Created By: Andrew Palmer
Github: ahpalmerUNR

Mouth Music Tongue Detector

Version 1.0

January 30, 2021

Contents

1 Installation 5
1.1 Download Code 5
1.2 Dependencies 5
1.2.1 How to open Windows PowerShell 6
1.2.2 Update Pip 8
1.2.3 Install Numpy 8
1.2.4 Install PyTorch 8
1.2.5 Install OSCPy 8
1.2.6 Installing OpenCV 8
1.2.7 Installing Other Dependencies 9
1.3 Create Shortcut 9

2 Mouth Model System 11
2.1 Stream 12
2.2 Stream and Record 13
2.3 Positioner 14
2.4 Settings 15
2.5 Test Speed 16

3 Positioner Tool 17
3.1 How to Position 17
3.2 Positioner Buttons and Projection Circle 19

4 Changing Machine Learning Models 21

Index 23

Installation

The detection program has been tested on a **Windows 10** virtual machine running **Python 3.8**. This follows the system specifications provided by the client. The following section details how to install the dependencies and create an executable shortcut to run the system without using the command prompt.

1.1 Download Code

The code for this detection system is found at `https://github.com/ahpalmerUNR/mouthMusicStreamer`. Download this repository onto your machine either through the Github Windows Desktop App or by .zip file download. Place the code in a place you can keep track of, like the **Desktop** or **Programs** folder.

1.2 Dependencies

Though some older versions of the dependencies might work, the versions tested are as follows.

- OpenCV 4.1.5
- numpy 1.19.5
- PyTorch 1.7.1
- oscpy 0.5.0
- PIL 8.1.0
- tkinter 8.6

If you already have these requirements satisfied, you can skip to Section 1.3.

1.2.1 How to open Windows PowerShell

To install the various dependencies, it is easiest to use a terminal or command prompt.

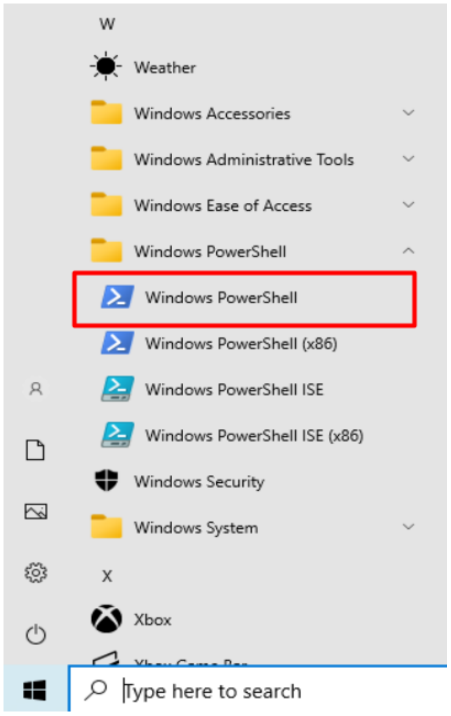
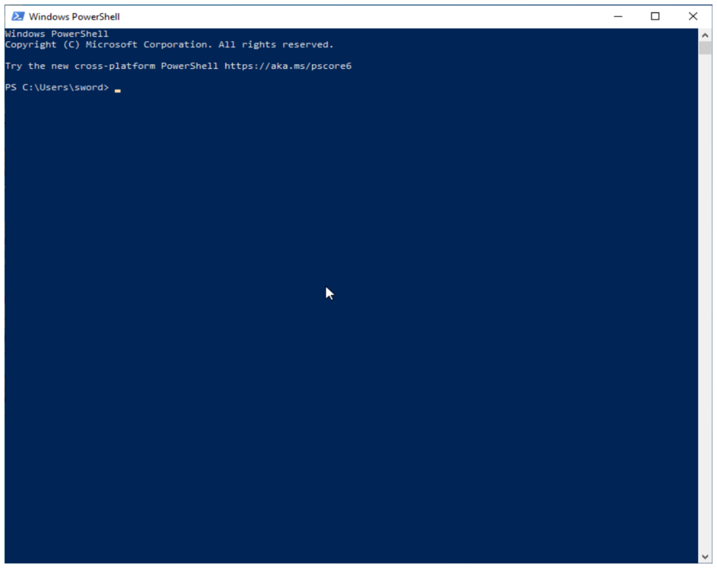


Fig. 1.1 Windows Powershell from Start Menu

On Windows 10, the system should already have Windows PowerShell installed. To open it from the **Start** menu go to **W** and select **Windows PowerShell**, Figure 1.1.

