

Created By: Andrew Palmer  
Github: [ahpalmerUNR](#)

# Mouth Music Tongue Detector

Version 2.0

May 7, 2021

Copyright © 2021



# Contents

<b>1</b>	<b>Installation</b>	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	Installing Cuda	8
1.2.5	Install PyTorch	9
1.2.6	Install OSCP	10
1.2.7	Installing OpenCV	10
1.2.8	Installing Other Dependencies	11
1.3	Create Shortcut	11
<b>2</b>	<b>Mouth Model System</b>	13
2.1	Stream	14
2.2	Stream and Record	15
2.3	Positioner	16
2.4	Settings	17
2.5	Test Speed	18
<b>3</b>	<b>Positioner Tool</b>	19
3.1	How to Position	19
3.2	Positioner Buttons and Projection Circle	21
<b>4</b>	<b>Changing Machine Learning Models</b>	23
	<b>Index</b>	25



# Installation

The detection program has been tested on a **Windows 10** running **Python 3.9.4**. 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

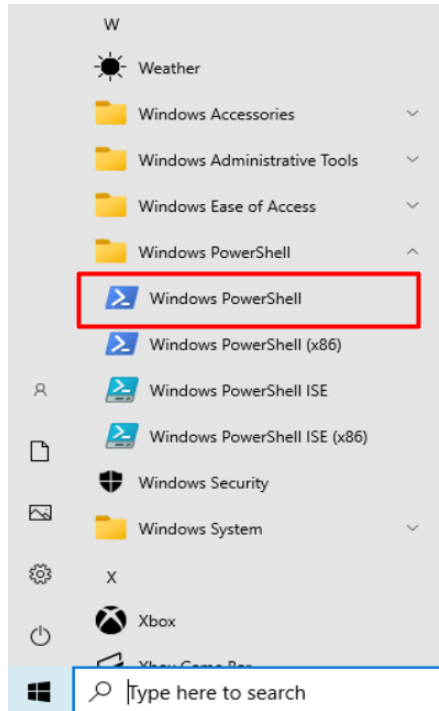
Though some older versions of the dependencies might work, the versions tested are as follows.

- 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

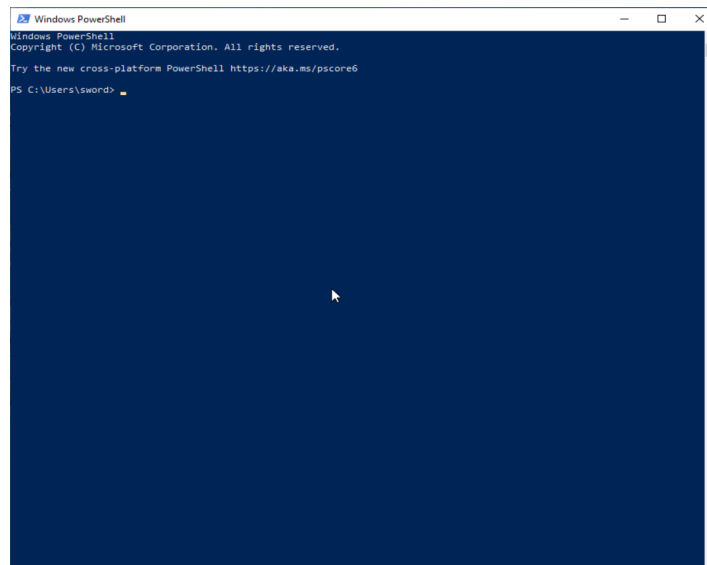
To install the various dependencies, it is easiest to use a terminal or command prompt.



**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.



**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 Installing Cuda

To offload the model onto the GPU and increase the speed of the system, I recommend you install Cuda. I have added Cuda functionality thru PyTorch to the system. Installing Cuda support is relatively straight forward. You do need an NVIDIA graphics card to run Cuda.

Go to: <https://developer.nvidia.com/cuda-toolkit> to download the toolkit. The page should look something like Fig. 1.3. Click the Download Now button to select the version.

