LocIT Milestone #1 Report by Anindita Deb

08/18/2024 - 09/20/24

Overview

I oversee the facial recognition software implemented in LocIT. Using various machine learning models, I aim to create a system capable of recognizing individuals and addressing multiple functionalities, including:

- Recognizing familiar faces
- Flagging unknown faces during significant interactions
    - Identifying conditions that trigger a significant interaction
- Flagging certain facial expressions
    - Angered/upsetting expressions could imply that the LocIT user is in a potentially unsafe situation
    - Tracking negative expressions is a key feature being developed to alert the user to potential risks, improving situational awareness and safety
- Addressing additional issues like sleep/awakening conditions
    - Limiting battery strain
    - Handling privacy concerns

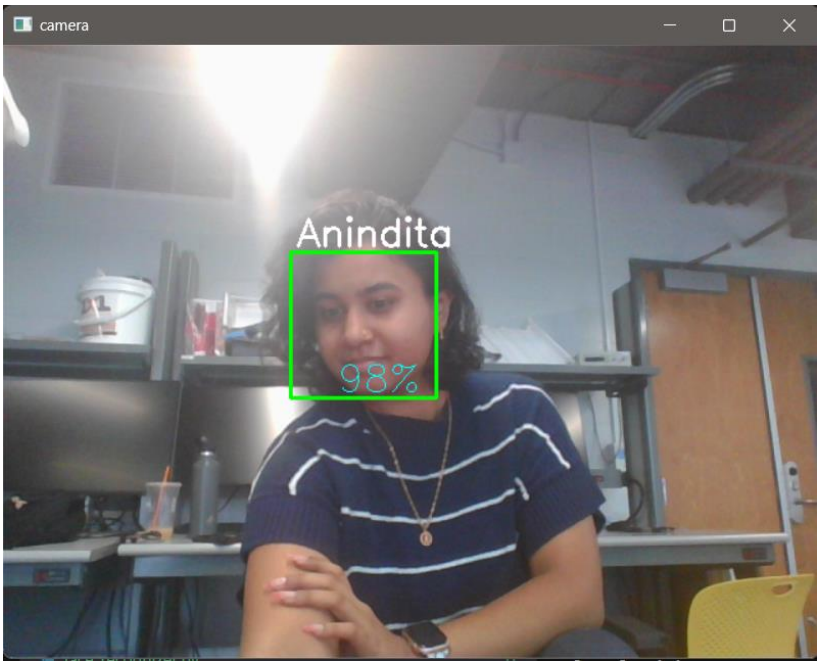
Facial Recognition Development

I've updated the code for this portion of the project to the GitHub Project page (https://github.com/dita-deb/LocIT/tree/main/Camera%20Module/Facial_Recognition_PC)

I wanted to eliminate the stress of collecting data through other means and inserting it into the model for training for the user. There's a built-in file that will take snapshots of the user for 30 seconds (1 frame per second) to train the model to recognize the user.

Here is a break down of each individual file and its application to the overall model:
Face_taker.py: Captures images of the target individual and stores them in the dataset for later use in training the model.
Face_train.py: Trains the facial recognition model using the images captured in Face_taker.py, allowing the system to recognize the individual's face.
face_recognizer.py: Runs the trained facial recognition model, identifying individuals from the dataset in real-time or from new images.
names.json: Stores a mapping between each individual's name and their corresponding image data in the dataset.
haarcascade_frontalface_default.xml: A pre-trained Haar Cascade classifier that detects facial features (e.g., eyes, nose, mouth) to identify and classify faces for recognition.

The way that it displays a recognized face is by drawing a green box around the face and having the person labeled with a certain amount of confidence as shown below:



These just screen shots of a live video feed; I have a converter (video to image parser) not yet added to the model because there is yet to add a storage system to save these images and files for someone to view later. This will be a Milestone 2 issue, handling data and its storage and manipulation from the app.

