Hereafter - Mumbai

Swagat Shubham Bhuyan
Intern

Avatar Generation SDK Documentation

30/10/2020

Introduction

The Avatar Generation SDK was built internally, through various reconstruction and face detection repositories already available in Github, under fair use policies and guidelines. Many files, such as various morph files, the network folder present in the directory, and BFM files for texture generation are protected under copyright guidelines, and hence must be looked upon carefully before use. Otherwise, Citations have been provided by our personal github repository to replicate the final Avatar generation SDK repository. All the relevant links are provided in this documentation.

System Requirements

- **NumPy** : A fundamental package for scientific computing with Python.
- **OpenCV** : A library of Python bindings designed to solve computer vision problems.
- **dlib** : A toolkit for making real world machine learning and data analysis applications.
- **imutils** : A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.
- **Reconstruction** : Reconstructions can be done on both Windows and Linux. However, we suggest running on Linux because the

rendering process is only supported on Linux currently. If you wish to run on Windows, you have to comment out the rendering part.

- **scipy**
- **pillow**
- **Tensorflow 1.4 ~ 1.12**
- **Basel Face Model 2009 (BFM09)** : Expression Basis (transferred from Facewarehouse by Guo et al.). The original BFM09 model does not handle expression variations so extra expression basis are needed.
- **tf mesh renderer** : We use the library to render reconstruction images. Install the library via pip install mesh_renderer. Or you can follow the instruction of tf mesh render to install it using Bazel. Note that current rendering tool does not support tensorflow version higher than 1.13 and can only be used on Linux.
- **BLENDER:** Blender is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, motion graphics, interactive 3D applications, virtual reality, and computer games.

Server Installation

A server side installation would make sense as the dependencies are very operating system / device specific. A simple AWS ec2 instance connected with Lambda or a Google Cloud Server can be set up, and above dependencies can be integrated.

1. Clone the repository:

```
git clone https://github.com/SwagatSBhuyan/Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai.git
```

2. Download and install Python 3.8 or Anaconda 4 on your system as per your operating system. It is crucial to have only upto Python 3.8 for the TensorFlow codes to work, hence it is strongly advised to install Python along with Anaconda.
   The given setup.exe or setup.py files have been built with options to download and install the same
3. Install the required python packages needed to run the scripts. Use the given setup.py or setup.exe in the dedicated server to install the dependencies.
   To install special dependencies like dlib into an online server, it is advised to separately seek out documentations, as their installations also require other pre-built dependencies and builders like Visual Studio.
   Make sure to turn down option to extract data.rar manually.
4. Download the Basel Face Model. Due to the license agreement of Basel Face Model, you have to download the BFM09 model after submitting an application on its home page. After getting the access to BFM data, download "01_MorphableModel.mat" and put it into *./BFM* subfolder.
5. Download the Expression Basis provided by Guo et al. You can find a link named "CoarseData" in the first row of the Introduction part in their repository. Download and unzip the Coarse_Dataset.zip. Put "Exp_Pca.bin" into *./BFM* subfolder. The expression basis are constructed using Facewarehouse data and transferred to BFM topology.
6. Download the trained reconstruction network, unzip it and put "FaceReconModel.pb" into *./network* subfolder.
7. The directory should now have a *./main_pipeline.exe* file ready for execution.
   If not, simply run the *./main_pipeline.py* script file in your server, to open the dedicated GUI to browse a JPG image and generate its 3D Face Reconstructed obj file for further rendering in Unreal.

8. Check the *./output* folder to acquire the generated .obj files with built-in vector textures, that can be rendered using softwares, such as meshlab.
9. Check the ./final_json for the final coordinate difference Json file between 68 landmark facial features of original head mesh and generated 3D face reconstructed mesh.
10. Check ./textures to receive the final texture map generated to be applied along with generated .ply face mesh.