This should open a terminal that you can use to run the rest of the commands in this section, Figure 1.2.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\sword> _
```

Fig. 1.2 Windows PowerShell Terminal

1.2.2 Update Pip

Pip is a Python module installer. In an open terminal, enter the following command to update the Python Pip installation tool for Python modules.

```
C:\Users\user> python -m pip install --upgrade pip
```

1.2.3 Install Numpy

Numpy is a common Python library used in our program. To install Numpy, enter the following command.

```
C:\Users\user> pip install numpy
```

1.2.4 Install PyTorch

Our machine learning model is created and run using PyTorch. Most versions of PyTorch should work fine to run the detection system. To find non-pip install instructions, visit `https://pytorch.org/get-started/locally/`. Otherwise, run the following command in the terminal.

```
C:\Users\user> pip install torch==1.7.1+cpu torchvision==0.8.2+cpu
 torchaudio===0.7.2 -f https://download.pytorch.org/whl/torch_stable.html
```

1.2.5 Install OSCPy

In order to stream the data using an OSC client, the OSCPy package is needed. To install this, run the following command.

```
C:\Users\user> pip install oscpy
```

1.2.6 Installing OpenCV

There are many ways to install OpenCV. Go to `https://opencv.org/releases/` for different methods of installation and OpenCV versions. For convenience, the pip installation command automatically links OpenCV for Python to find.