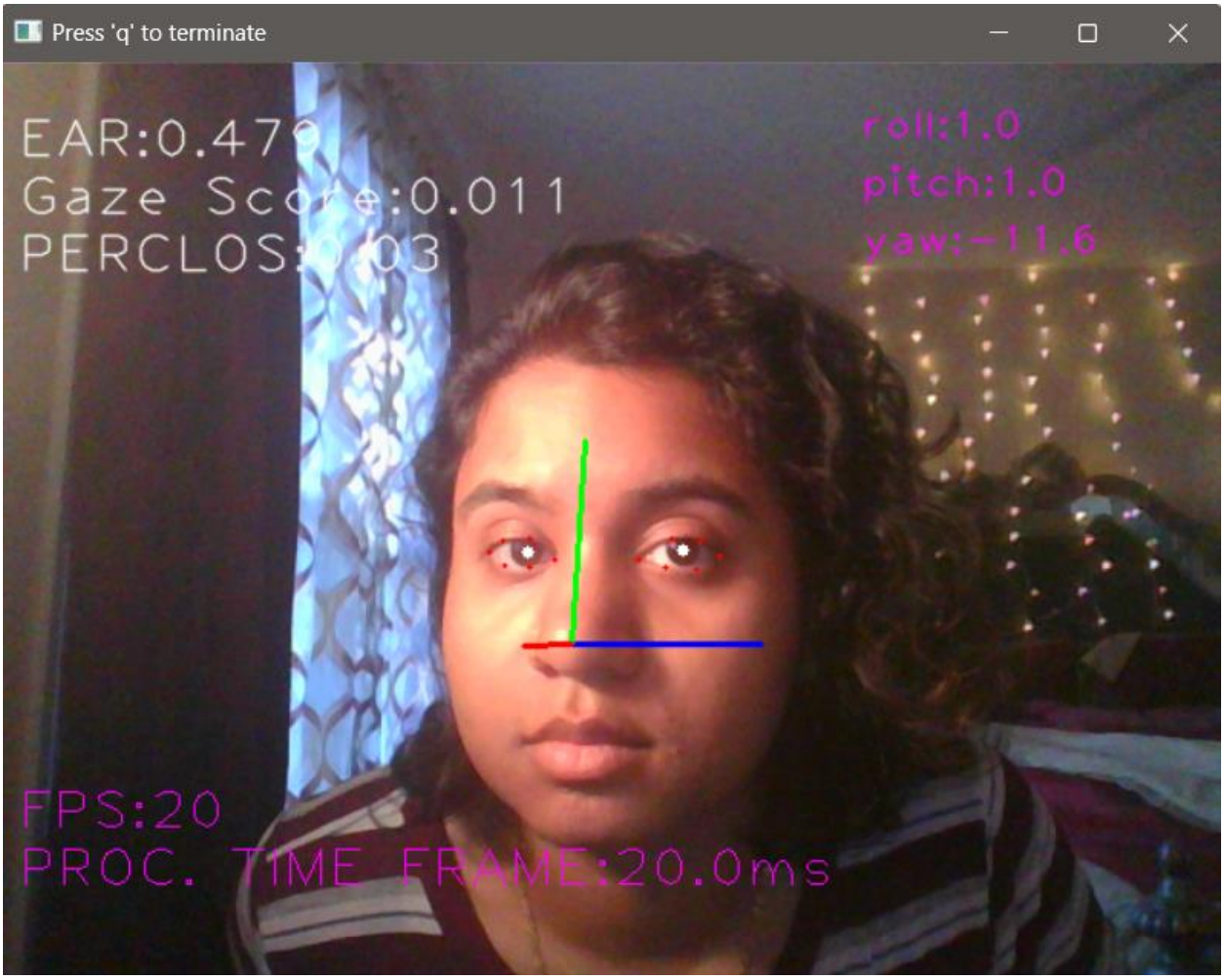
Facial Expressions Development
I've updated the code for this portion of the project to the GitHub page (https://github.com/dita-deb/LocIT/tree/main/Camera%20Module/Proposal)

This phase of the project focuses on calculating a person's eye gaze direction and blink rate, which form the initial part of addressing the expressions challenge.
These two factors—gaze direction and blink rate—are key indicators of whether a person is attentive or distracted. This classification helps determine how individuals around the LocIT wearer engage with them, affecting social interactions and spatial awareness.

