## Computer Pointer Controller

This project, uses a gaze detection to control the mouse pointer on the computer. App uses the Gaze Estimation model to estimate the gaze of the user's eyes and change the mouse pointer position accordingly.

## How it works

The app uses InferenceEngine API from Intel's OpenVino ToolKit to build the **gaze estimation** model which requires three inputs:

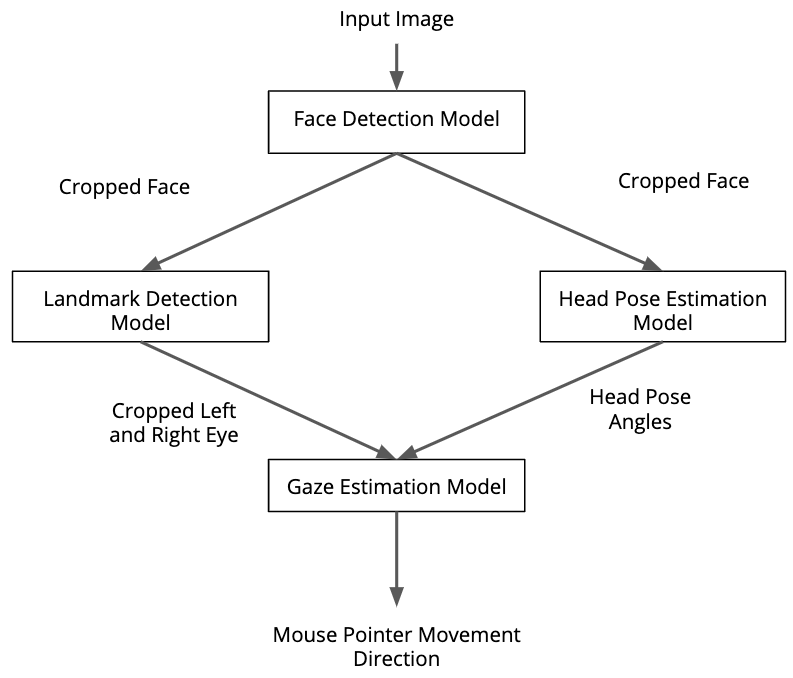
* The head pose
* The left eye image
* The right eye image

To get these inputs, three other OpenVino models are used:

* Face Detection
* Head Pose Estimation
* Facial Landmarks Detection.

## The Pipeline

The data frames flow from the input source (video or camera), and then inference of the frames happens through different models and finally the output which represents the direction of the gaze is fed to the mouse controller. The flow of data will look like this:



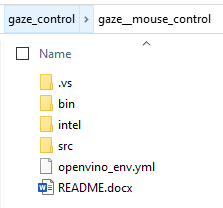
## Project Set Up and Installation

System info:

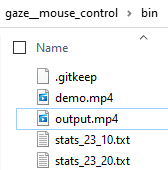
* MS Windows 10 Enterprise version Build 18363
* Intel Core i7-8665U

Project directory structure:

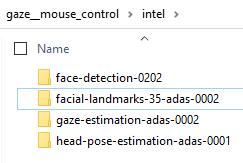
1. The project “gaze\_mouse\_control” directory structure is as follows:



* “bin” directory has the input demo video, output videos, and the stats files:



* “intel” directory has all the model files:



**Note:**

These models were downloaded using the model\_downloader application on Linux and then copied over to Windows.

Example cmd line on linux:

python3 /opt/intel/openvino\_2021/deployment\_tools/open\_model\_zoo/tools/downloader/downloader.py --name face-detection-0202

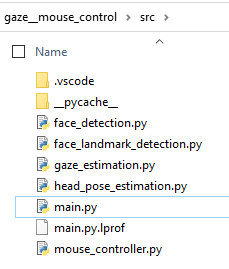
* “src” directory has all the source files:

The main.py file contains the main logic of the appicaltion.

There are separate files for each of the models (with the model’s name) that contains the code for abstracting the class and methods are each model eg: load\_model(), predict() etc.

**Note:**

* input\_feeder.py is unused.
* main.py.lprof is the output of lineprofiler that can be ignored.



## Installation steps:

1. Download and install Anaconda on Windows 10- <https://www.anaconda.com/products/individual>.
2. Open Anaconda command prompt and type the following to create and activate the virtual conda environment:

> cd gaze\_mouse\_control

> conda env create -f openvino\_env.yml

> conda activate openvino\_env

## Running the application:

1. For help with Running the application:

>cd src

>python main.py --help

**Output**:

usage: main.py [-h] [--device DEVICE] [--video\_source VIDEO\_SOURCE]

[--output\_path OUTPUT\_PATH] [--precision PRECISION]

[--show\_output SHOW\_OUTPUT]

optional arguments:

-h, --help show this help message and exit

--device DEVICE Select the inference device (eg: CPU, GPU,

MULTI:CPU,GPU) for all 4 models. Default is CPU.