```
C:\Users\user> pip install opencv-python
```

Important: you may encounter and error related to Visual C++ when running the program. To fix this, download and run the following program. `https://aka.ms/vs/16/release/vc_redist.x64.exe`

1.2.7 Installing Other Dependencies

Other dependencies should follow similar installation techniques if errors are encountered during installation. Both PIL and tkinter should not need to be installed. They should come with Python 3.8.

1.3 Create Shortcut

Creating a shortcut to run the program is relatively straight forward. In the repository directory, select the **mouth_music_stream.py** file using **Right Click**. Then select **Create shortcut** from the menu. This will create a shortcut object in the directory, **mouth_music_stream.py - Shortcut**. Move this to the **Desktop** to run the detector without having to open a terminal.

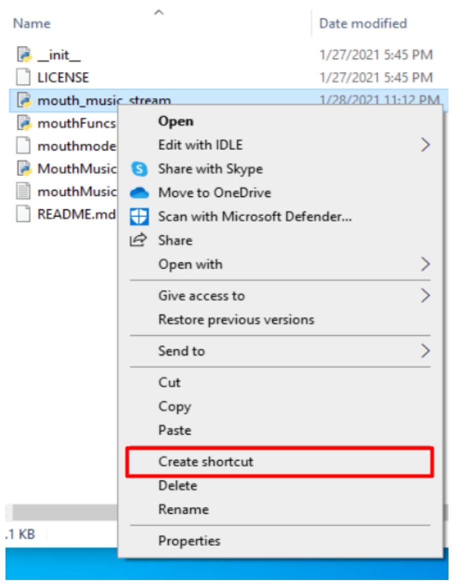


Fig. 1.3 Create Shortcut

Mouth Model System

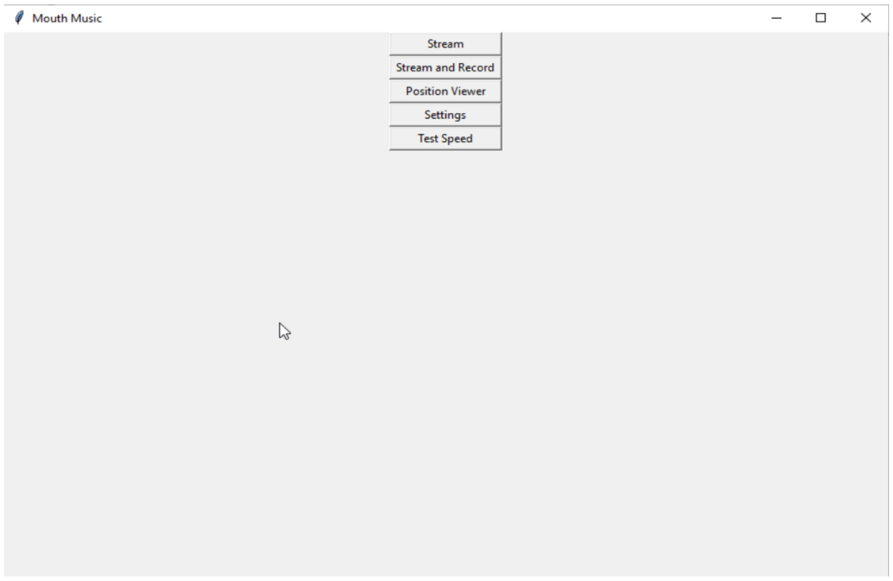


Fig. 2.1 Mouth Music Home Screen

When the program starts, you should see the home screen, Figure 2.1. This home screen has several options. Each one is described below.

2.1 Stream