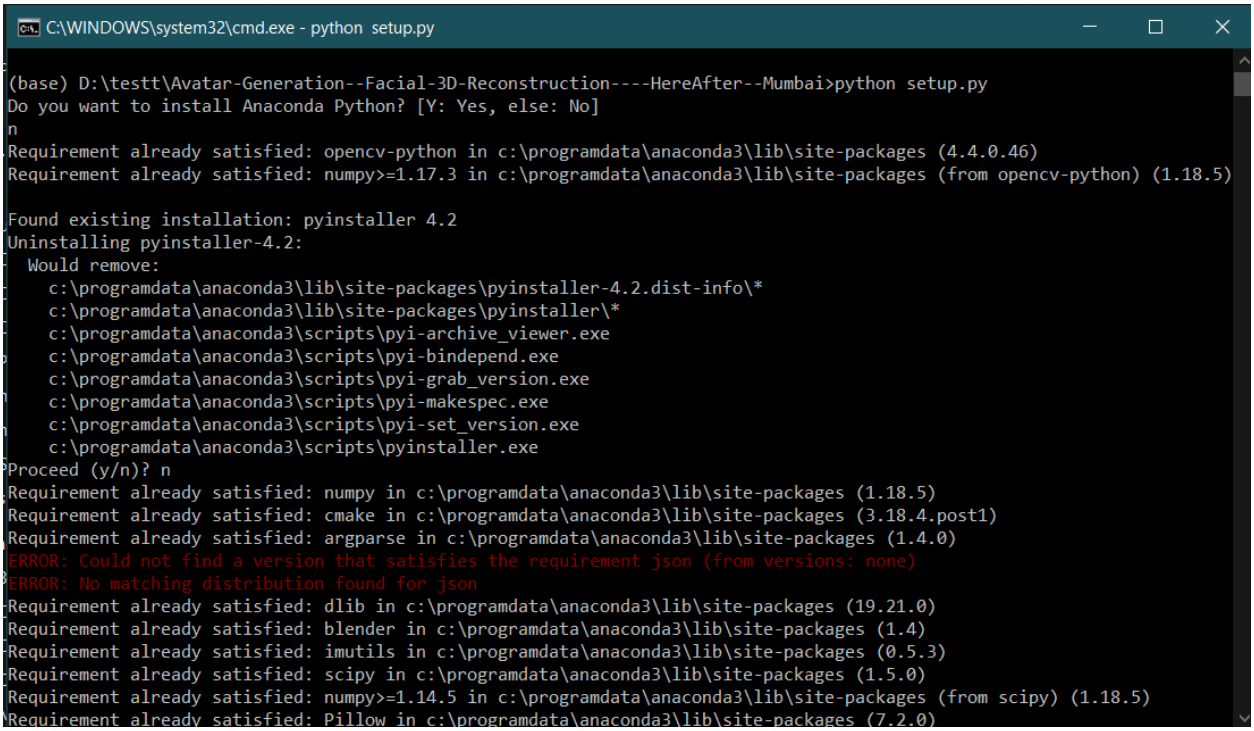
SCRIPT EXPLANATIONS

1. **68_landmark_detection.py**
   Using the 68 landmark data file, extract key facial features from given image file and store it as json file.
2. **5_landmark_detection.py**
   Simply converts the 68 landmark detection to extract 5 important coordinate systems to feed into the reconstruction pipeline.
3. **bake.py**
   Bake the generated 3d file to obtain required texture maps after rendering using blender automation using python blender package.
4. **blend.py**
   Transform generated 3d model using blender package automation to fit according to provided head mesh
5. **convert.py**
   Convert the generated obj file from the 3d face reconstruction pipeline to ply format for ease of manipulation and dedicated texture generation in blender environment.

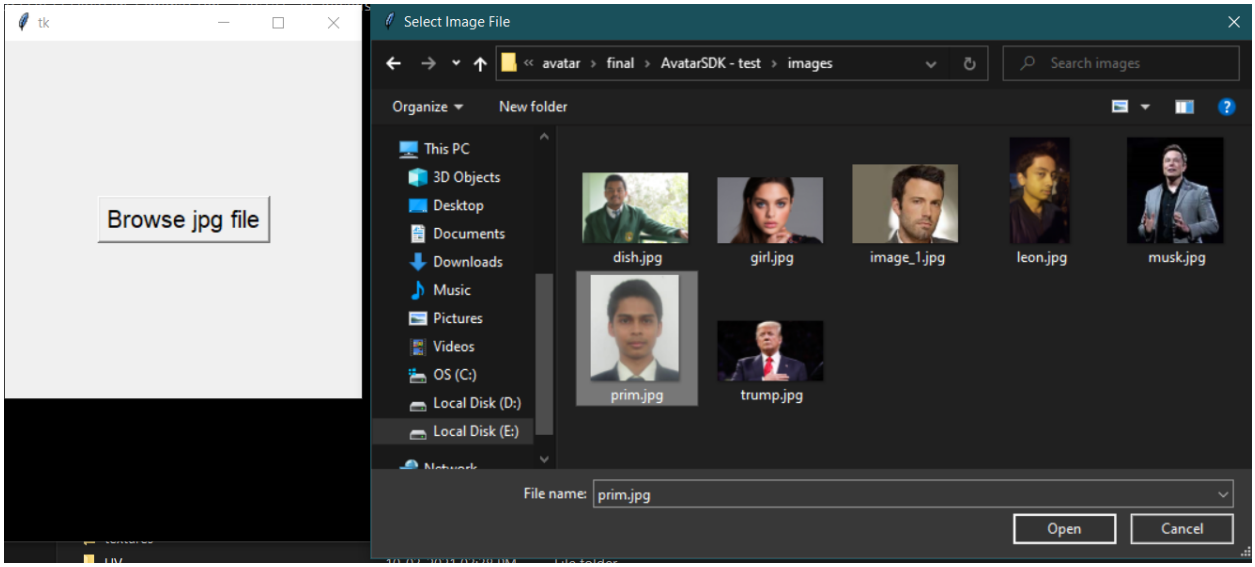
6. **face_decoder.py**
   Decodes image file and image text coordinate files to generate 3d face reconstruction model using the dedicated BFM models.
7. **generate_cord_py**
   Python script to generate the coordinate systems of both head mesh and generated face mesh in JSON format
8. **load_data.py**
   Load data to finally feed into dedicated pipeline.
9. **main_pipeline.py**
   Main python script orchestrating the entire program from acquiring image file to texture map generation, 3d face reconstruction, JSOn coordinate system generation, etc.
10. **main.py**
    Main python script dedicated to 3d face reconstruction to generated required 3d model in obj format
11. **Preprocessing_img.py**
    Deals with preprocessing acquired image file for feeding into 3d face reconstruction model
12. **render.py**
    Renders the obtained ply final face mesh to acquire 2D images to feed into Facial Feature detection subroutines again in order to obtain the final coordinate difference json file.
13. **setup.py**
    Installs prerequisites into the dedicated machine for python dependencies to work smoothly. It DOES NOT put dependencies to the path variable, this must be done manually.
14. **shape_predictor_68_face_landmarks.dat**
    68 landmark detection data file to be used by facial feature detection subroutines.

