Missouri State University

Department computer Science

Smart Mouse Interface

Jacob Kuhnert
Christain Leslie
Jabez Nuetey
Tylor Phillippe
Trevor Cameron

Client Info:
Mohammed Belkhouche
YassineBelkhouche@missouristate.edu
(417)-836-5789

December 8, 2022

Contents

1 Project description

1.1 Smart Mouse Interface

Smart Mouse Interface is a project in which we use deep learning so the program can recognize human gestures. We assign gestures commands to replace the mouse universally. The user will provide gestures via live video on camera as input, and the software will perform the specified mouse actions. To track hand movements and recognize the gesture we use OpenCV and MediaPipe, to teach the program which gestures will be used for actions, and the corresponding action, TensorFlow is used. The actions Smart Mouse Interface is able to perform are: Scroll Up/Down/Left/Right, Zoom In/Out, Left/Right Click, and Control Cursor.

2 Requirements Specification

Functional Requirements:

1. ID: Data Collection
   Description: Collect x, y, and z coordinates of each gesture, and store them in a csv file
   Rational: To train our deep learning model
   Input(s): x, y, and z coordinates
   Outputs(s): Data csv
   Dependency: None

2. ID: Train Model
   Description: Use the collected data to train our deep learning model.
   Rational: So our program can recognize designated gestures.
   Input(s): Data csv
   Outputs(s): Trained model
   Dependency: Data csv

3. ID: Live video access
   Description: Access camera to have a live video feed.
   Rational: To track hands for gesture recognition.
   Input(s): Live Video
   Outputs(s): Video Frames
   Dependency: Camera

4. ID: Calibrate camera
   Description: Create a calibration system for the camera.
   Rational: Accurately track hands in any environment.
   Input(s): Live Video
   Outputs(s): Calibration Parameters
   Dependency: Camera

5. ID: Track hand movements
   Description: Use openCV and mediapipe to track hand movements from the live video feed.
   Rational: Need to recognize gestures from the live video feed.
   Input(s): Video Frames
   Outputs(s): Hand Gestures
   Dependency: Camera

6. ID: Assign actions
   Description: Assign actions to specified gestures using the recorded landmark coordinates.
   Rational: Actions need to be preformed when specific gestures are recognized.
   Input(s): Gestures, Video Frames
   Outputs(s): Actions specified in description
   Dependency: Camera, hand tracking program, trained model