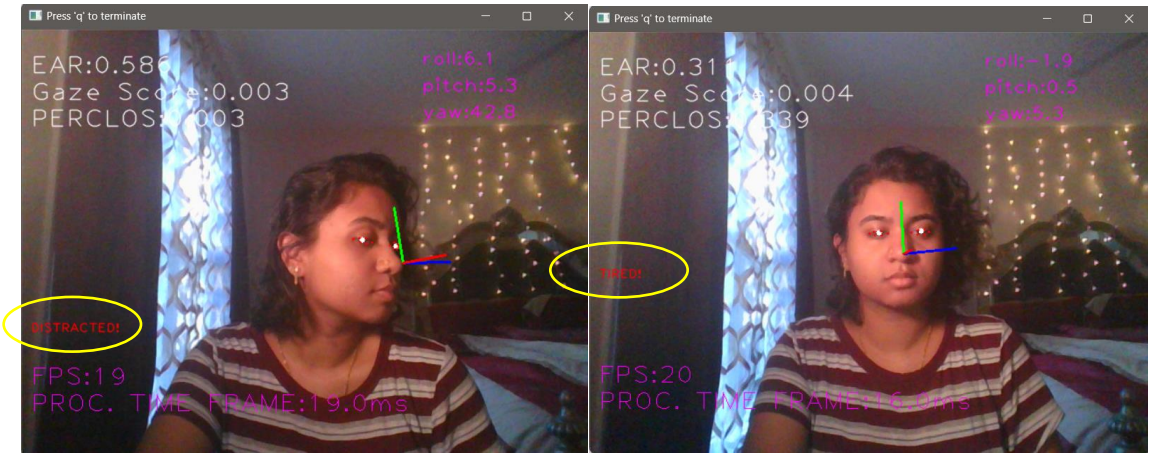
The model uses Haar cascades to detect facial features, followed by PERCLOS for blink rate analysis, and geometric methods for gaze direction estimation.

Additionally, I plan to implement a system for tracking negative or threatening expressions. By flagging such expressions, the system can alert the wearer to potential risks, enhancing their safety and situational awareness. This is also not yet implemented to the hardware or Application yet, as it is a milestone 2 issue.

Here is a sample of what this model does:



The measurements here are displayed to check whether the model is working or not with “maths” that calculates tiredness/distraction/paying attention etc. This math is pre-built into the base model which I’m just changing some values to better check and flag certain expressions.