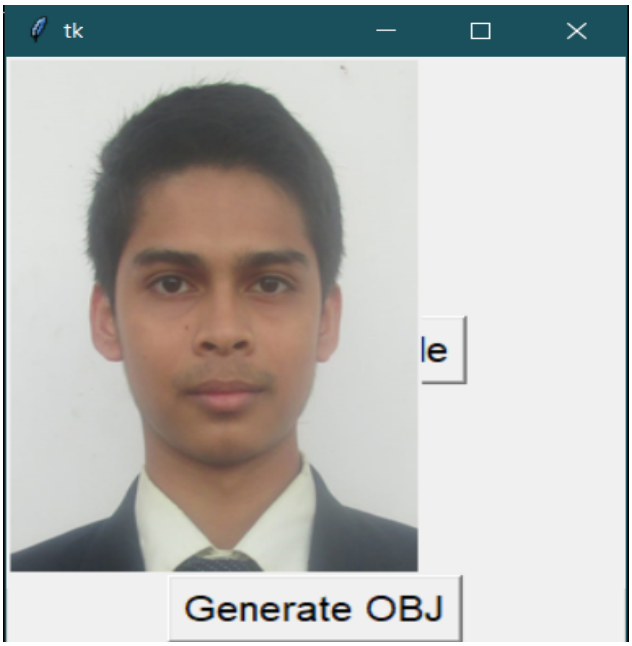
OUTPUT

Setup



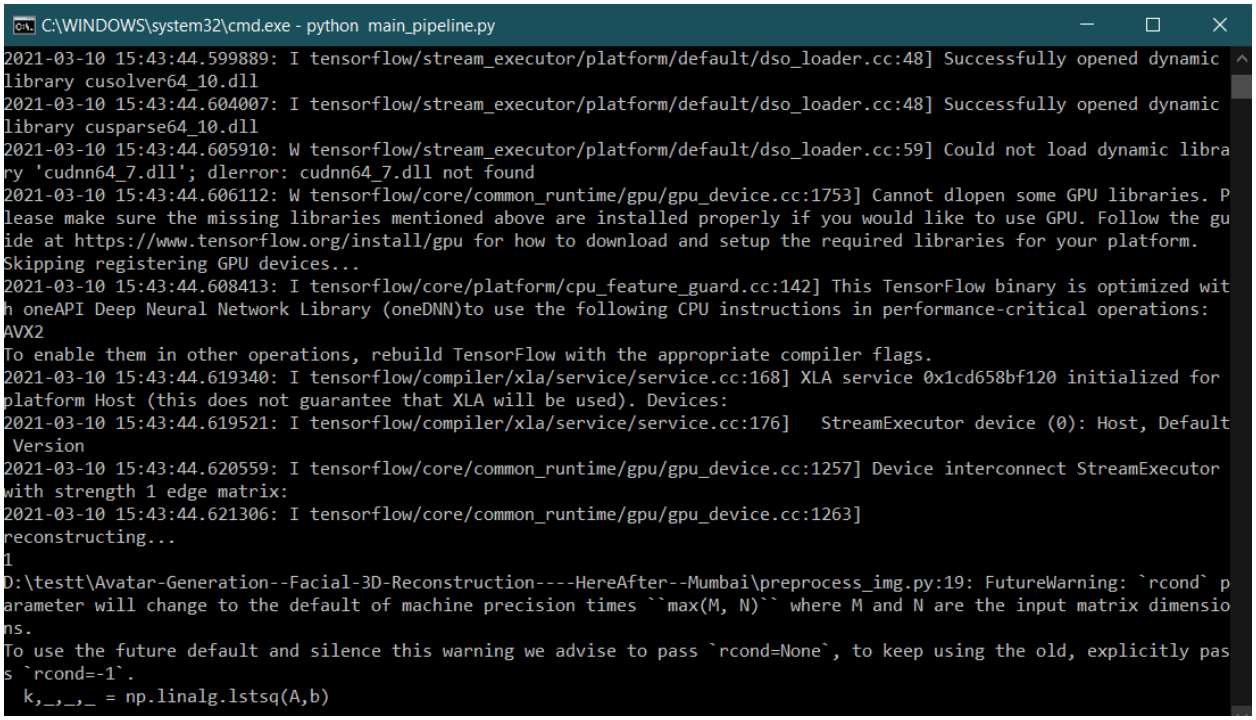
GUI for image browsing

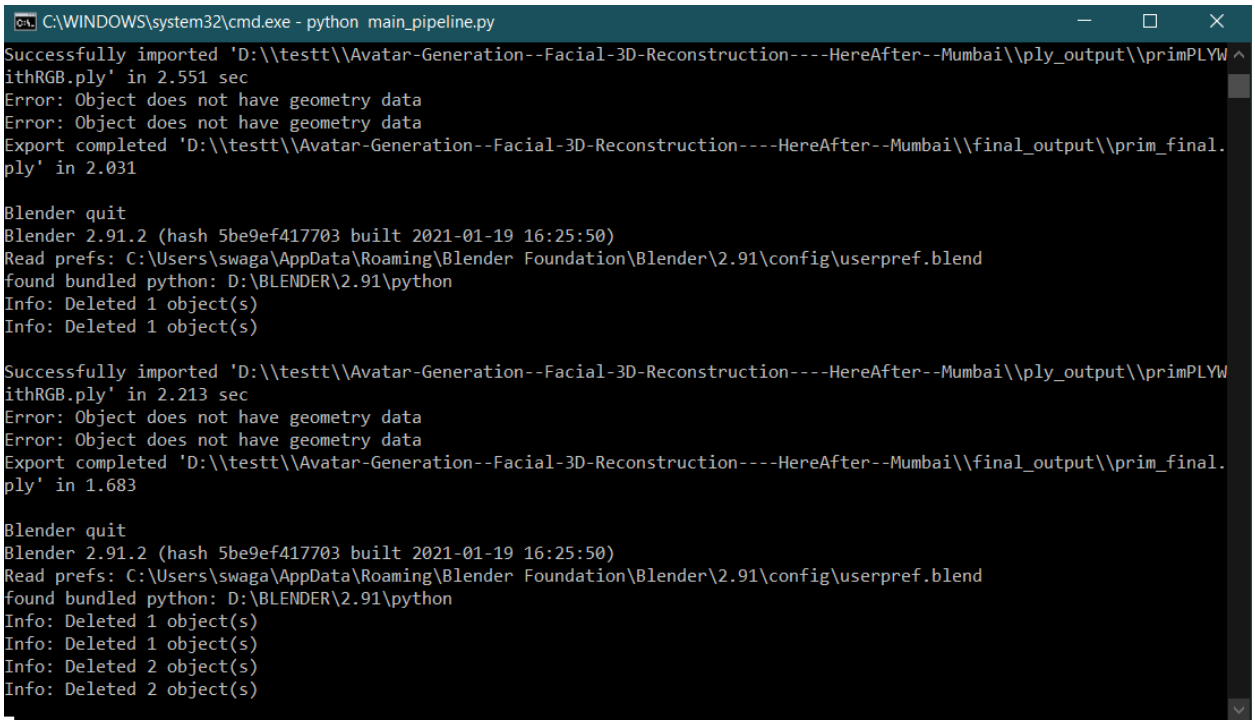




MAIN_PIPELINE TERMINAL SCREENSHOTS

```
C:\WINDOWS\system32\cmd.exe - python  main_pipeline.py
2021-03-10 15:43:44.599889: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic
library cusolver64_10.dll
2021-03-10 15:43:44.604007: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic
library cusparse64_10.dll
2021-03-10 15:43:44.605910: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic libra
ry 'cudnn64_7.dll'; dlerror: cudnn64_7.dll not found
2021-03-10 15:43:44.606112: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1753] Cannot dlopen some GPU libraries. P
lease make sure the missing libraries mentioned above are installed properly if you would like to use GPU. Follow the gu
ide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2021-03-10 15:43:44.608413: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized wit
h oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-critical operations:
AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-03-10 15:43:44.619340: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x1cd658bf120 initialized for
platform Host (this does not guarantee that XLA will be used). Devices:
2021-03-10 15:43:44.619521: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default
 Version
2021-03-10 15:43:44.620559: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1257] Device interconnect StreamExecutor
with strength 1 edge matrix:
2021-03-10 15:43:44.621306: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1263]
reconstructing...
1
