**3D Face recognition report**

CODE WILL BE GIVEN ON REQUEST

Nycer84@gmail.com

# Introduction:

This report shows the method that was used, give some instructions on running and explains the codes.

# Method

The method that was used is shown briefly in this diagram:



Figure : A holistic flow chart of the work scheme

Generally, in face recognition problems, the test input image is compared against a set of images from the database, that’s why the block diagram is divided into two main routes, the training route on the left is responsible of preparing the 3D reconstructions of the training data and store it in a database directory. The test image goes through the same process and gets compared in the final stage with images from the database.

The first stage of the processing, is face detection, in this step the input image either from the database or a test image is fed a deep learning model called MTCNN, which output the detected face bounding box and 5 landmark points, the five landmarks are as follows; the left eye, right eye, nose, left mouth point and right mouth point. The landmarks points are used for the 3D reconstruction stage.

The second stage in the processing is the 3D reconstruction, in this stage a model from Microsoft called 3D FaceRecon is utilized, it can input a single image or multiple images to perform the reconstruction. In most of the cases here, a single image was used for reconstruction. The original paper of the work is found here [[1903.08527] Accurate 3D Face Reconstruction with Weakly-Supervised Learning: From Single Image to Image Set (arxiv.org)](https://arxiv.org/abs/1903.08527), whereas the original codes are located in this other link [sicxu/Deep3DFaceRecon\_pytorch: Accurate 3D Face Reconstruction with Weakly-Supervised Learning: From Single Image to Image Set (CVPRW 2019). A PyTorch implementation. (github.com)](https://github.com/sicxu/Deep3DFaceRecon_pytorch).

The very abstract explanation of their work, is they use a predefined base face model BFM and they render the face features to match this template. It utilizes two networks, one called R-Net and the other is C-Net which depict the image features and poses respectively.

The 3D reconstruction is done firstly for the images in the database which were separated for training, then the test image is passed through the same process prior the face recognition stage. The images of the database after they were all rendered they were placed all in a single directory for the recognition process.

The final stage of the work is the face recognition, in this stage, the training dataset is prepared and the image to be tested is also prepared and rendered. The model that’s being used for face recognition is VGG-Face model imported from DeepFace library. The VGG-Face doesn’t do the recognition directly, instead, it extracts what so called “embeddings” out of each image, those embeddings count are 128 and represent unique features extracted from each face, the face recognition task then aims to reduce the distance between those embeddings of the test image and the image that it matches in the database, and increase the distance between the test image and other images in the database that doesn’t belong to this test image.

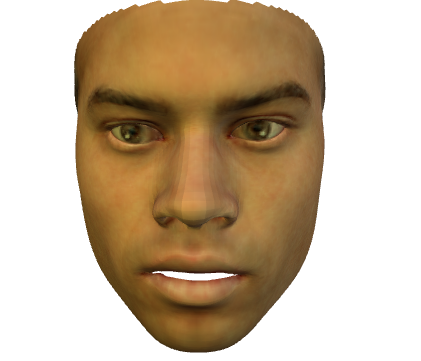
# Sample results:

Here this section shows some screenshots of the result of the reconstruction and the recognition stages.

## Rendering results

To view the rendered object you need to install Meshlab in your system, however, there are online viewers for the .obj files.

You can use this link for quick visualization: [Online 3D Viewer](https://3dviewer.net/) and drop the (.obj) file and you will be able to view the reconstruction.



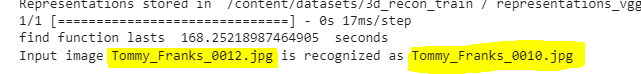


## Recognition result:

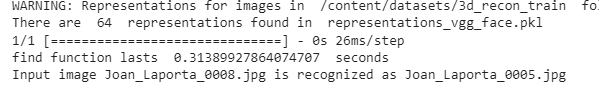
Code result for a subjects from the database. The results are represented with this psudo-code:

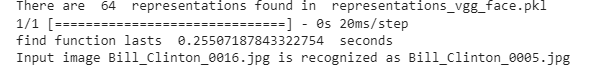
*Print(Input image “Test\_input\_image” is recognized as “Closest\_database\_match”)*

Note that the image that was used for test, is for the same subject but NOT INCLUDED in the training subjects as shown in the database organization section.

