UNIVERSITY OF MICHIGAN

Hands-free Typing and Text-to-Speech using Eye-Tracking

Haoxiang Li, Pengyang Wu, Simon Cadavid

Dec 3, 2024

Contents

Project Description

Introduction

- Our project allows people with motor disabilities (of the hands and/or face) to communicate by typing visually
- Generally, uses:
    - Image processing
    - Computer vision
    - Machine learning (e.g., clustering)
- Implementable on cheap hardware
    - Raspberry Pi 4
    - Picamera (vol. 2)
    - Mini-speaker

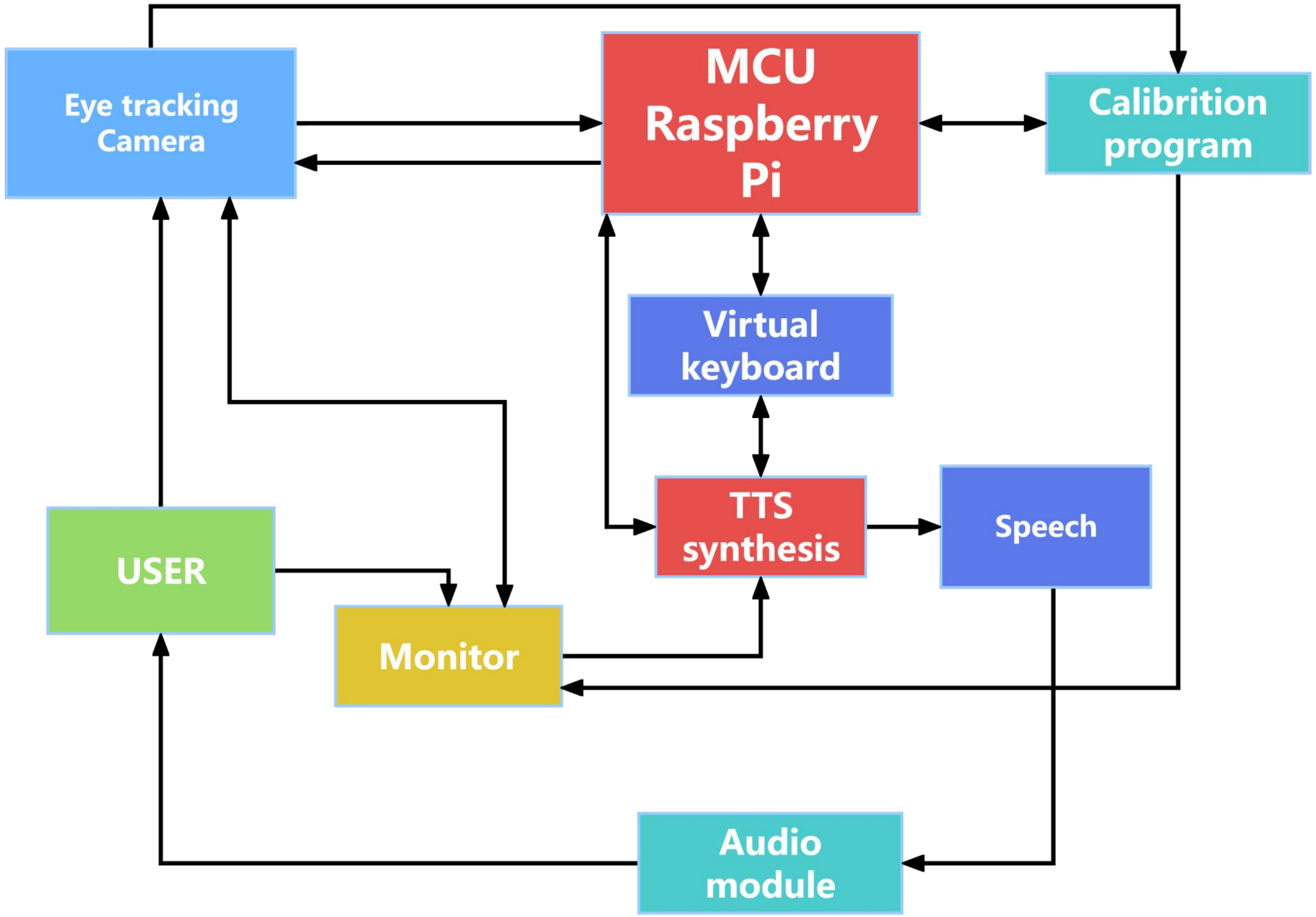
Motivation

- People with hand disabilities or more general movement disorders have difficulty typing
  - e.g., Parkinson's, ALS
- Our project allows them to type with their eyes
- Some people also can't speak due to facial muscle disorders
- Our project includes a Text-to-Speech (TTS) component