D:\testt\Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai\preprocess_img.py:19: FutureWarning: `rcond` p
arameter will change to the default of machine precision times ``max(M, N)`` where M and N are the input matrix dimensio
ns.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pas
s `rcond=-1`.
  k,_,_,_ = np.linalg.lstsq(A,b)
```

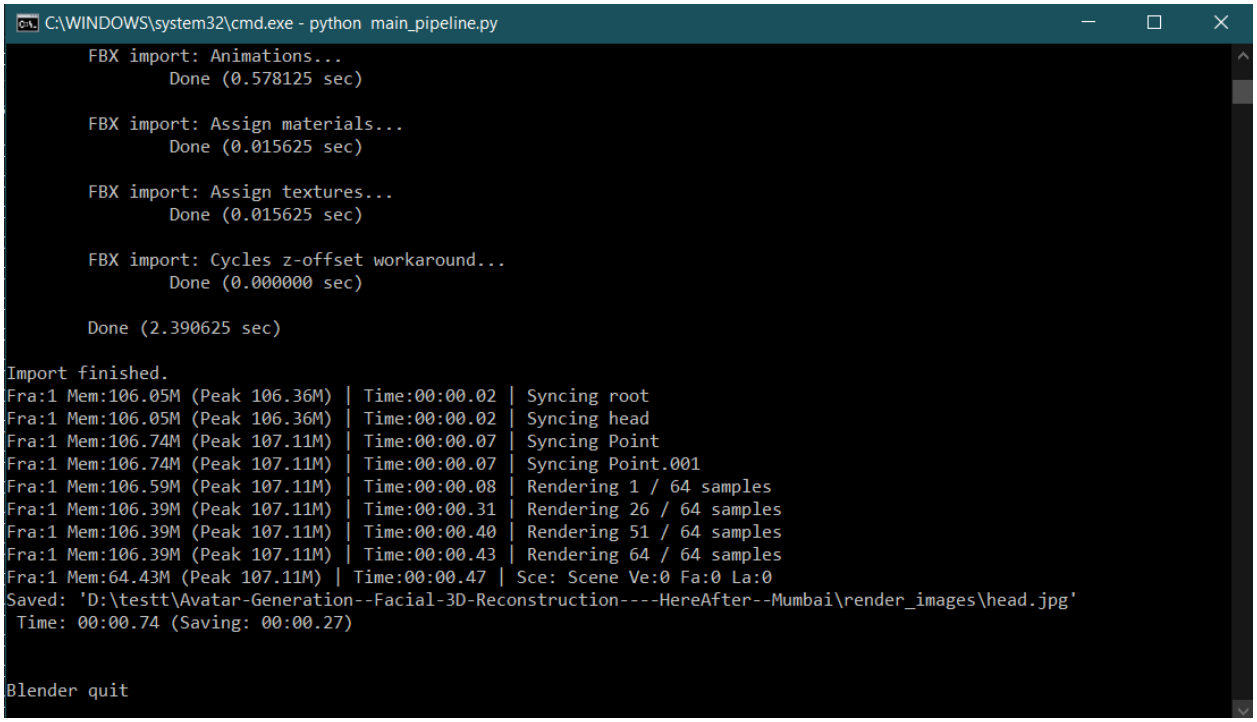

C:\WINDOWS\system32\cmd.exe - python main_pipeline.py
2021-03-10 15:43:44.599889: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cusolver64_10.dll
2021-03-10 15:43:44.604007: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cusparse64_10.dll
2021-03-10 15:43:44.605910: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library 'cudnn64_7.dll'; dlerror: cudnn64_7.dll not found
2021-03-10 15:43:44.606112: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1753] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2021-03-10 15:43:44.608413: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-critical operations:
AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-03-10 15:43:44.619340: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x1cd658bf120 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2021-03-10 15:43:44.619521: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default Version
2021-03-10 15:43:44.620559: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1257] Device interconnect StreamExecutor with strength 1 edge matrix:
2021-03-10 15:43:44.621306: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1263]
reconstructing...
1
D:\testt\Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai\preprocess_img.py:19: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  k,_,_,_ = np.linalg.lstsq(A,b)