You will want to select the version that matches your OS. In Fig. 1.4 I have show the selection for the Windows 10 OS using the Network exe. The Network exe means that the initial download is smaller and uses the network to pull the rest of the data once installation begins.

Once the program is finished downloading run the program to finish the installation (Default installation is sufficient). You will see a series of prompts, click through them as you would any other install. Once finished, GeForce Experience may run and ask you to create an account, you don't need to if you don't want to. This is not needed to utilize Cuda in our mouth detector system.

Once finished, Cuda is now installed and can work with our system to speed up the rate of detections.



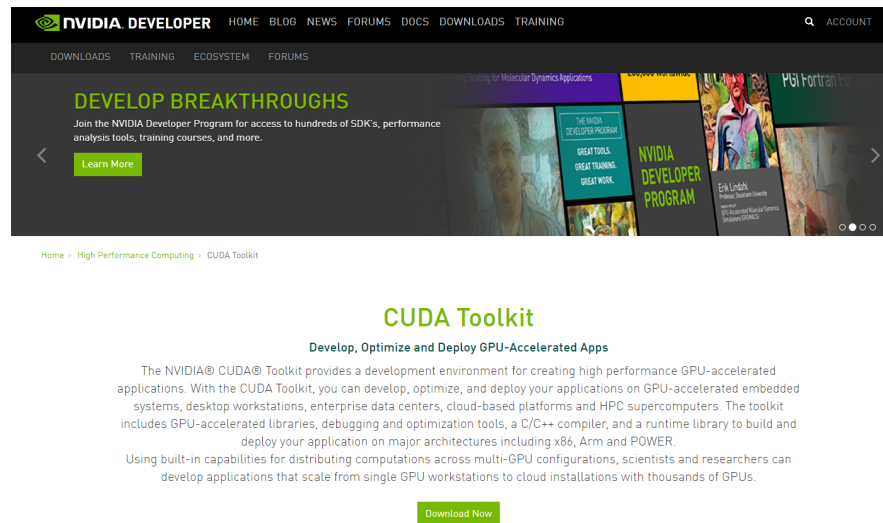


Fig. 1.3 NVIDIA Cuda Website

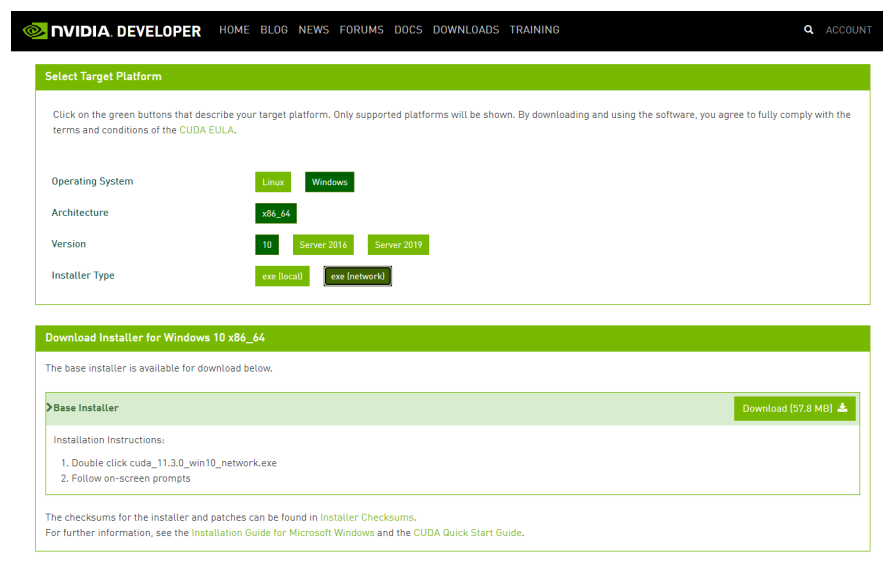


Fig. 1.4 NVIDIA Cuda Selection

## 1.2.5 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 instruc-

tions, 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
```

If you have installed Cuda, you will want to run one of the following instead. For Cuda 10.2, run:

```
C:\Users\user> pip install torch==1.8.1+cu102 torchvision==0.9.1+cu102
torchaudio==0.8.1 -f https://download.pytorch.org/whl/torch_stable.html
```

Or for Cuda 11.1, run:

```
C:\Users\user> pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111
torchaudio==0.8.1 -f https://download.pytorch.org/whl/torch_stable.html
```

**Important Note:** if you installed PyTorch and then installed Cuda, you will need to reinstall PyTorch with Cuda. To uninstall PyTorch use the following:

```
C:\Users\user> pip uninstall torch
C:\Users\user> pip uninstall torchvision
C:\Users\user> pip uninstall torchaudio
```

Then install PyTorch again using the prior commands based on your needs.

## 1.2.6 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
```

## 1.2.7 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
```

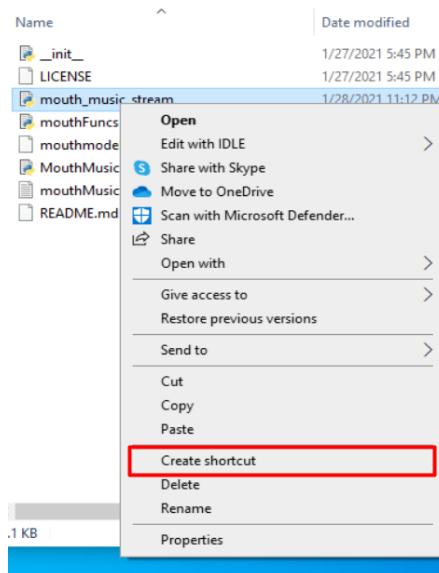
**Important:** you may encounter an error related to Visual C++ when running the program. To fix this, download and run the following program. [https://aka.ms/vs/16/release/vc\\_redist.x64.exe](https://aka.ms/vs/16/release/vc_redist.x64.exe)

### 1.2.8 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

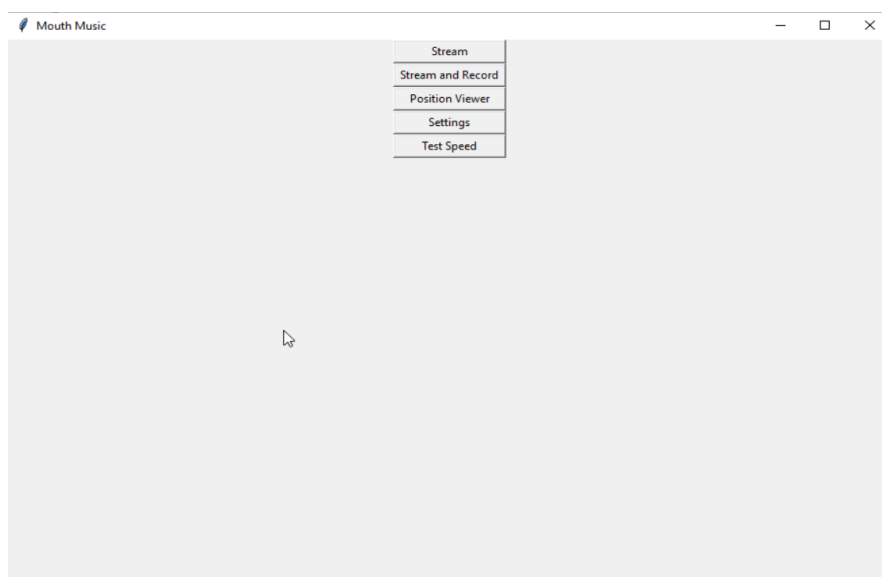
Creating a shortcut to run the program is relatively straight forward. In the repository directory, select the **mouth\_music\_stream.py** file using **Right Click**. Then select **Create shortcut** from the menu. This will create a shortcut object in the directory, **mouth\_music\_stream.py - Shortcut**.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