Figure:
Stephen Hawking with his IBM machine

Gaze Tracking + TTS System Setup

System Diagram



Demo

Implementation & Results

Eye Tracking Architecture

Initial Version

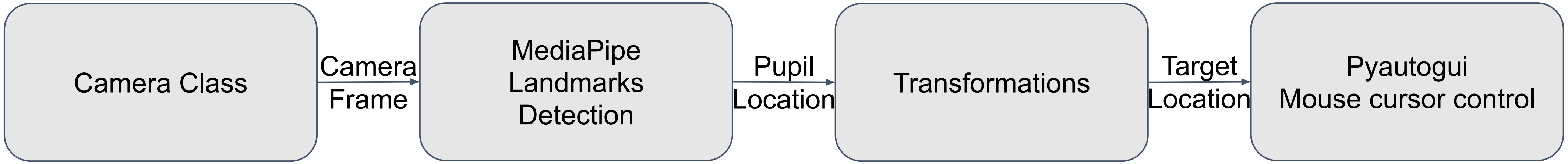
- Camera frame to Facial Landmarks (using dlib model)
- Isolate eyes from frames and locate pupil (using openCV contour)

- Pupil Location (X,Y) to Target on Screen (X,Y)

- Could be integrated with applications easily

Eye Tracking Architecture

Improved Version



- Produce camera frame
- Help make it smoother to debug on PC and deploy on PI due to PC and PI using different camera class

- Camera frame to Facial Landmarks
- More stable and faster than dlib solution
- Directly produce pupil locations

- Code refactor to enable easy switch of transformation function by just choosing the desired transform class
- Added perspective transformation (choice in final version)
- **EMA** filter

- Could be integrated with applications pretty easily

Calibration

Transformations from pupil location to screen target location needs calibration

Display Calibration Points and record pupil locations → Data Cleaning → Calibrate Transformation Parameters

Display Calibration Points and record pupil locations

- 3 Points for affine transformation
- 4 points for perspective transformation
- Maybe multiple points for neural networks

Data Cleaning

- Use clustering-like method to filter outliers
- Assume gaussian distribution
- In each iteration, calculate the means and variation, filter-out points with low possibility
- Redo the iterations until no changes

Calibrate Transformation Parameters

- Use the data to calculate transformation parameters
- calculate the transformation matrix in the case for affine and perspective transformation