This option starts the streaming of position data through an OSC client. Streaming is done without visual so as to reduce the latency within the system. While streaming, only a **Back** button will be visible. This button will stop the OSC client stream and return to the home screen. The streaming screen is shown in, Figure 2.2.

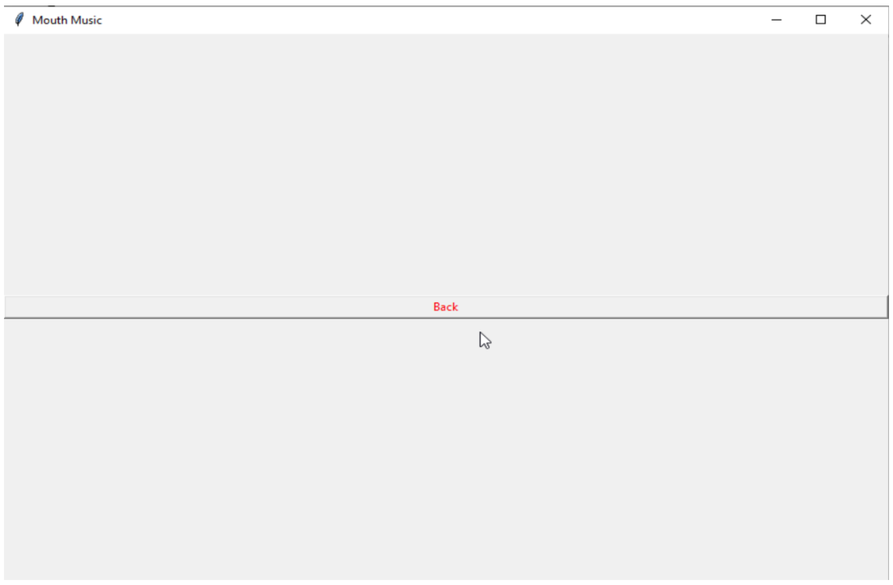
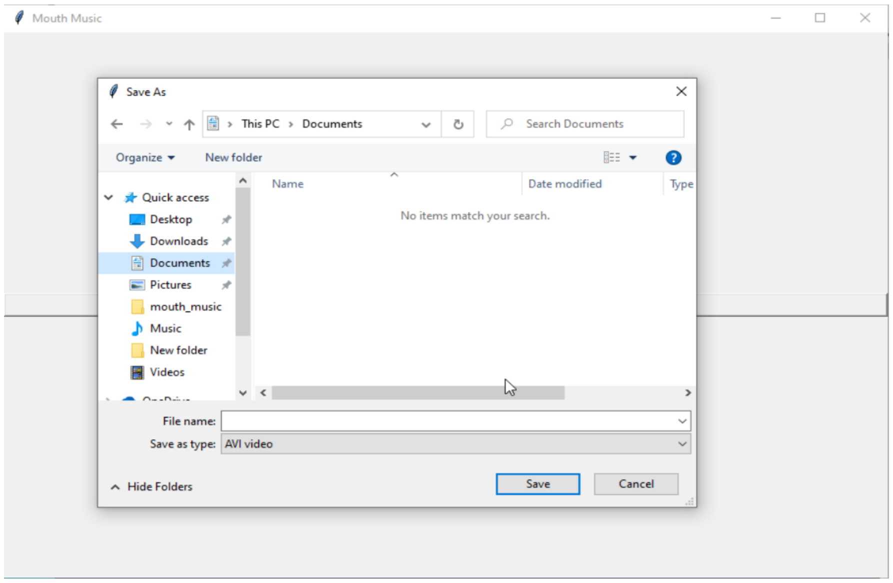


Fig. 2.2 Streaming Screen

2.2 Stream and Record

This option allows the user to select an .avi file to save the captured images to, Figure 2.3. After a name is selected, the system starts streaming position data through an OSC client and saving the images to .avi file. Though the images are saved, streaming is done without visual so as to reduce the latency within the system as much as possible. While streaming, only a **Back** button will be visible. This button will stop the OSC client stream and return to the home screen. The streaming screen is shown in, Figure 2.2.



No items match your search.

File name:

Save as type: AVI video

Fig. 2.3 Record File Selection Screen

2.3 Positioner

This option allows the user to visually see the webcam stream, the detections from the model, and the mouth offset and projection circle. The positioner has it’s own chapter detailing the usage.

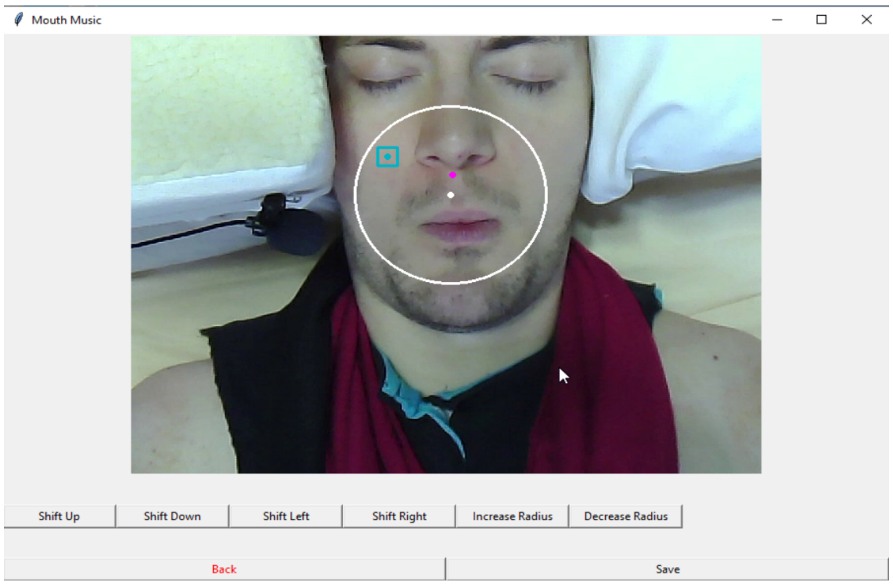
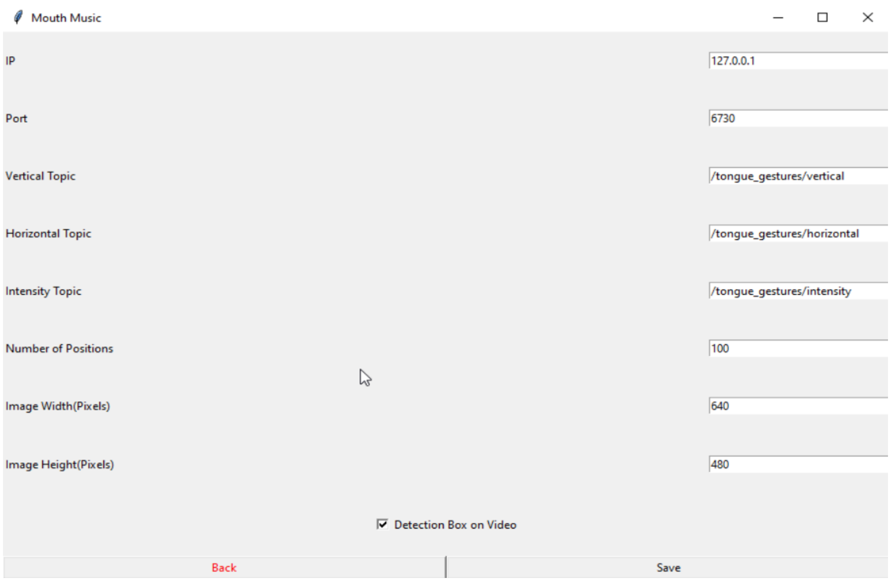