**Fig. 1.5** Create Shortcut



# Mouth Model System

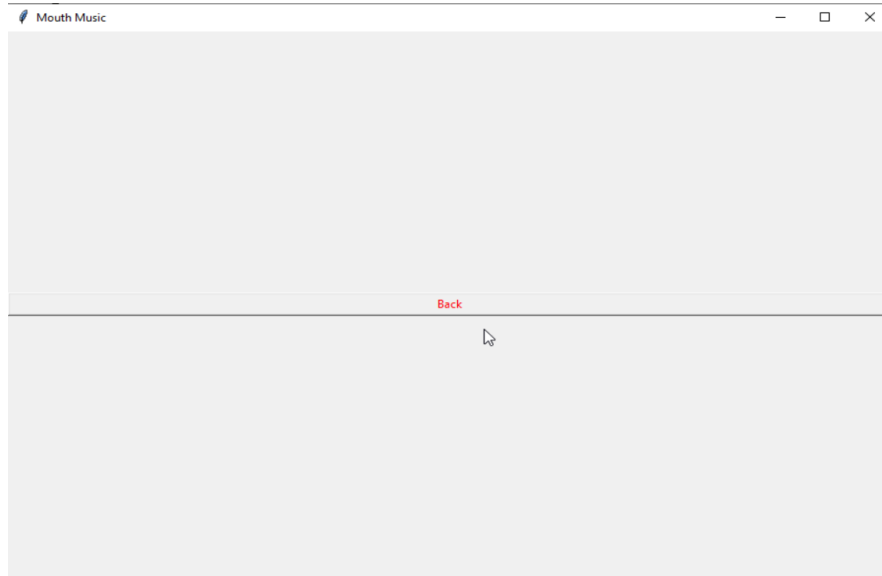


**Fig. 2.1** Mouth Music Home Screen

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

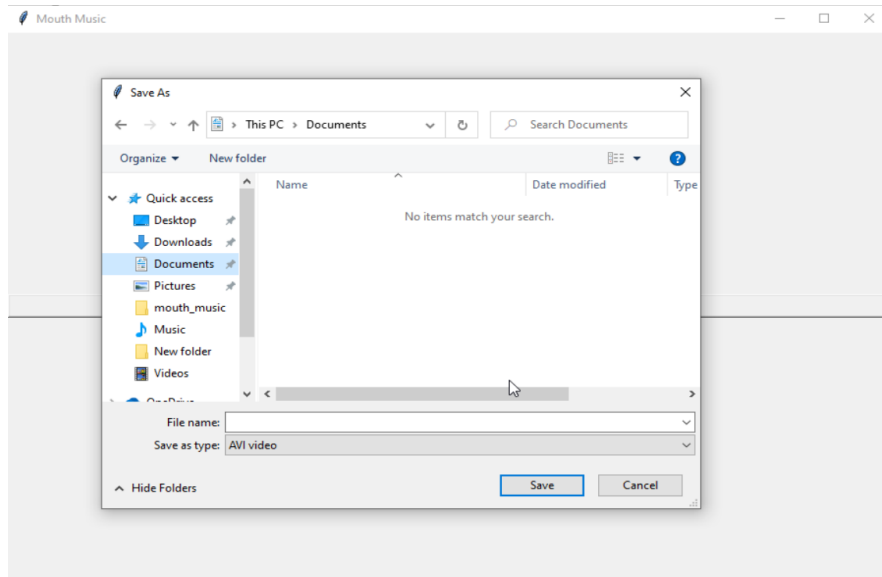
This option starts the streaming of position data through an OSC client. Streaming is done without visual so as to reduce the latency within the system. While streaming, only a **Back** button will be visible. This button will stop the OSC client stream and return to the home screen. The streaming screen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



**Fig. 2.2** Streaming Screen

## 2.2 Stream and Record

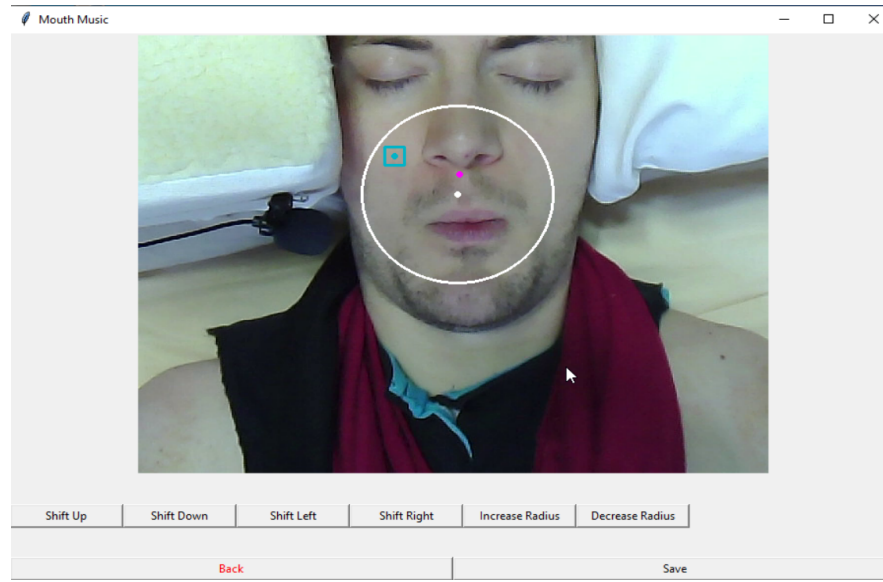
This option allows the user to select an .avi file to save the captured images to, Figure 2.3. After a name is selected, the system starts streaming position data through an OSC client and saving the images to .avi file. Though the images are saved, streaming is done without visual so as to reduce the 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.



**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.

IP	127.0.0.1
Port	6730
Vertical Topic	/tongue_gestures/vertical
Horizontal Topic	/tongue_gestures/horizontal
Mouth Intensity Topic	/tongue_gestures/mouth_intensity
Pucker Topic	/tongue_gestures/pucker
Tongue Out Topic	/tongue_gestures/tongue_out
Left Wink Topic	/tongue_gestures/left_eye