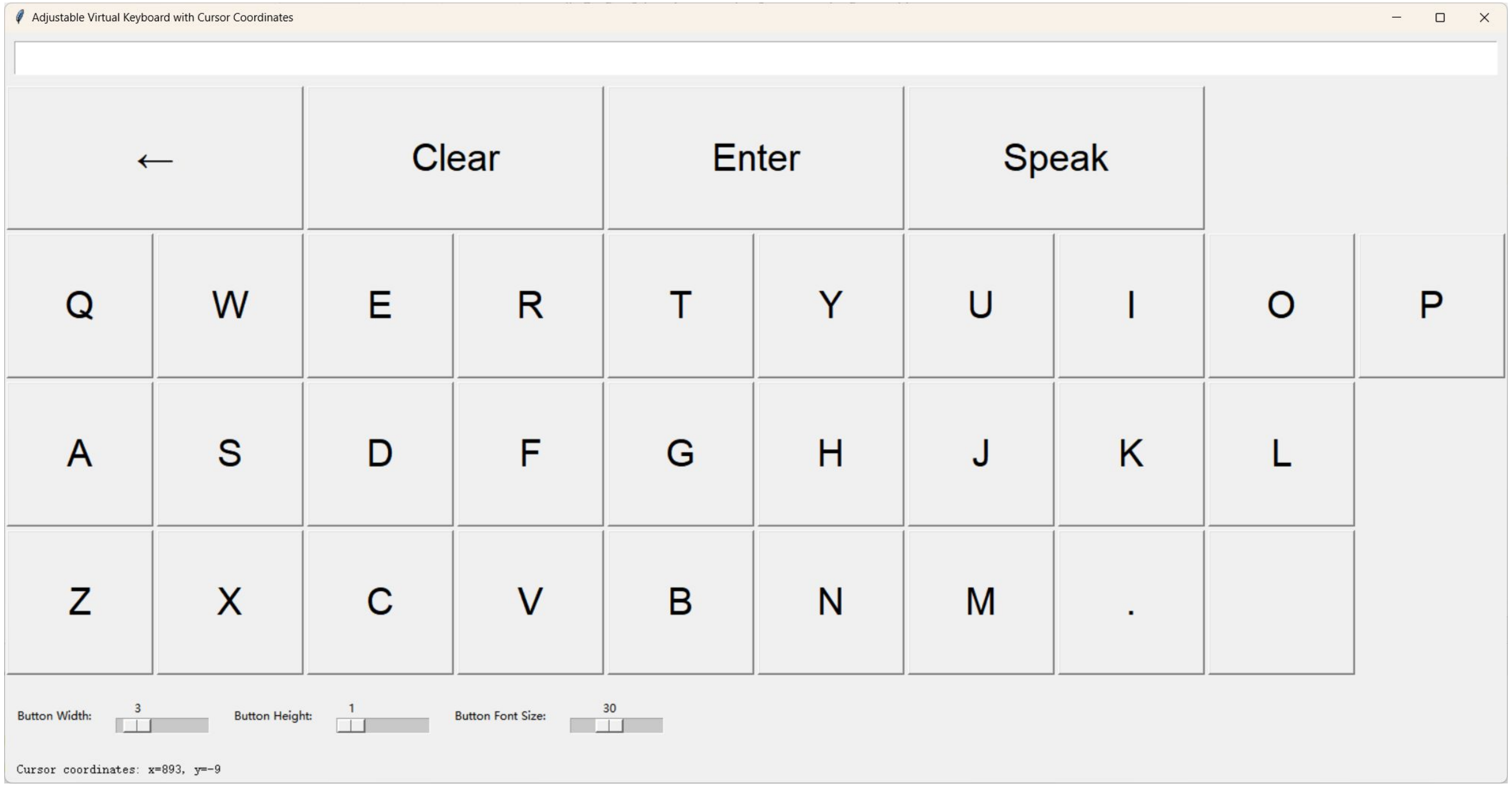
Attempts & Future Discussion

- Attempted to use machine learning methods for the transformation
- Pupil locations w/ other landmarks as input
- Doesn’t work at all
- Hard to obtain training data set
- Networks like DNN, Transformers might not be suitable for such task

- **Future Discussion:**
- Near-infrared camera
- High-resolution & high-frame-rate camera
- End to end machine learning method (from image frame to target location)
- Or non-linear transformation function