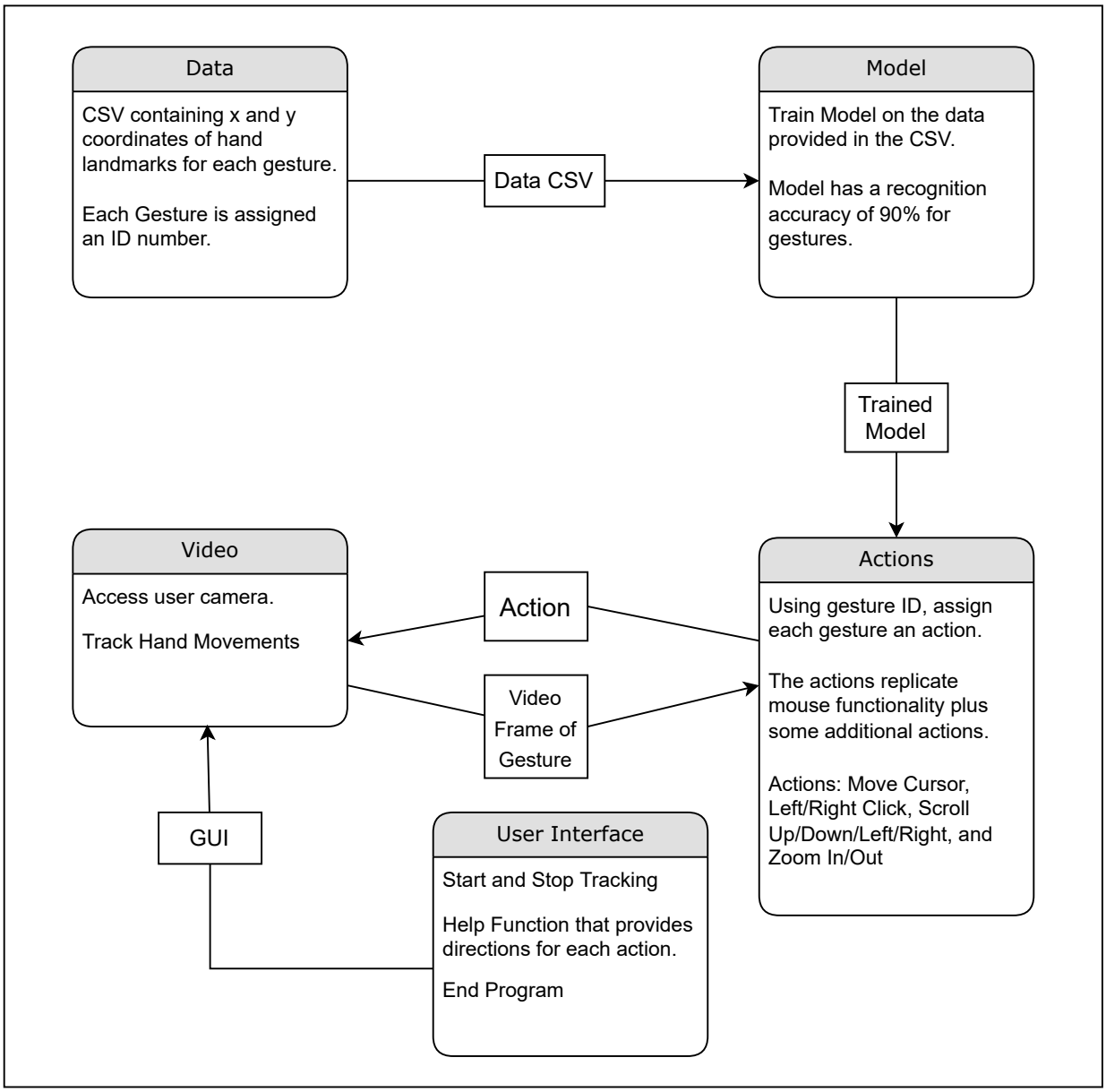
Non-Functional Requirements:

1. Software should have a max delay of 10 millisecond, so the user does not experience lag.
2. Software should recognize gestures at a minimum of 90 percent accuracy, assuming the user is properly using the program.
3. Software should only preform specified actions.

3 Schedule

Task	Member(s)	Start	End	Depenecies
1. Documentation	Jacob	1-Sep	12-Dec	none
2. Design Interface	Christian	14-Oct	21-Oct	none
3. Collect Coordinate Data	Jacob	10-Oct	24-Oct	none
4. Collect Photo Data	Jacob	10-Oct	24-Oct	none
5. Train Deep Learning Model	Tylor	25-Oct	31-Oct	Tasks 3, 4
6. Code/Assign Actions	Jabez, Tylor	1-Nov	28-Nov	Task 5
7. Implement Hand Tracker	Trevor	10-Oct	20-Oct	none
8. Implement Interface	Trevor, Christian	22-Oct	14-Nov	Task 2
9. Optimize Code	Jabez, Trevor	29-Nov	10-Dec	Tasks 6, 7, 8
10. Prep Presentation	Jacob	9-Dec	12-Dec	All