Right Wink Topic	/tongue_gestures/right_eye
Left Brow Topic	/tongue_gestures/brow
Eye Intensity Topic	tongue_gestures/eye_intensity
Number of Positions	100
Image Width(Pixels)	640
Image Height(Pixels)	480
Mouth Gesture Detection Threshold (0.0 to 1.0)	0.3
Tongue Detection Threshold (0.0 to 1.0)	0.2
Eye Gesture Detection Threshold (0.0 to 1.0)	0.2
Mouth Trigger Intensity Threshold (0 to 100)	15
Eye Trigger Intensity Threshold (0 to 100)	15

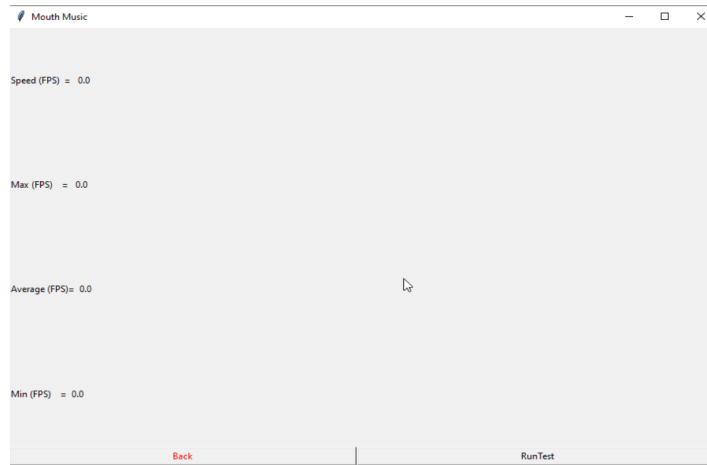
☐ Detection Box on Video

[Back](#) [Save](#)

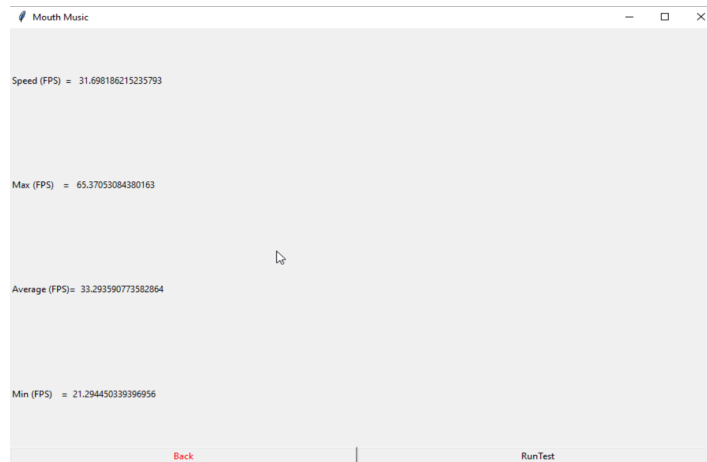
**Fig. 2.5** Settings Screen

## 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.



**Fig. 2.6** Speed Test Screen

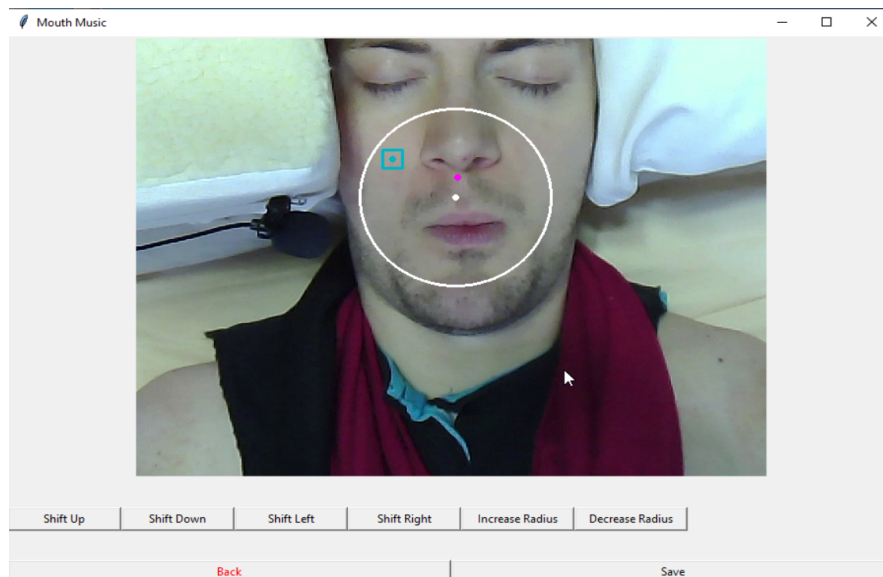


**Fig. 2.7** Speed Test Results Screen

# 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.

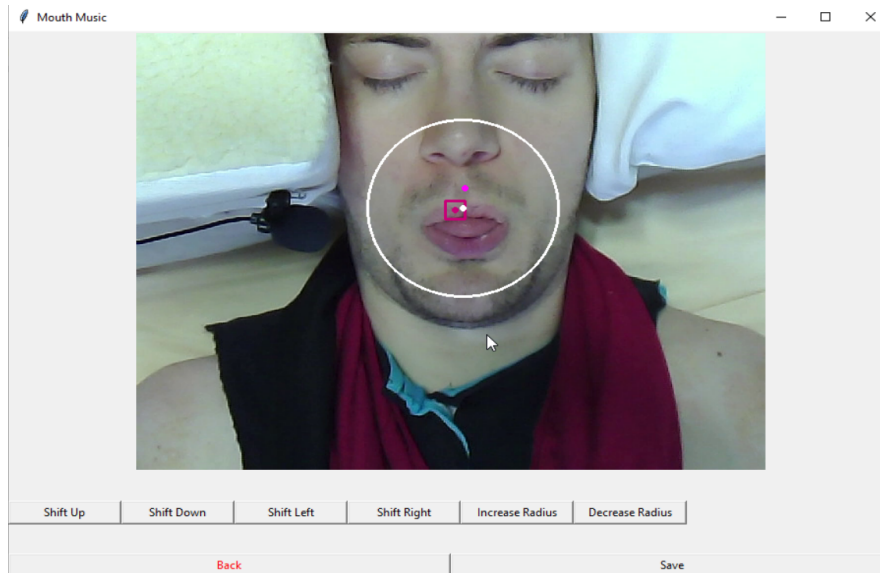


**Fig. 3.1** Positioner Tool

**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.

**Note:** Version 2.0 now uses two models to get both eye and mouth triggers. A single model can be used if desired, but appropriate changes in the code should be made to use a single model.

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. **mouthIntensity**—mouth trigger detection intensity
4. **mouthTrigger**—identified trigger from the mouth. Triggers implemented are: Pucker, Tongue Out, and In Cheek.
5. **mouthTriggerConf**—mouth trigger confidence value