Fig. 2.4 Positioner Tool

2.4 Settings

This option allows the user to specify the various OSC stream settings: ip address, port, and position topics. The user can also select the range of integers sent to the position topics (0-100 default), the input image size, and whether or not the detection box appears in the recorded video. The user must press the **Save** button to commit the changes to the settings.


Mouth Music

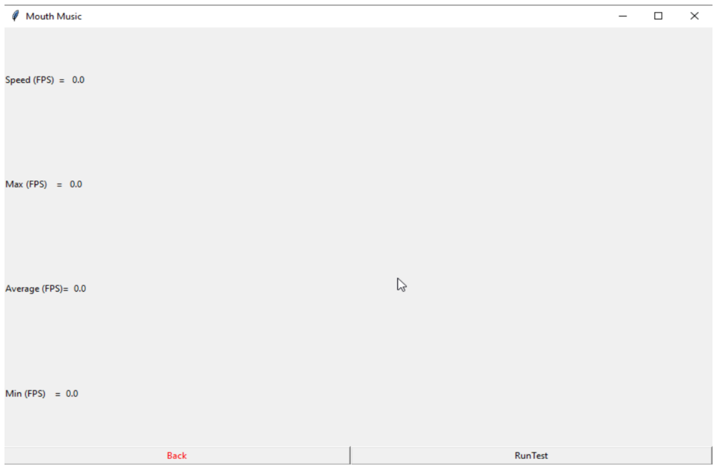
IP	127.0.0.1
Port	6730
Vertical Topic	/tongue_gestures/vertical
Horizontal Topic	/tongue_gestures/horizontal
Intensity Topic	/tongue_gestures/intensity
Number of Positions	100
Image Width(Pixels)	640
Image Height(Pixels)	480

[✓ Detection Box on Video]

Fig. 2.5 Settings Screen

2.5 Test Speed

This option allows the user to test the speed of the detection system. To run a test, the user should press **Run Test**. This will initiate a 120 frame test from the webcam. The first 30 frames are disregarded to allow the OSC client and model to get up to speed. The next 90 frames are timed for how long it takes to get a frame, pass it through the system, and submit the position information to the OSC server. The resulting statistics are displayed to the user.



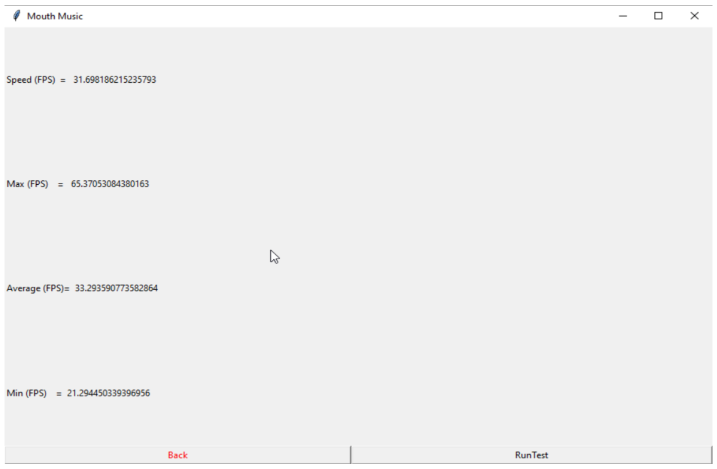
```
Speed (FPS)  =  0.0