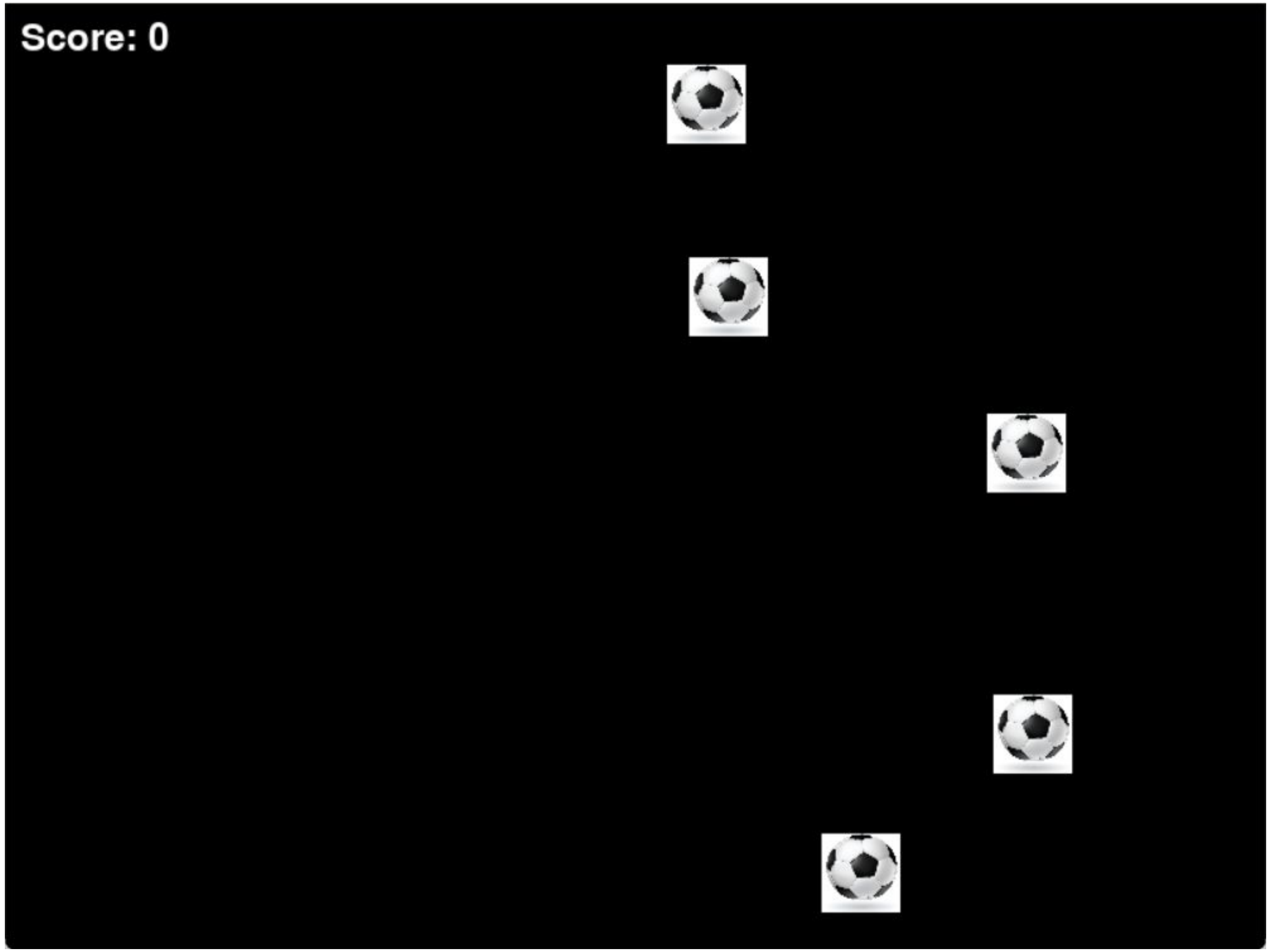
Virtual keyboard

1. Library: tkinter(small amount and fast enough)
2. The updated UI with simpler button layout and hover detection mechanism



Small game to test its accuracy

Library: pygame, numpy, sys



Brief technical highlights

1. A fully functional on-screen keyboard with keypress support and an editable text field.
2. Adjustable button sizes (width, height, font size) via sliders for customization.
3. A Text-to-Speech (TTS) feature using pyttsx3, which reads out the entered text aloud.
4. An interactive mouse tracking system to capture and display cursor coordinates in real-time.
5. Enhanced flexibility in the layout design and user experience:

   Wider first-row for important buttons (['←', 'Clear', 'Enter', 'Speak']).

   All other rows retain a consistent appearance.

   A linear(EMA) filter for the mouse movement to make better adaptation to gaze tracking.

Comparison with Commercial Product

Tobii Eye Tracker 5

- State-of-the-art
- Robust to head/face movement
- “Remote” (i.e., non-contact)
- $300 on Amazon

Our System

- Functional with some noise
- “Head-stabilized”
- Under $100

Other systems include non-remote (e.g., headsets, glasses), most more expensive than our system.