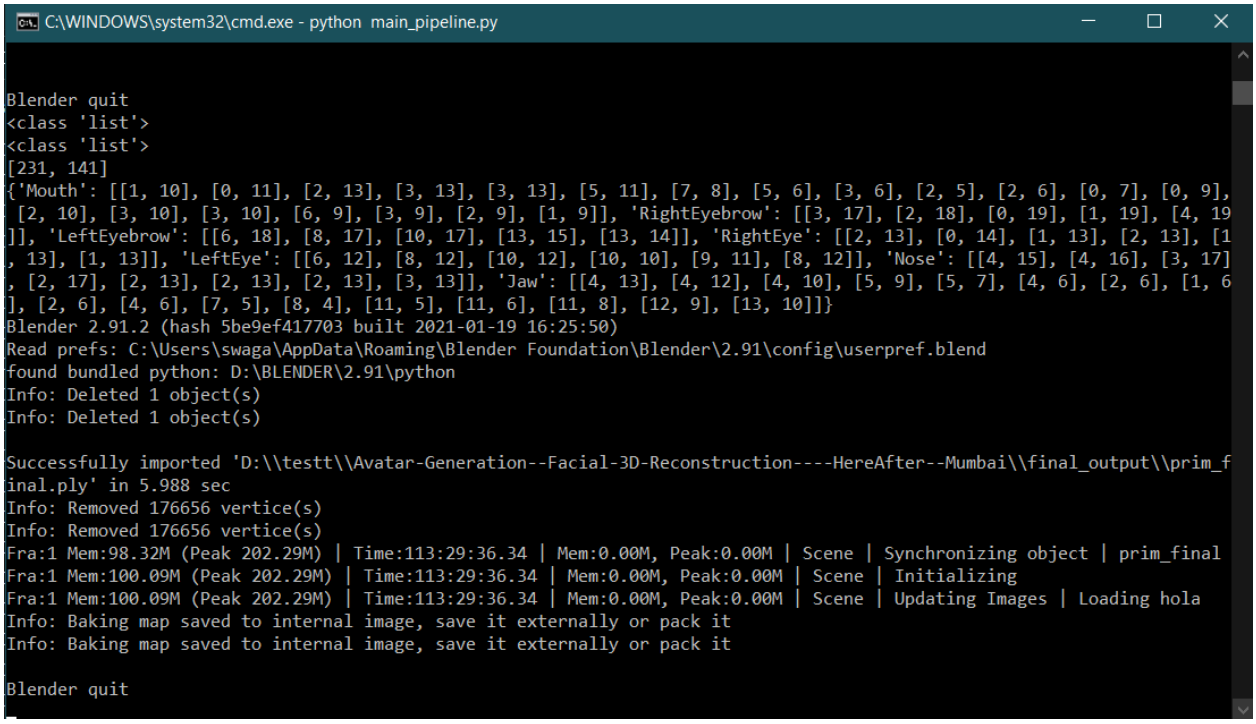

C:\WINDOWS\system32\cmd.exe - python main_pipeline.py
Successfully imported 'D:\\testt\\Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai\\ply_output\\primPLYWithRGB.ply' in 2.551 sec
Error: Object does not have geometry data
Error: Object does not have geometry data
Export completed 'D:\\testt\\Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai\\final_output\\prim_final.ply' in 2.031

Blender quit
Blender 2.91.2 (hash 5be9ef417703 built 2021-01-19 16:25:50)
Read prefs: C:\Users\swaga\AppData\Roaming\Blender Foundation\Blender\2.91\config\userpref.blend
found bundled python: D:\BLENDER\2.91\python
Info: Deleted 1 object(s)
Info: Deleted 1 object(s)

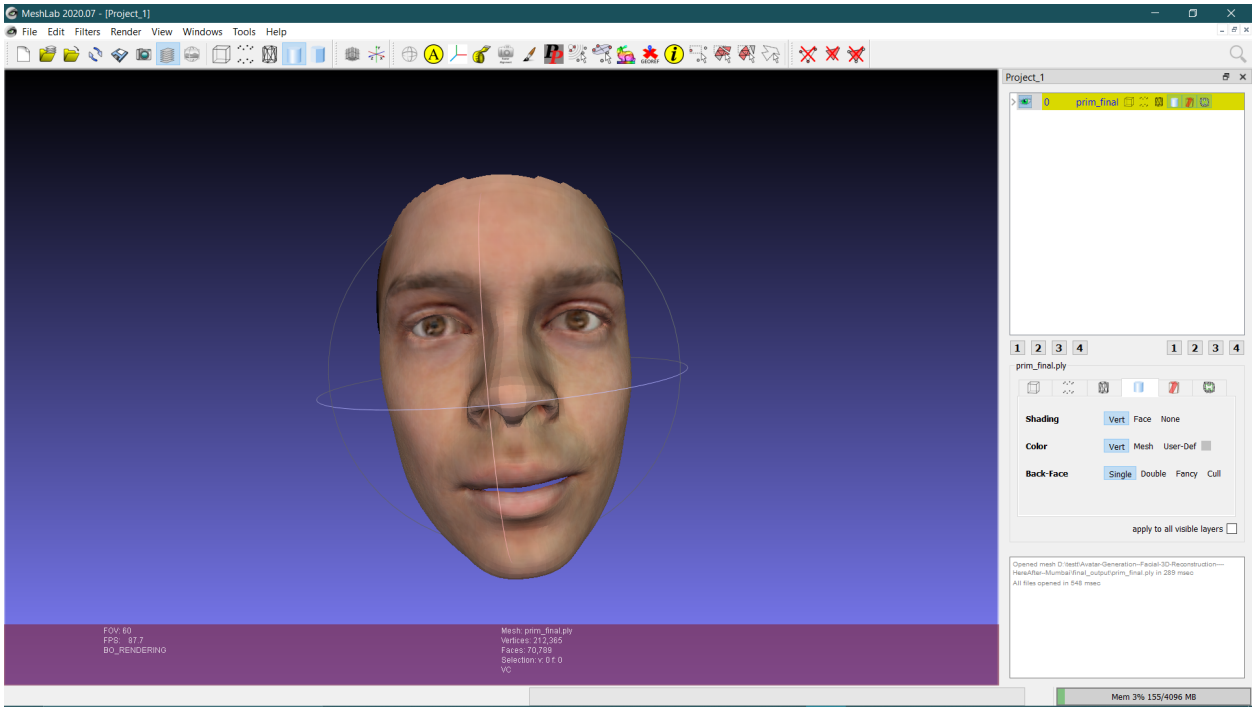
Successfully imported 'D:\\testt\\Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai\\ply_output\\primPLYWithRGB.ply' in 2.213 sec
Error: Object does not have geometry data
Error: Object does not have geometry data
Export completed 'D:\\testt\\Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai\\final_output\\prim_final.ply' in 1.683