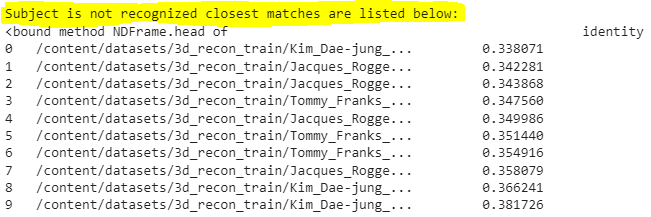








Subjects NOT INCLUDED in the database:



It directly prints “this subject is not recognized”, however, it gets list of the closest matches, but none of them is the correct subject, I have designed the code this way as precaution, if the face is in the database but the distance was further than the threshold. The nearest subject in this way must be the subject to be recognized.

# Instructions to run the codes:

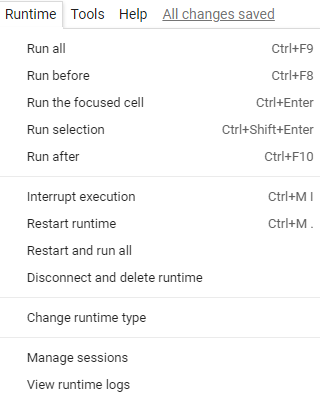
## General instructions:

I have done all of the development on Google Colab, because it offers a free GPU able to test all the functions and preform the predictions.

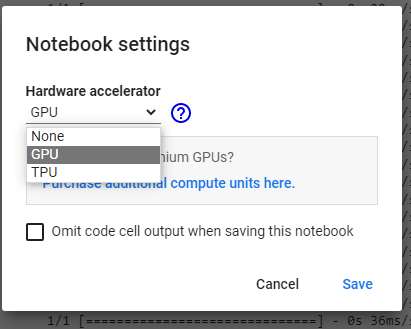
To run the code locally, a Linux system is needed, because some of the libraries recommend Linux. I have also used in my codes some Linux commands to request some downloads like database, model weights, etc. If you don’t have Linux installed just visit this Google Colab from this link: [https://colab.research.google.com/] and upload the files there, then run.

The codes are well explained in the notebook, so just read the text embedded within the code and comments, it will set you in a good position to understand all the codes, and if any help needed, just leave me a message.

Make sure that the GPU is enabled on Colab settings from this menu:



Go to “Change runtime type” and the following menu should pop us:



To run all the cells you can use then the shortcut key CTRL + F9, to run a single cell use CTRL + ENTER.

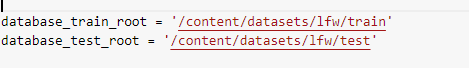
## Files organization:

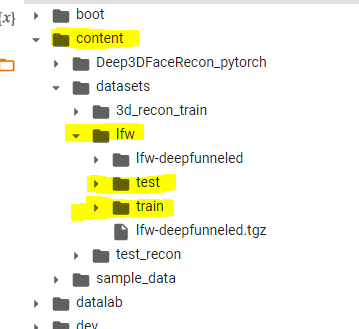
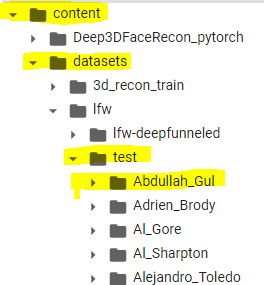
This section will show how to arrange the files properly to be able to run the code, and the locations where you can find the results for visualization purposes:

### Database folders:

#### Raw input images dataset

In the code there are two variables to determine the locations for the training root directory and test root directory as follows, this file contains the RAW database image before rendering. Each of these files contains subject names, then, images for that subjects in their corresponding folders.