Tobii Eye Tracker Setup

Hands-free Typing & Text-to-Speech using Eye-Tracking

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY OF MICHIGAN

UNIVERSITY OF MICHIGAN

EECS 452: Digital Signal Processing Design Lab – Fall 2024

Simon Cadavid, Pengyang Wu, Haoxiang Li

Introduction

Many disabilities (e.g., facial or other muscle paralysis) affect people's ability to use electronic devices, specifically typing. Furthermore, those with facial paralysis often can't speak normally. By using eye-tracking to map eye movement onto a virtual keyboard, users will be able to type out words and sentences, hands-free, and then have it play back via text-to-speech.

Techniques

Pupil Detection: First, we use an open-source Python library called mediapipe [1] to do face landmark detection. These "facial landmarks" are pictured in Figure 2. They are produced by a pre-trained end-to-end neural network. We then select the facial landmarks corresponding to the pupil centers, i.e., two (x,y) pairs of coordinates in the camera image.
Calibration: We use a four-step calibration process that learns a perspective transformation to map the pupil centers to a cursor on the virtual keyboard (Figure 4).
Cursor Output: The perspective transform is applied to the camera feed to map the person's pupil positions to the virtual keyboard in real-time.
Text-to-Speech: Implemented with open-source Python library 'pyttsx3'.

System Setup

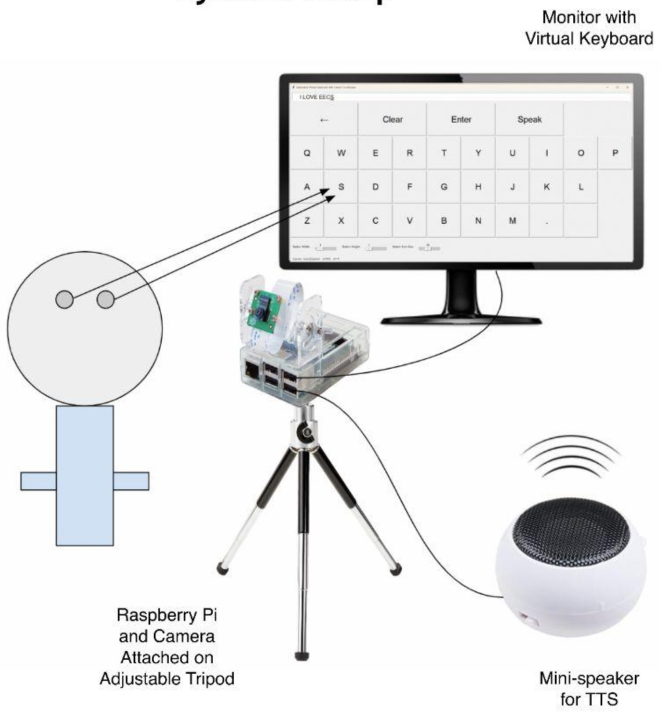
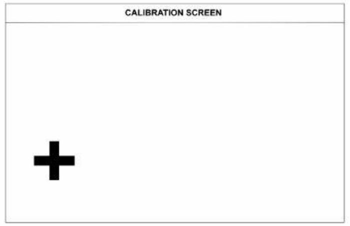


Fig 1: Final System Setup

Fig 2: Facial Landmarks from Mediapipe

Fig 3: Pupil Positions

Fig 4: Step 1/4 of Calibration



Results

With practice, it takes about 1 minute to type and speak "HELLO WORLD" (hence, 11 CPM). The cursor noise, although we significantly reduced it through filtering, is still present, which increases the typing time.

Conclusion / Future Improvements

Our system provides a relatively cheap (<$100) functional eye tracker, assuming no head or face movement, with some noise. Future improvements include using near-infrared light for better eye measurements, and a higher resolution / framerate camera for less noisy estimates. Also, more advanced (e.g., deep neural network / transformer) models for pupil -> cursor mapping.

Acknowledgements

We express our deep gratitude to Professor Jiasi Chen and GSI Cary Shu. We also thank the EECS Department for providing funding for our project.

System Architecture