6. **lipPosition**—tuple with the (x,y) position of the lip detection point
7. **tonguePosition**—tuple with the (x,y) position of the tongue detection when In Cheek is detected.
8. **leftEyeConf**—left eye confidence value
9. **rightEyeConf**—right eye confidence value

10. **leftBrowConf**—left brow confidence value
11. **leftEyePosition**—tuple with the (x,y) position of the left eye detection point.
12. **rightEyePosition**—tuple with the (x,y) position of the right eye detection point.
13. **leftBrowPosition**—tuple with the (x,y) position of the left brow detection point.
14. **leftBrowBoxWidth**—integer width of brow detection box
15. **leftBrowBoxHeight**—integer height of brow detection box
16. **eyeIntensity**—eye trigger detection intensity
17. **eyeTrigger**—identified trigger from the eye. Triggers implemented are: Left Wink, Right Wink, and Left Brow.
18. **eyeTriggerConf**—eye trigger confidence value

These are the minimal number of functions needing to be changed should the model be replaced. Examine the `mouthFuncs.py` document and the `MouthMusic-Model.py` document to see current implementations. Additional changes may be necessary to achieve desired behavior for other machine learning models.



# Index

## A

align 19  
anomalies 21

## B

Back button 14, 15  
best placement 19

## C

center 21  
change machine learning models 23  
circle 20  
commands 6  
Cupid's Bow 19

## D

decrease 21  
dependencies 5, 11  
Desktop 11  
detection box 17  
detection square - blue 20  