Please arrange your database in this format for proper operation of the codes:

|  |  |  |
| --- | --- | --- |
| Training | | |
| Database\_train\_root/ | Subject 1/ | Img1.jpg |
|  |  | Img2.jpg |
|  |  | Img3.jpg |
|  | Subject 2/ | Img1.jpg |
|  |  | Img2.jpg |
|  |  | Img3.jpg |
|  | Subject 3/ | Img1.jpg |
|  |  | Img2.jpg |
|  |  | Img3.jpg |
| … | … | … |

|  |  |  |
| --- | --- | --- |
| Testing | | |
| Database\_test\_root/ | Subject 1/ | Img1.jpg |
|  |  | Img2.jpg |
|  |  | Img3.jpg |
|  | Subject 2/ | Img1.jpg |
|  |  | Img2.jpg |
|  |  | Img3.jpg |
|  | Subject 3/ | Img1.jpg |
|  |  | Img2.jpg |
|  |  | Img3.jpg |
| … | … | … |

Images name can be any arbitrary name, preferred to contains the subject name as well but not necessary.

#### Detections directory

Another important directory contains the detection results, which is in raw\_img\_filename.txt organized as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Deep3DFaceRecon\_pytorch/ |  |  |  |  |
|  | <folder\_to\_test\_images>/ |  |  |  |
|  |  | (\*.jpg/\*.png)/ |  |  |
|  |  |  | Detections/ |  |
|  |  |  |  | \*.txt |

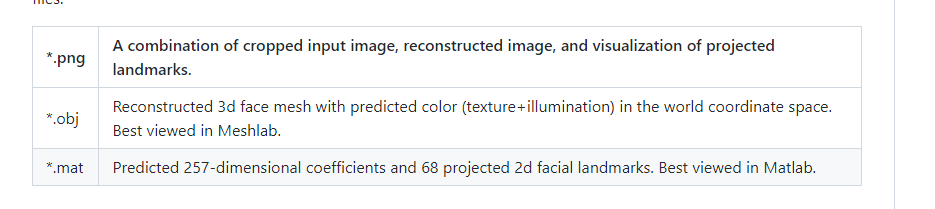
The detections text file contains the 5 landmarks automatically generated by the code and placed properly in the train / test image directory.

#### Rendered images folder

The rendered database images are located in: “/content/datasets/3d\_recon\_train” you don’t need to modify this file as it’s generated automatically.

### Rendered images results:

The rendered images are presented with different formats as the following table:



The directory in the files that contains the reconstruction result is shown in the following path:

/Content/Deep3DFaceRecon\_pytorch/checkpoints/face\_3d\_recon/result/Subject\_name\*/img\_filename\*.png

/Content/Deep3DFaceRecon\_pytorch/checkpoints/face\_3d\_recon/result/Subject\_name\*/img\_filename\*.mat

/Content/Deep3DFaceRecon\_pytorch/checkpoints/face\_3d\_recon/result/Subject\_name\*/img\_filename\*.obj

The final results directory contains the result according to the subject’s name then the image\_filename with three different results as shown in the previous table

# Note regarding other variations:

There is the first model that I have shown you the results for, which was rendering the ears as well, but unfortunately it went through too many steps and final reconstruction wasn’t quite successful. The model is called FLAME and here is the link for the GitHub repo [Zielon/MICA: MICA - Towards Metrical Reconstruction of Human Faces [ECCV2022] (github.com)](https://github.com/Zielon/MICA).

There are other 3D rendering models that can output reconstructed 3D images from multiple 2D images of the SAME SCENE. They are not dedicated to face recognition, and you can consider them for reconstruction from multiple views:

1. Meshroom: [alicevision/Meshroom: 3D Reconstruction Software (github.com)](https://github.com/alicevision/Meshroom)
2. openMVG [openMVG/openMVG: open Multiple View Geometry library. Basis for 3D computer vision and Structure from Motion. (github.com)](https://github.com/openMVG/openMVG)