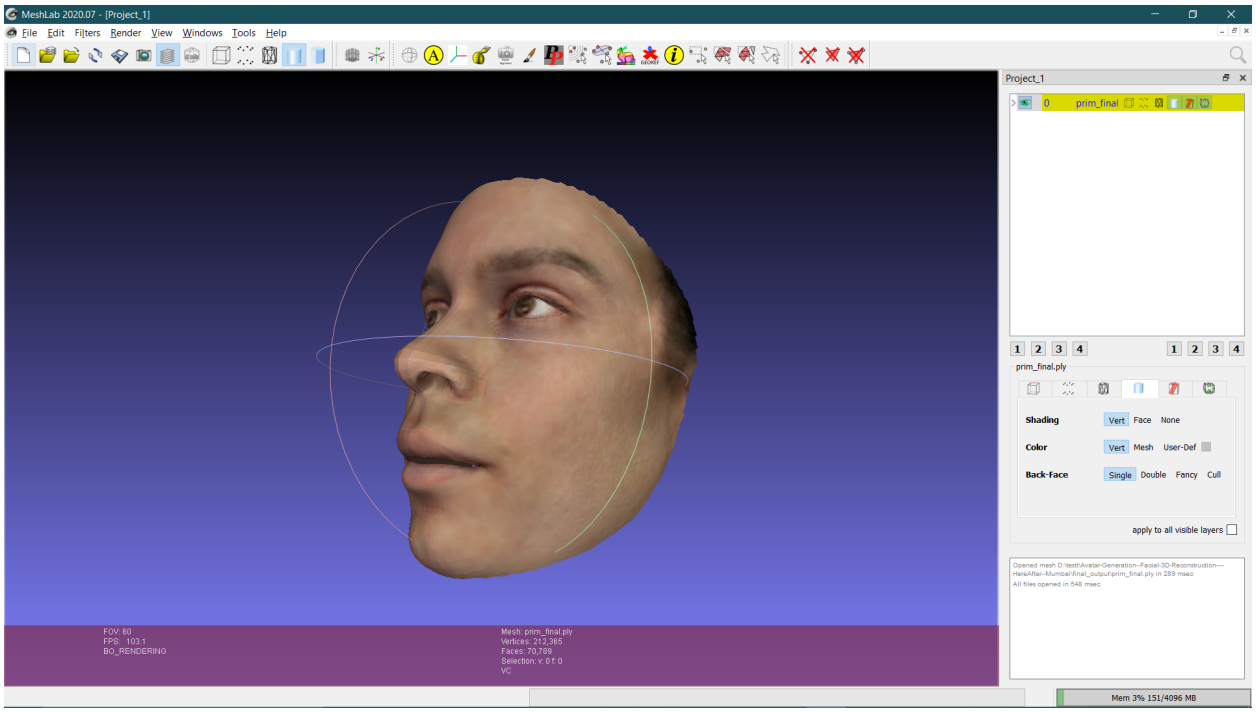
Blender quit
Blender 2.91.2 (hash 5be9ef417703 built 2021-01-19 16:25:50)
Read prefs: C:\Users\swaga\AppData\Roaming\Blender Foundation\Blender\2.91\config\userpref.blend
found bundled python: D:\BLENDER\2.91\python
Info: Deleted 1 object(s)
Info: Deleted 1 object(s)
Info: Deleted 2 object(s)
Info: Deleted 2 object(s)