Max (FPS)   =  0.0

Average (FPS)=  0.0

Min (FPS)   =  0.0
```

Fig. 2.6 Speed Test Screen



```
Speed (FPS)  =  31.698186215235793

Max (FPS)   =  65.37053084380163

Average (FPS)=  33.293590773582864

Min (FPS)   =  21.294450339396956
```

Fig. 2.7 Speed Test Results Screen

Positioner Tool

3.1 How to Position

The positioner tool allows the user to view the detections while the webcam is streaming to determine the best placement of the webcam. This tool is shown in Figure 3.1 for reference.

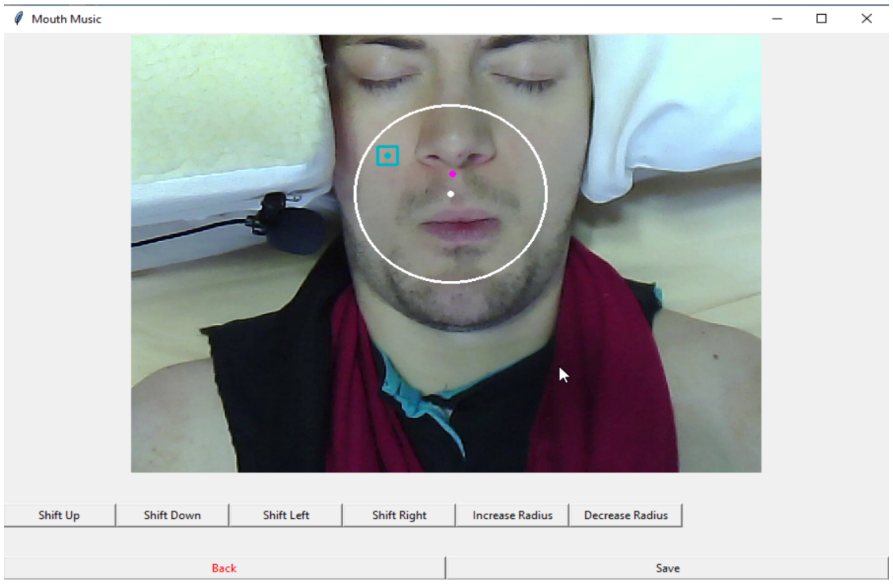


Fig. 3.1 Positioner Tool

The best placement will satisfy the following placement conditions.

1. The lip detection (Pink Dot) will align with the middle of the top lip. Roughly the middle of Cupid's Bow (point between the Philtrum Ridges). Figure 3.1 has

poor lip detection placement, and the camera should be moved until placement is better.

2. Detections identified by the blue box and point should correspond to the tip of the tongue. It is best to moved the tongue inside the mouth slowly in a circle (about where the white line is in Figure 3.1). If a particular region experiences poor detection, repositioning the camera or the user's head may be needed.
3. The blue detection square should appear purple when the user's tongue protrudes through the lips. This is shown in Figure 3.2 (keep in mind this figure has poor camera placement).

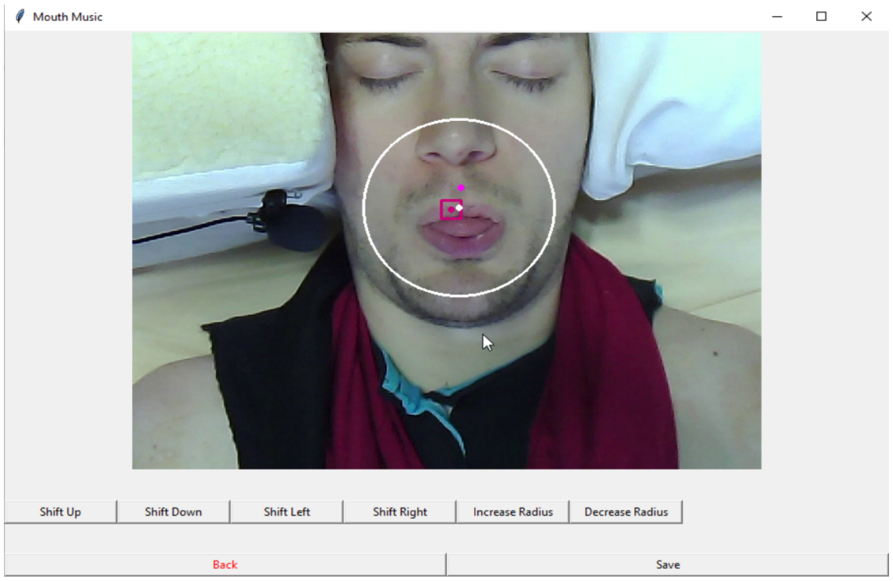


Fig. 3.2 Positioner Tool Tongue Sticking out

3.2 Positioner Buttons and Projection Circle

The positioner tool has a series of buttons to adjust the projection circle. Because the circle is used to identify the bounds of tongue placement, any detections outside the circle are projected onto the edge instead. Getting the circle placed appropriately will help increase the fidelity between tongue positions and desired OSC streamed values.

Shift Up–moves the center of the projection circle up one step.
Shift Down–moves the center of the projection circle down one step.
Shift Left–moves the center of the projection circle left one step.
Shift Right–moves the center of the projection circle right one step.
Increase Radius–increases the radius of the projection circle one step.
Decrease Radius–decreases the radius of the projection circle one step.

For the projection circle to be placed appropriately, the circle must satisfy the following requirements

1. The center of the projection circle should align with the center of the mouth opening.
2. The radius of the projection circle should not exceed the bounds of the user's face.

In order for the user to get a square field of detection points, the projections are cast onto a square inscribed in the projection circle. For ease of use, the tongue protruding from the lips is cast as half the maximum value on the horizontal position and 0 on the vertical position.

Note: It is important to run the positioner each time the webcam is moved to identify any anomalies created by the shift in camera position.

Changing Machine Learning Models

