference. Make your own object detector! [online]. `http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html` (visited: 2018-05-01).

[6] Reference. The movie database api [online]. `https://developers.themoviedb.org` (visited: 2018-05-01).

[7] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision (IJCV)*, July 2016. `"https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/"` (visited: 2018-05-01).

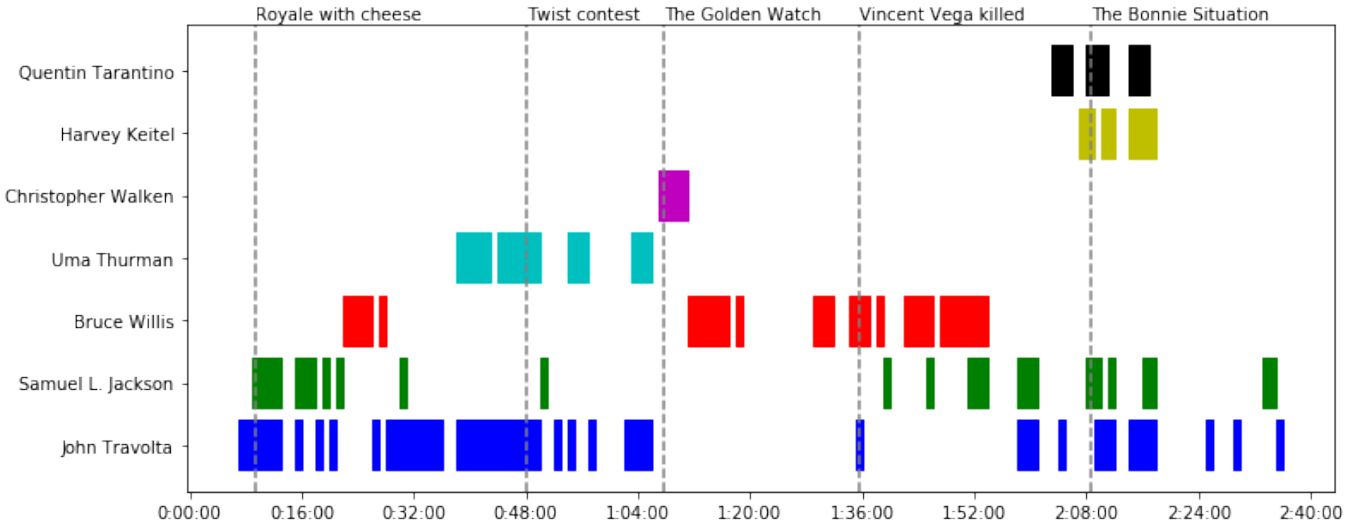

Figure 5: Pulp fiction analysis.