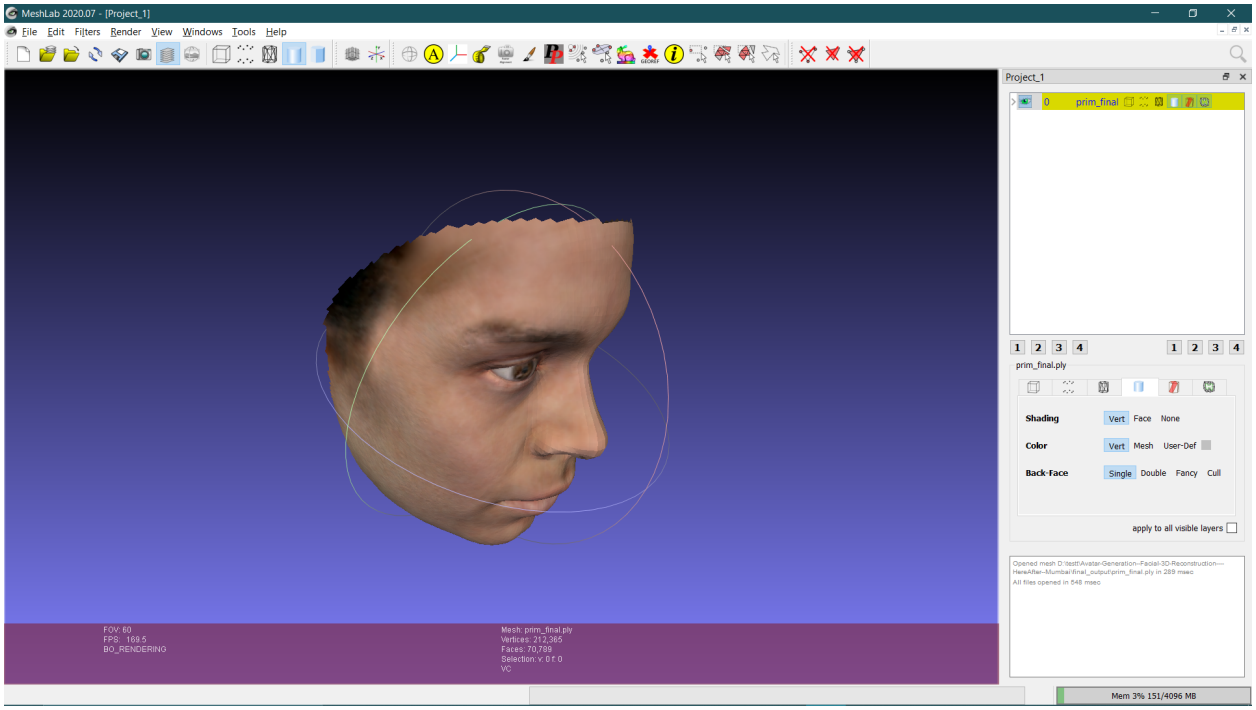

C:\WINDOWS\system32\cmd.exe - python main_pipeline.py
FBX import: Animations...
Done (0.578125 sec)
FBX import: Assign materials...
Done (0.015625 sec)
FBX import: Assign textures...
Done (0.015625 sec)
FBX import: Cycles z-offset workaround...
Done (0.000000 sec)
Done (2.390625 sec)
Import finished.
Fra:1 Mem:106.05M (Peak 106.36M) | Time:00:00.02 | Syncing root
Fra:1 Mem:106.05M (Peak 106.36M) | Time:00:00.02 | Syncing head
Fra:1 Mem:106.74M (Peak 107.11M) | Time:00:00.07 | Syncing Point
Fra:1 Mem:106.74M (Peak 107.11M) | Time:00:00.07 | Syncing Point.001
Fra:1 Mem:106.59M (Peak 107.11M) | Time:00:00.08 | Rendering 1 / 64 samples
Fra:1 Mem:106.39M (Peak 107.11M) | Time:00:00.31 | Rendering 26 / 64 samples
Fra:1 Mem:106.39M (Peak 107.11M) | Time:00:00.40 | Rendering 51 / 64 samples
Fra:1 Mem:106.39M (Peak 107.11M) | Time:00:00.43 | Rendering 64 / 64 samples
Fra:1 Mem:64.43M (Peak 107.11M) | Time:00:00.47 | Sce: Scene Ve:0 Fa:0 La:0
Saved: 'D:\testt\Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai\render_images\head.jpg'
Time: 00:00.74 (Saving: 00:00.27)
Blender quit


C:\WINDOWS\system32\cmd.exe - python main_pipeline.py
Blender quit
<class 'list'>
<class 'list'>
[231, 141]
{'Mouth': [[1, 10], [0, 11], [2, 13], [3, 13], [3, 13], [5, 11], [7, 8], [5, 6], [3, 6], [2, 5], [2, 6], [0, 7], [0, 9], [2, 10], [3, 10], [3, 10], [6, 9], [3, 9], [2, 9], [1, 9]], 'RightEyebrow': [[3, 17], [2, 18], [0, 19], [1, 19], [4, 19]], 'LeftEyebrow': [[6, 18], [8, 17], [10, 17], [13, 15], [13, 14]], 'RightEye': [[2, 13], [0, 14], [1, 13], [2, 13], [1, 13], [1, 13]], 'LeftEye': [[6, 12], [8, 12], [10, 12], [10, 10], [9, 11], [8, 12]], 'Nose': [[4, 15], [4, 16], [3, 17], [2, 17], [2, 13], [2, 13], [2, 13], [3, 13]], 'Jaw': [[4, 13], [4, 12], [4, 10], [5, 9], [5, 7], [4, 6], [2, 6], [1, 6], [2, 6], [4, 6], [7, 5], [8, 4], [11, 5], [11, 6], [11, 8], [12, 9], [13, 10]]}
Blender 2.91.2 (hash 5be9ef417703 built 2021-01-19 16:25:50)
Read prefs: C:\Users\swaga\AppData\Roaming\Blender Foundation\Blender\2.91\config\userpref.blend
found bundled python: D:\BLENDER\2.91\python
Info: Deleted 1 object(s)
Info: Deleted 1 object(s)
Successfully imported 'D:\\testt\\Avatar-Generation--Facial-3D-Reconstruction----HereAfter--Mumbai\\final_output\\prim_final.ply' in 5.988 sec
Info: Removed 176656 vertice(s)
Info: Removed 176656 vertice(s)
Fra:1 Mem:98.32M (Peak 202.29M) | Time:113:29:36.34 | Mem:0.00M, Peak:0.00M | Scene | Synchronizing object | prim_final
Fra:1 Mem:100.09M (Peak 202.29M) | Time:113:29:36.34 | Mem:0.00M, Peak:0.00M | Scene | Initializing
Fra:1 Mem:100.09M (Peak 202.29M) | Time:113:29:36.34 | Mem:0.00M, Peak:0.00M | Scene | Updating Images | Loading hola
Info: Baking map saved to internal image, save it externally or pack it
Info: Baking map saved to internal image, save it externally or pack it
Blender quit

