Universidad Pontificia Comillas

ETSI-ICAI

Research documentation

AIRTRACKPAD

Lydia Ruiz Martínez, Pablo Tuñón Laguna

December 2024

Index of contents

1. Introduction

Human interaction with computers has evolved significantly, seeking more intuitive and polyvalent interfaces. AirTrackPad is part of this trend, allowing devices to be controlled through hand gestures detected in the air, without the need for additional hardware such as a Kinect or a Leap Motion. It has been developed to be used either on a capable computer or a Raspberry Pi, making it a versatile tool for different applications. At the end of this document, there is also a part that explains how to modify and re-train the model with different gestures, making it more adaptable to different needs.

2. Methodology

The project has been structured in 7 directories, 2 main files and 1 utils file that helps coordinating some functions between the different modules. That structure can be seen in the following list:

- **3d files**: two specific accessories designed and 3D printed to improve the user experience while using an external camera.
- **actions_handler**: responsible for managing the actions derived from the classified gestures.
- **camera_calibration**: the environment for the camera calibration process.
- **hand_tracking**: implements landmark detection with `Mediapipe` with the verification of Sobel.
- **movement_classifier**: contains models and scripts for gesture classification through machine learning techniques.
- **movement_follower**: is the responsible of continuing the movement of the landmarks between frames.
- **AirTrackPad.py&AirTrackPadLinux**: main scripts that integrate and coordinates all modules of the system to create the final product.
- **utils.py**: auxiliary functions that some modules repeat its use.

2.1. Camera Calibration

The camera calibration process is essential to ensure the accuracy of the hand tracking system. It is carried out by capturing images of a chessboard pattern from different angles and distances, allowing the estimation of the intrinsic and extrinsic

parameters of the camera. The calibration module provides a graphical interface to facilitate the process, generating the necessary files for the subsequent configuration of the tracking system. The cameras used for this project are a Raspberry Pi Camera Module 3 and the webcam of a laptop. It is important to note that the calibration process is independent for each camera. The intrinsic parameters of the external camera are:

$$K = \begin{pmatrix} 346{,}97 & 0{,}0 & 226{,}29 \\ 0{,}0 & 347{,}78 & 129{,}31 \\ 0{,}0 & 0{,}0 & 1{,}0 \end{pmatrix} \tag{2.1}$$

2.2. Hand Detection and Tracking

At the beggining it was used a Canny filter over a MOG2 mask to detect the hand, however it was discarded due to the low performance of the system and due to the high computational cost. The final solution was to use the `Mediapipe` library, which provides a robust identification of 21 key points (*landmarks*) in the hand:

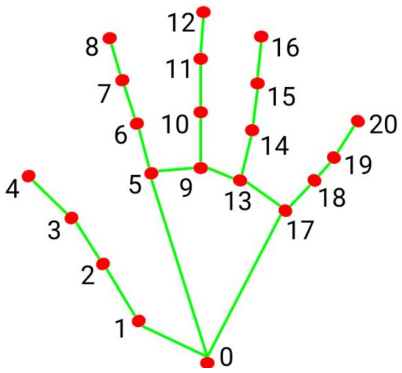

Figura 1: Hand landmarks detected by Mediapipe, ResearchGate

The landmarks detection of Mediapipe where complemented with a sobel filter over the ROI of the hand. This allowed to extract an accuracy of the model, changing the detection of the landmarks when the landmarks were not detected correctly.

2.3. Gesture completion

As it has been mentioned, thet Mediapipe landmarks are considerabely accurate, however for the times where the Sobel filter detects that the performance is not optimal, or in case the do disappear due to a phisical problem (like a hand occlusion), the system continues the movement of the previous landmarks using the Lucas-Kanade optical flow algorithm. It is applied to the ROI of the detected points, which means 21 different optical flows. This may sound as a highly computacional algorithm, however Sobel and Lucas-Kanade have an option for devices with less computational power that reduce the padding of the ROIs and the number of iterations of the algorithm.

2.4. Gesture Classification

The classification of gestures was originally carried by a logistic regression model, however it wasn't very accurate. For this reason, a neural network was developed with a hidden layer of 10 neurons and ReLU activation function for the classification of gestures. The Raspberry Pi is a device that needed a lightweight architecture therefore after some tests, 10 neurons was identified as a model that allowed real time use of the device with an acceptable degree of classification success.

2.5. Action Handling

Classified the actions, the system needed to convert them into actions that the computer could understand. To tackle this problem, the library `pyautogui` was used that allowed to define actions such as scrolling, clicking or moving the cursor.

3. Results

A video has been uploaded to the contents directory that shows the system working. The AirTrackPad is capable of working in real time, with a frame rate of 30 fps without never loosing the hand and with a classification accuracy of around 90 %.

4. Future developments

Although the system is already functional and identifies correctly many gestures, it doesn't detect at the perfection the 10 movements. Scrolling and 2 hand inputs are the gestures that have a lower accuracy. However this might be solved through a more extensive training. Another improvement that could be done is to add a new layer to the neural network, which would allow to increase the accuracy of the model. Some tests where performed on this direction and although at the beggining the raspberry wasn't capable of running the model fluently, reducing the resolution of the camera gave some promising results. Finally the system could be improved by adding a user interface that allows more interaction and adaptability to the user.

5. Trainment of the model

As it has been mentioned, the model can be retrained with different gestures and more accurate datasets, to perform this, some modifications should be performed on the `movement_classifier` directory, more specifically on the `train_model.py` file. The user should modify the gestures and number of movements, it should also modify the line 109 of the *ClassifierTrainer.py* to 14 x the number of gestures to use

(number of features). Finally it should modify the number of features and gestures on the *AirTrackPad.py* file.

The training process is very intuitive as there is a graphical interface that informs the user of the movement that is being trained as well as the number of pictures for the identification. It is important that the user tries to make as many movements as possible to ensure the accuracy of the model. Another interesting feature of the training process is that the training process is not realized with a set of images and points but with a set of movements with context that allows the model to learn the movement in different situations.