--video\_source VIDEO\_SOURCE

Path to the video source. For camera source use "CAM".

Default is video file at "../bin/demo.mp4".

--output\_path OUTPUT\_PATH

Set the path to store the output video and stats file.

Default is "../bin/".

--precision PRECISION

Select the Precision (eg: FP32, FP16, FP16-INT8) of

the inference models. Default is FP32.

--show\_output SHOW\_OUTPUT

Set '1' to show the annotations on the output. By

default annotations are disbaled.

1. Run the application in default mode:

>cd src

>python main.py

In this mode, demo.mp4 will be the input video source. Precision for all models will be FP32. Inference device will be CPU. No output will be displayed.

**Note:**

To end the application hit cltr+c on the keyboard.

**Output:**

Mouse pointer on screen will follow the gaze of the instructor in demo.mp4.

Output video will be saved in “bin”. Stats will also be saved under “bin” with time stamp:

eg:

stats\_hh\_mm.txt

precision:FP32

device:CPU

---------------

Model Load Time (secs)

---------------

face\_detection: 0.21

face\_landmark: 0.25

head\_pose: 0.07

gaze\_estimation: 0.07

total\_load\_time: 0.6

---------------

Inference Time (msecs)

---------------

face\_detection: 19.165000000000003

face\_landmark: 3.3200000000000003

head\_pose: 2.1683333333333334

gaze\_estimation: 1.8233333333333333

---------------

FPS:0.83

---------------

mc\_update\_time (msecs): 1161.34

1. Use camera as input, show output annotations, device=GPU and precision FP16:

>cd src

>python main.py --video CAM --show\_output 1 --device GPU --precision FP16

**Note:**

To end the application hit Esc on the keyboard.

eg: Stats\_hh\_mm.txt:

precision:FP16

device:GPU

---------------

Model Load Time (secs)

---------------

face\_detection: 21.22

face\_landmark: 61.16

head\_pose: 7.37

gaze\_estimation: 7.06

total\_load\_time: 96.81

---------------

Inference Time (msecs)

---------------

face\_detection: 10.191728395061736

face\_landmark: 5.366296296296296

head\_pose: 2.096172839506173

gaze\_estimation: 1.9069135802469137

---------------

FPS:0.84

---------------

mc\_update\_time (msecs): 1160.92

**Benchmarking and Observations:**

1. The bottleneck

The Throughput/FPS of the pipeline suffers badly due to the mouse controller API in pyautogui i.e pyautogui.moveRel()takes about 1 sec. This is the biggest bottle neck in the application!

Running after commenting this API the FPS is 26.

Eg:

>cd src

>python main.py --video CAM --show\_output 1 --device GPU --precision FP16

precision:FP16

device:GPU

---------------

Model Load Time (secs)

---------------

face\_detection: 21.28

face\_landmark: 67.78

head\_pose: 8.52

gaze\_estimation: 8.18

total\_load\_time: 105.76

---------------

Inference Time (msecs)

---------------

face\_detection: 11.26759036144579

face\_landmark: 6.679999999999997

head\_pose: 2.264658634538152

gaze\_estimation: 2.1151405622489943

---------------

FPS:26.49

---------------

mc\_update\_time (msecs): 0.0

1. Comparing CPU v/s GPU inference in FP16

>cd src

>python main.py --video CAM --show\_output 1 --device CPU --precision FP16

precision:FP16

device:CPU

---------------

Model Load Time (secs)

---------------

face\_detection: 0.3

face\_landmark: 0.34

head\_pose: 0.08

gaze\_estimation: 0.09

total\_load\_time: 0.81

---------------

Inference Time (msecs)

---------------

face\_detection: 25.06295454545453

face\_landmark: 4.213484848484849

head\_pose: 2.538257575757576

gaze\_estimation: 2.677651515151515

---------------

FPS:19.7

---------------

mc\_update\_time (msecs): 0.0

* Using GPU as the inference device improves the **FPS from 19 to 26.**
* However, using GPU the model load time is very high.

Note, however, according to Intel documentation this problem can be solved for subsequent runs by enabling caching of the OpenCL kernels.

1. Edge Cases

Initially in the application I used only p12 and p15 coordinates from face landmark detection results for left eye and right eye corners, and then used fixed lengths to crop a square ROI around the eyes. However, this method failed to scale especially when the face was away from the camera. Hence, switched to using p14 and p17 as well to use four coordinates around both eyes and this produced better results.

It was also noted that the application does poorly under low light conditions because the face detection fails.