ROBT 310 - Final Project Progression report

Danissa Sandykbayeva
Robotics and Mechatronics
Almaty, Kazakhstan

Nurlan Kabdyshev
Robotics and Mechatronics
Almaty, Kazakhstan

Azhara Naubetova
Robotics and Mechatronics
Almaty, Kazakhstan

***Abstract*—This progression report contains some updates on our project. We introduced the programs we used, as well as libraries and examples of codes that were used. The progress could be**

I. Introduction

How much time we have spent:: At the beginning we spent two days to decide about the topic itself. After that we already started to choose a suitable platform and work on our code. As a result we are working on our project for a week, including decision making process and working on code.

Language and Program:: As main language of our program we used Python. Python is a high-level genera-programming language, which is widely used and has a lot of beneficial libraries. As a collaboration platform we used GitHub, which allows us to work together and keep track of changes.

Animation:: For animation we will use specific libraries, which are listed and described in Materials part. Overall, animation is based on face tracking.

II. Progress Report

****All of the changes and progress results in a .gif format could be seen in this Github Repository**

Facial Landmarks Detection

To realize our project, we will need to detect face features in real time. For this, we are going to use OpenCV library, especially Dlib. This algorithm detects identifiable face features, like eyes, brows, nose, mouth, lips. It creates map of points, which highlight those features.

To do this, Dlib has functions get_frontal_face_detector() and shape_predictor(), however predictor needs a pre-trained model, which we have downloaded from the internet, the link shown in the Github. The map could be used to assign controls to facial features. Since our project is focused on real time control, we need to use webcam too. OpenCV also has such function. We have made the gif of our output.

Blender Python Scripting

For our project, we are using Blender 2.92.0 Python API (bpy) which allows developers to incorporate Python scripts into Blender. Using this library's capabilities we are going to create an add-on that would use PC's camera in order to then detect facial landmarks and mark them on the 3D avatar created in Blender. To start out we got ourselves familiar with Python scripting in Blender. In particular, we created a testing

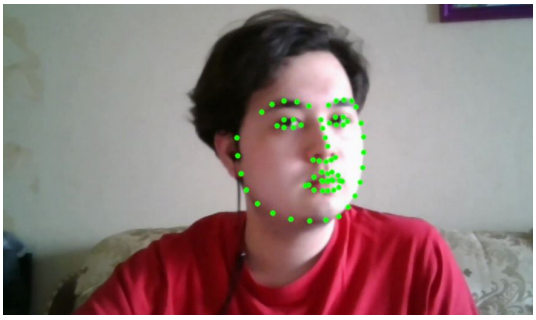

Fig. 1. Visualization of the facial landmark generated by dlib

custom panel (shown on the gif in **Github**) named "Custom Panel" under "My Custom tab". This panel is here just to test out some basic functionalities that are available in Blender's Python scripting, such as:

- Creating and laying out panels in add-ons
- Creation of the new object (creating a new cube and UV sphere mesh)
- Changing current object's properties (Euler's rotation, scaling along various axes, etc.)
- Changing viewport display options (displaying object bounds)

Those scripting properties would then be used to incorporate our facial detection algorithm into Blender and bound it with an existing avatar with a possible ability to tweak some properties of it.

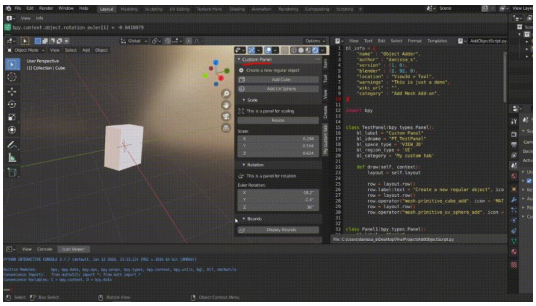

Fig. 2. Visualization of the facial landmark generated by dlib

III. Materials

- **Numpy** - highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays.

- **OpenCV-Python** - Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language. OpenCV-Python is a Python wrapper around original C++ implementation which makes it an appropriate tool for fast prototyping of computer vision problems.
- **'dlib'** - library for detection of the facial landmarks. 'dlib' uses a labeled iBUG 300-W dataset in order to train the predictor for the facial features. Each facial map like that consists of 68-point mappings. The facial landmark mapping mentioned earlier can bee sen on Fig. 1

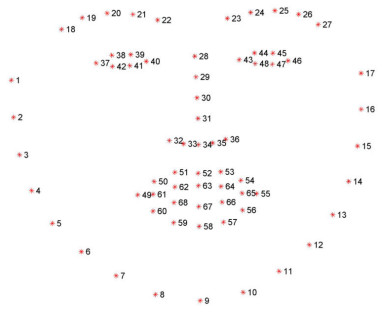

Fig. 3. Visualization of the facial landmark generated by dlib

- **Blender** - free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, video editing and 2D animation pipeline.
- **blenderpy** - library meant for installation into a virtualenv or wherever, for unit testing of Blender extensions being authored, or developement of a Blender 3d-enabled Python application.

REFERENCES

[1] https://opencv.org/
[2] https://towardsdatascience.com/detecting-face-features-with-python-30385aee4a8e
[3] https://pypi.org/project/bpy/
[4] https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9009369&fileOId=9009370