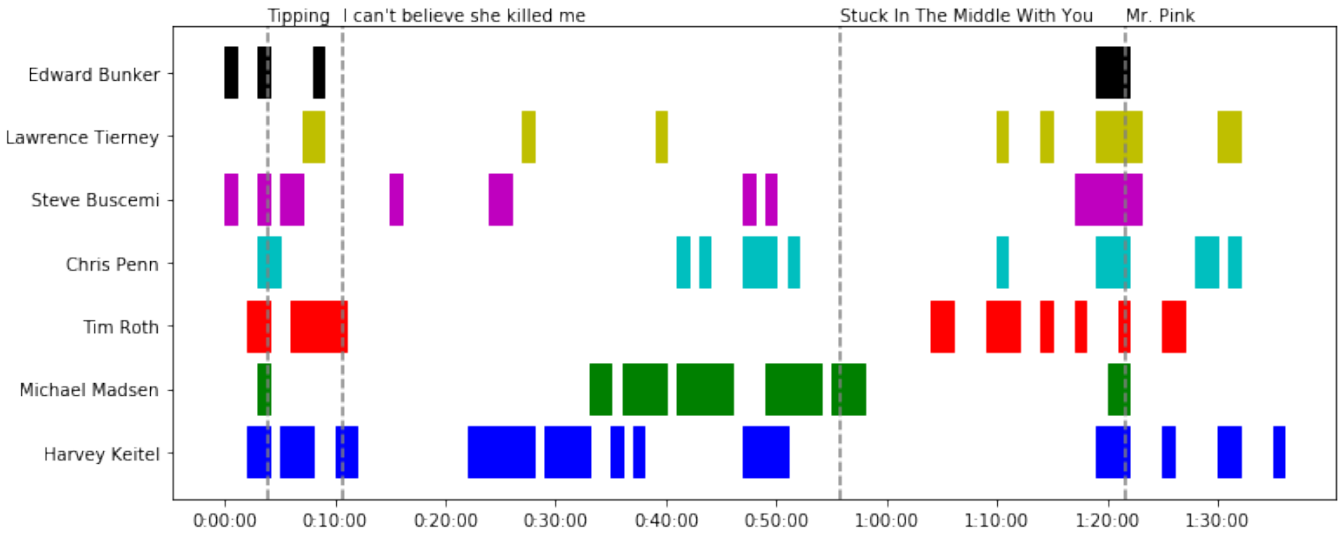

Figure 6: Reservoir dogs analysis.