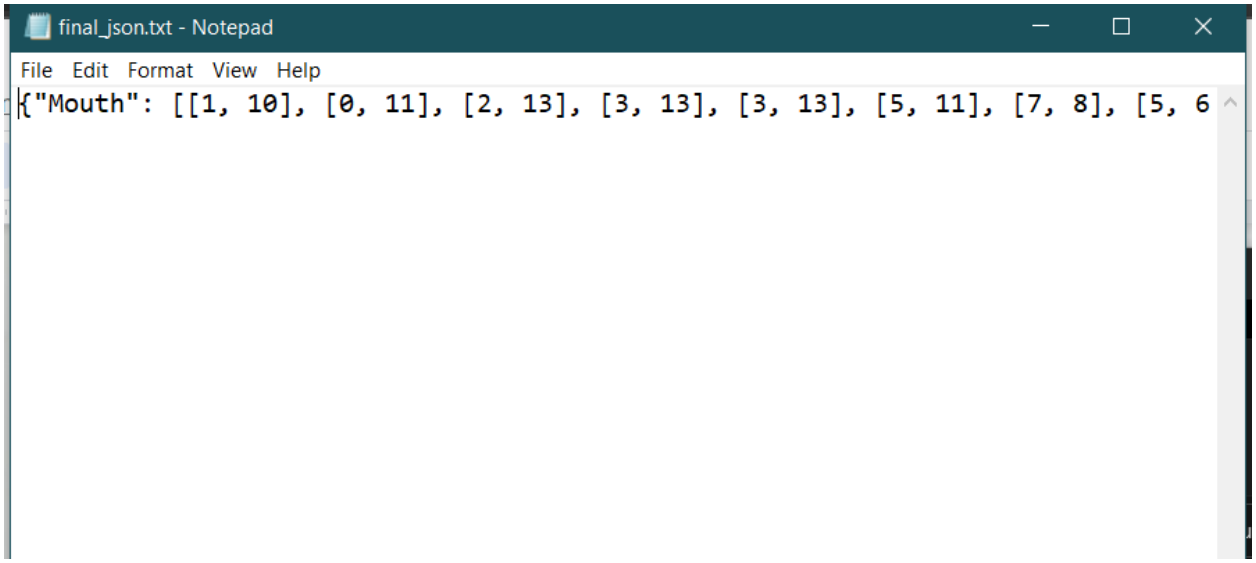
3D RECONSTRUCTED FACE







FINAL JSON FILE



{"Mouth": [[1, 10], [0, 11], [2, 13], [3, 13], [3, 13], [5, 11], [7, 8], [5, 6], [3, 6], [2, 5], [2, 6], [0, 7], [0, 9], [2, 10], [3, 10], [3, 10], [6, 9], [3, 9], [2, 9], [1, 9]], "RightEyebrow": [[3, 17], [2, 18], [0, 19], [1, 19], [4, 19]], "LeftEyebrow": [[6, 18], [8, 17], [10, 17], [13, 15], [13, 14]], "RightEye": [[2, 13], [0, 14], [1, 13], [2, 13], [1, 13], [1, 13]], "LeftEye": [[6, 12], [8, 12], [10, 12], [10, 10], [9, 11], [8, 12]], "Nose": [[4, 15], [4, 16], [3, 17], [2, 17], [2, 13], [2, 13], [2, 13], [3, 13]], "Jaw": [[4, 13], [4, 12], [4, 10], [5, 9], [5, 7], [4, 6], [2, 6], [1, 6], [2, 6], [4, 6], [7, 5], [8, 4], [11, 5], [11, 6], [11, 8], [12, 9], [13, 10]]}

FINAL TEXTURE MAP (PNG FORMAT)

NOTES

- ***Both the Blender environment Software and the pip Blender Package Module will be required for functioning of the rendered 3d face reconstructions.***
    - `pip install blender`
- ***Blender has to be inculcated into the root of the system environment variables Path***
- ***JSON has to be inculcated into the root of the system environment variables Path***
- ***Python 3.8 is suggested as Anaconda doesn't yet support Python 3.9 dependencies, hence downloading Python through Anaconda is highly advised***
- ***Python has to be installed to root of the system environment variables Path***
- ***dlib dependency requires Visual Studio to build in the [arent system, hence Visual Studio must be installed in the particular server host***
- ***In the setup.py file, an option to extract the data.rar file manually is given. It is obsolete as data.rar was only present in the beta version, hence it is advised to proceed by saying no ('n')***
- ***It is advised to rely on the python scripts for execution and not the generated .exe files while installing into the server.***
- ***Tkinter Python GUI is used for ease of beta testing, which can be easily modified to accommodate what files to select for avatar generation.***
- ***Unreal python module package can be installed, but can be used as long as it only automates unreal functions, and nothing else.***

Citation

Please cite the following paper if this model helps your research:

```
@inproceedings{deng2019accurate,
   title={Accurate 3D Face Reconstruction with Weakly-Supervised Learning: From Single Image to Image Set},
   author={Yu Deng and Jiaolong Yang and Sicheng Xu and Dong Chen and Yunde Jia and Xin Tong},
   booktitle={IEEE Computer Vision and Pattern Recognition Workshops},
   year={2019}
}
```

References

1. @microsoft/Deep3DFaceReconstruction Deep 3D face Reconstruction GitHub
2. @raviranjan0309/Detect-Facial-Features Detect Facial Features GitHub
3. @nabeel3133/file-converter-.obj-to-.ply GitHub