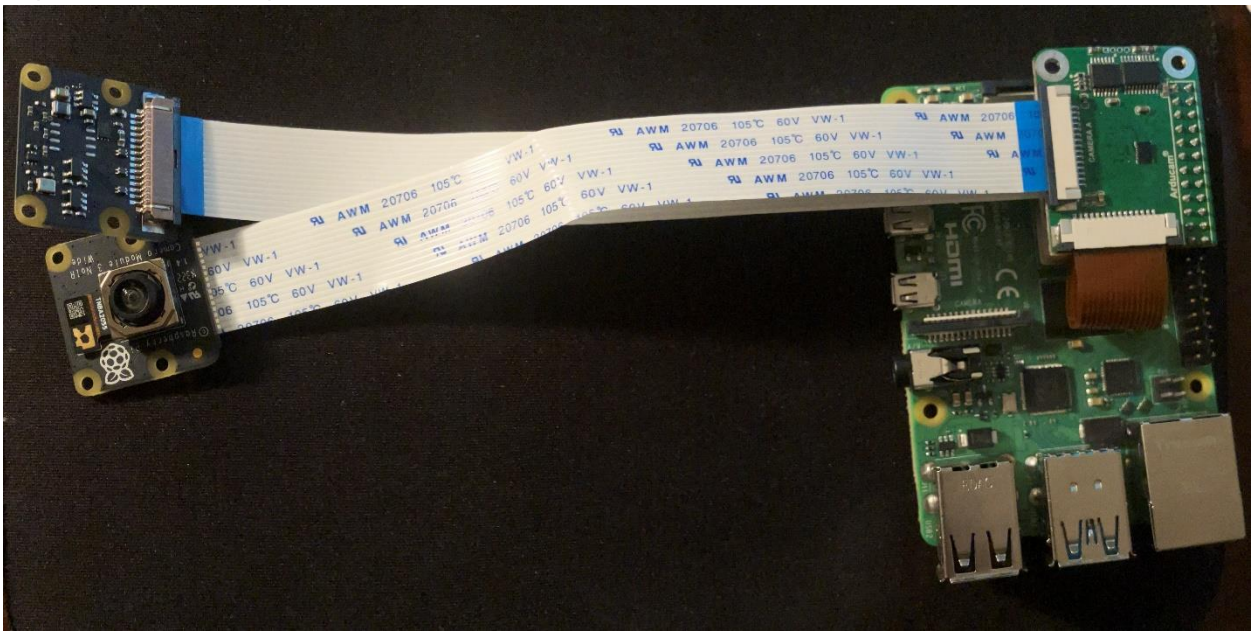


These images show a red text labeled “Distracted” and “tired” which can flag the interaction in different situations. Even if there is a person in front of the LocIT it would not flag the interaction as anything dangerous for the user or demand the individual be added to the dataset.

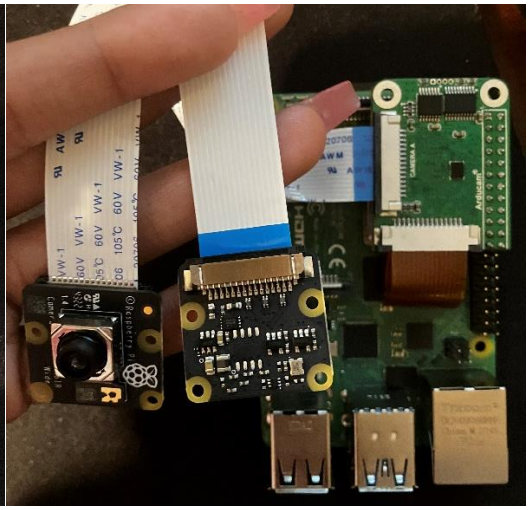
Software to Hardware Integration

The software to hardware implementations are in development as the model requires better storage for the images and video stream.

I have started using the Raspberry Pi 4B+, 2 PiCam Noir 3 Wide Module and ArduCam Header for implementation setup.



This is the set up, the cameras and header on the raspberry pi 4b+ are also compatible with the RaspberryPpi Zero 2 W. I'm using this board mainly as a way to quickly check the model working rather than than the Zero 2 board as a way to avoid any SW/HW issues that may come up when creating the final product.



The ArduCam header allows me to connect 2 cameras to the Pi and have it checking whether there is a feed coming through one side or not to turn off a camera or not depending on the side that has live feed. The Pi on it's own only allows one camera to operate at a time. This header allows for multiple cameras with a single component addition that is sized to fit the Pi Zero 2 W perfectly.

Issues/Challenges

IoT AWS implementation is going to be what our next milestone will encompass. The App development will also play a large part in data collection/storage/display for the next milestone. The integration portion of the hardware to software will be our biggest hurdle in the upcoming weeks.

Additional Information

Any additional information will be provided by the other reports.