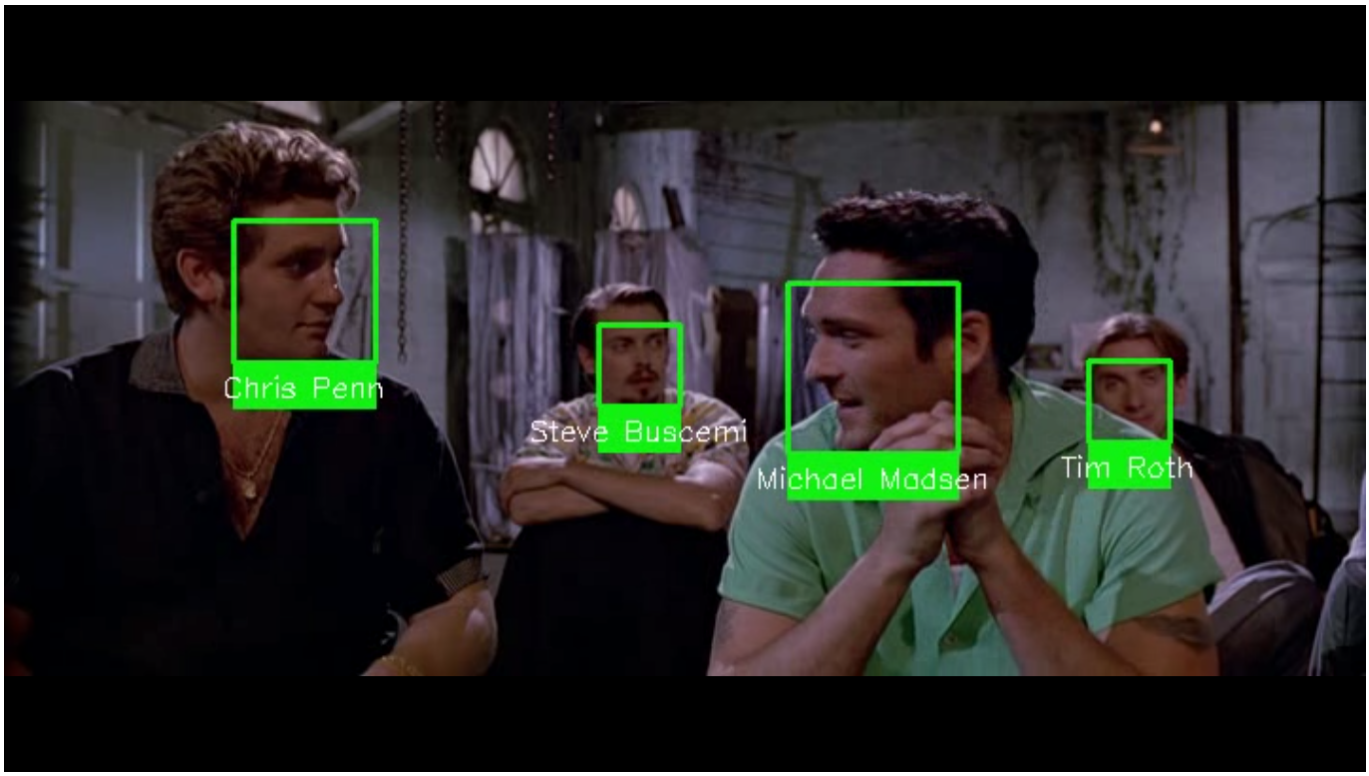

Figure 7: Reservoir dogs frame with highlighted and named faces.

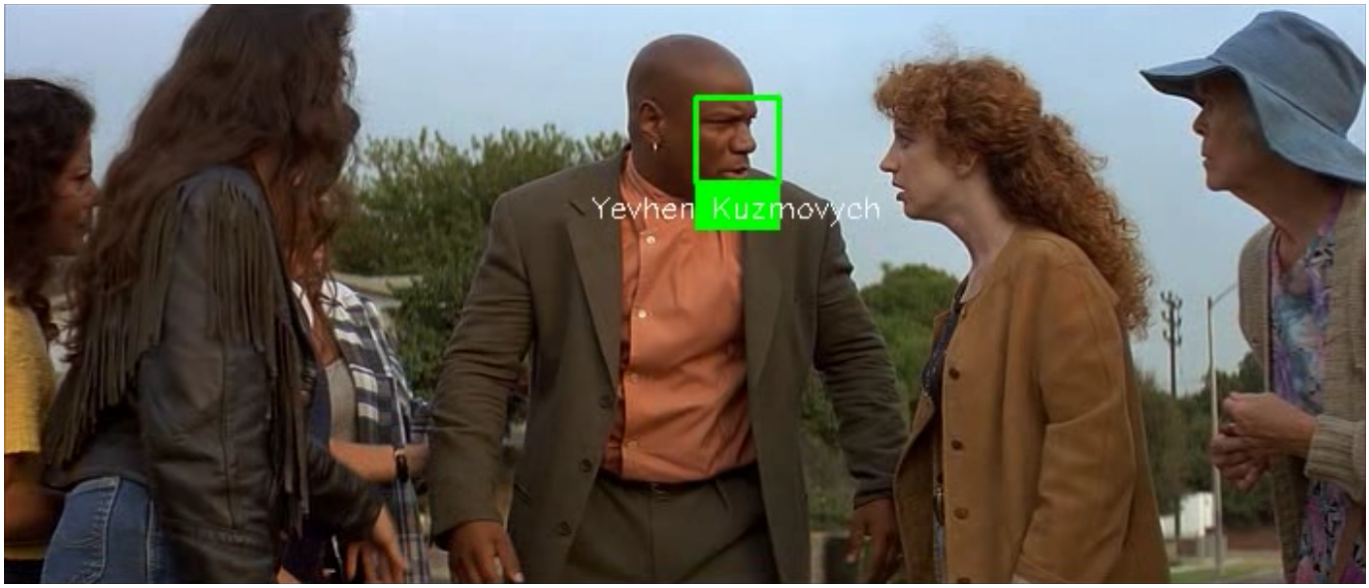

Figure 8: Misclassification of non european face (actual frame from the outputed video). Athors photo was included in the searched cast. It was (obviously falsely) classified in 5 frames in *Pulp Fiction*, in 3 frames in *Kill Bill: Vol. 1*, 1 in *Kill Bill: Vol. 2*, in 37! frames in *Inglourious Basterds* and none in *Reservoir Dogs*.