The model is designed only for the contracting client. Because of this person unique design, the mouth model is not likely to work with any other users of the system. If users other than the contracting client wish to change machine learning models to something trained for them, or for some wider audience, there are several things to pay attention to in order to make this transition easier.

The **import** statements at the top of **mouth_music_stream.py** should reflect the file names of the new model.

The model should implement a **forward** function that takes color image structured using the **getFrame** function in **mouth_music_stream.py**. This function should return a tensor.

There should be a **loadModel** function that takes a directory and file name as inputs and returns the loaded model as an output.

To limit the amount of code needing to be changed, it may be easier to implement a **setGridSize** function, even if it is just an empty function.

Along with the model should be a **decodeLabel** function to process the model output. This should return a dictionary with at least the following keys.

1. **lipConf**–lip confidence value
2. **tongueConf**–tongue confidence value
3. **intensity**–detection intensity
4. **tongueOut**–probability the tongue is protruding through the lips
5. **lipPosition**–tuple with the (x,y) position of the lip detection point
6. **tonguePosition**–tuple with the (x,y) position of the tongue detection

These are the minimal number of functions needing to be changed should the model be replaced. Additional changes may be necessary to achieve desired behavior for other machine learning models.

Index

A

align 17
anomalies 19

B

Back button 12, 13
best placement 17

C

center 19
change machine learning models 21
circle 18
commands 6
Cupid’s Bow 17

D

decrease 19
dependencies 5, 9
Desktop 9
detection box 15
detection square - blue 18
detection square - purple 18
detection system 5, 16
detections 17
directory 9
down 19
download 5

E

edge 19
error - Visual C++ dependency 9

F

face 19

H

head - user’s 18
home screen 11
horizontal 19

I

increase 19
input image size 15
install 5
ip address 15

L

left 19
lip detection 17
lips 19

M

maximum value 19
mouth 19
mouth model 21

N

new model 21
number of positions 15
numpy 8

O

offset 14
OpenCV 8
OSC 8, 19
OSC - settings 15

OSC - position topics 15
OSC Client 16
OSC client 12, 13
OSCPy 8

P

person unique design 21
Philtrum Ridges 17
PIL 9
pip 8
placement 17
placement conditions 17
poor detection 18
port 15
positioner 14, 17, 19
positioner buttons 19
PowerShell 6
program start 11
projection circle 14, 19
projection circle placement 19
protrudes 18, 19
Python 5, 8
PyTorch 8

R

radius 19
record 13
recorded video 15
repositioning the camera 18
repository 5
requirements 17, 19
right 19
Run Test 16
run the detector 9

S

save 13, 15
Save button 15
save settings 15
shortcut 9
specifications 5
speed 16
square 19
statistics 16
stop 12
streaming 12
streaming screen 12

T

terminal 6, 8, 9
tkinter 9
tongue 18
tongue placement 19
tongue tip 18
top lip 17

U

up 19

V

versions - dependencies 5
vertical 19
visual 14

W

webcam 14, 16, 17, 19
Windows 10 6