4 Software Design



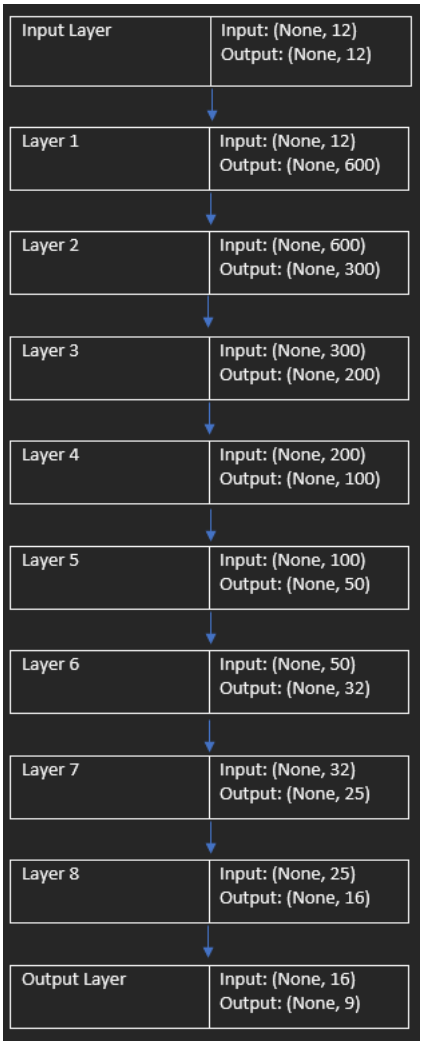
5 Implementation and Testing

5.1 General

This project is developed in python and uses several different libraries to replicate mouse actions by just using hand gestures. The libraries we use are pillow (PIL), mediapipe, pyautogui, cv2, tkinter, ttkthemes, and tensorflow. The most important library used is MediaPipe, which was developed by google. It allows us to track the hands using 21 different landmarks, and return coordinate data for each landmark. Full documentation is available at: https://google.github.io/mediapipe/solutions/hands.html.

5.2 Data Collection and Training the Model

To train our model we collected MediaPipe normalized coordinates for each gesture. To do this we created a program that reads video, then applies MediaPipe's coordinates system to each video frame, and saves that data into a CSV file. We collected data on 2650 instances of each gesture, consisting of both hands, different distances, and different angles. Tensorflow, more specifically keras, is used to train our model. Our model has ten layers, with each layer reducing the number of nodes. 1000 epochs were used when training in order to maximize accuracy.



5.3 Using the Model

To use our trained model we import it into our main file. We then are able to use the built in predict function, and feed it coordinate data from the video frames. This is done by storing coordinates in an array and calling the predict function on the array. The result of predict function will be an ID number, of which each action has been assigned. That ID number is then read and calls the corresponding action. The actions are preformed by using pyautogui functions. To minimize the risk of preforming an unwanted action, we created an idle gesture, which is just an open palm.

5.4 Hand Tracking and UI

Upon running the program, the UI will appear. This contains a view of yourself and three buttons: toggle tracking, help, and quit. The toggle tracking button allows the user to start and stop the hand tracking as needed. The help button opens a new window that will provide details and examples of each action that can be preformed. The quit button terminates the program. Using cv2 we are able to capture live video and format it into a usable form. Then using MediaPipe we track the hand, and using pyautogui the mouse cursor tracks with the wrist landmark. The wrist landmark was selected for the cursor so that when preforming a gesture, the cursor will not move unexpectedly.

6 Resources

Tools and Resources:

- Camera
- Python
- Anaconda
- Deep Learning
- OpenCV
- MediaPipe
- Tensorflow
- Overleaf
- Pyautogui

7 References

Gesture recognition has many uses, several of which have already been implemented. Many of the projects use python as the chosen language but there are a few which use C++. A constant between all the projects is the use of OpenCV and TensorFlow to track and map the gestures using deep learning. Another resource some projects utilize is MediaPipe which is a machine learning template developed by Google. The purpose of these projects vary from controlling robots to playing games to drawing.

https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/
https://becominghuman.ai/deep-learning-hand-gesture-recognition-b265f4e6cf02
https://github.com/topics/hand-gesture-recognition?l=pythono=descs=updated
https://www.pantechsolutions.net/top-10-gesture-recognition-projects