detection square - purple 20  
detection system 5, 18  
detections 19  
directory 11  
down 21  
download 5

## E

edge 21  
error - Visual C++ dependency 10

## F

face 21

## H

head - user's 20  
home screen 13  
horizontal 21

## I

increase 21  
input image size 17  
install 5  
ip address 17

## L

left 21  
lip detection 19  
lips 21

## M

maximum value 21  
mouth 21  
mouth model 23

## N

new model 23  
number of positions 17  
numpy 8

## O

offset 16  
OpenCV 10  
OSC 10, 21  
OSC - settings 17

OSC - position topics 17  
OSC Client 18  
OSC client 14, 15  
OSCPy 10

## P

person unique design 23  
Philtrum Ridges 19  
PIL 11  
pip 8  
placement 19  
placement conditions 19  
poor detection 20  
port 17  
positioner 16, 19, 21  
positioner buttons 21  
PowerShell 6  
program start 13  
projection circle 16, 21  
projection circle placement 21  
protrudes 20, 21  
Python 5, 8  
PyTorch 9

## R

radius 21  
record 15  
recorded video 17  
repositioning the camera 20  
repository 5  
requirements 19, 21  
right 21  
Run Test 18  
run the detector 11

## S

save 15, 17  
Save button 17  
save settings 17  
shortcut 11  
specifications 5  
speed 18  
square 21  
statistics 18  
stop 14  
streaming 14  
streaming screen 14

## T

terminal 6, 8, 11  
tkinter 11  
tongue 20  
tongue placement 21  
tongue tip 20  
top lip 19

## U

up 21

## V

versions - dependencies 5  
vertical 21  
visual 16

## W

webcam 16, 18, 19, 21  
Windows 10 6