# Posture Detector

Kao-Feng Hsieh
Department of Computer Science
Binghamton University
khsieh3@binghamton.edu

Wenzhe Leng
Department of Computer Science
Binghamton University
wleng1@binghamton.edu

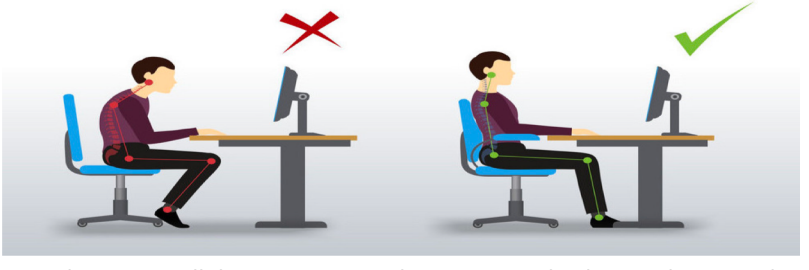
## 1 Introduction

Being Computer  Science students, we always suffer from using computers everyday. Moreover, if we use computers with poor sitting postures, it'll bring several negative impacts to our bodies. Including rounded shoulders, headache, bodily pain and so on. So we decide to build a device to monitor the postures when we are using computers. This device can help us not only adjust our postures while we are using computers but also remind us to take a rest after using it for a period of time. It  also has face recognition functionality to protect your computer from the users other than you.

## 2 Design

### 2-1 Idea

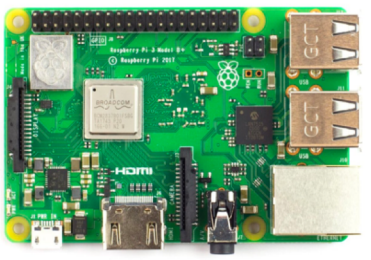
We measure the distance between the user's face and the computer to determine if the user has a poor sitting posture or not. In general, if the distance is too short, it means the user is too close to the computer. At this point, the user may slouch in front of the computer. We simply utilize a Raspberry Pi 3B+ and a camera module to achieve our goal. First of all, Raspberry Pi with camera is installed on the monitor of the computer. It uses camera to recognize whether there is a user or not. When there is a user, it then start measuring the distance and recognizing the face of the user. If others are trying to use the computer, our device will take a picture and send it to the user. We also have a AWS based Database to collect the data and display it on a website or a mobile device.



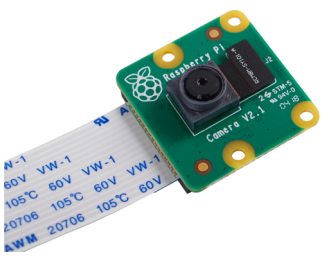
This figure shows why we can tell the sitting posture by measuring the distance between the user and the computer

### 2-2 Hardware components

Raspberry Pi 3B+



Camera Module

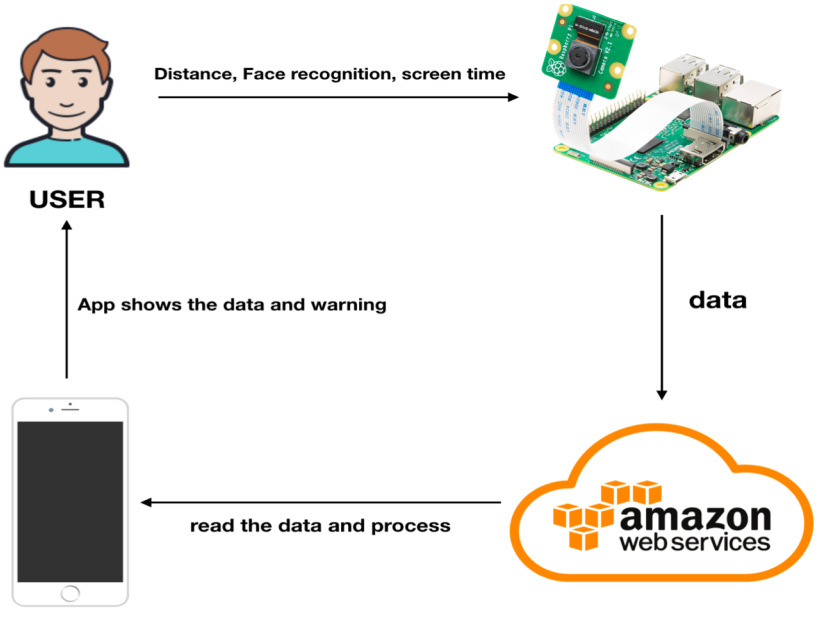


### 2-3 Software

For the Raspberry Pi, we use python3.7 and OpenCV4 for distance measurement and face recognition.

### 2-4 Flow

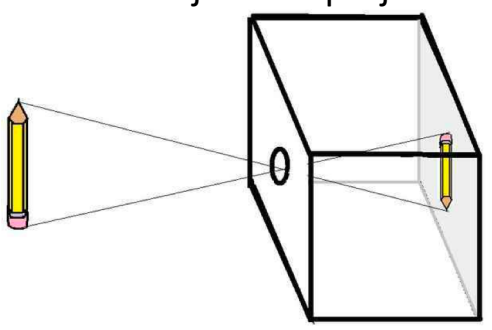
First of all, the Raspberry Pi will collect the data from the user, including distance, face recognition and screen time. After processing the data, it'll upload them to the database in every 5 seconds. Next, our end-user app will read the data from the data base and process them. Finally, the app will show the information for the user on the device.

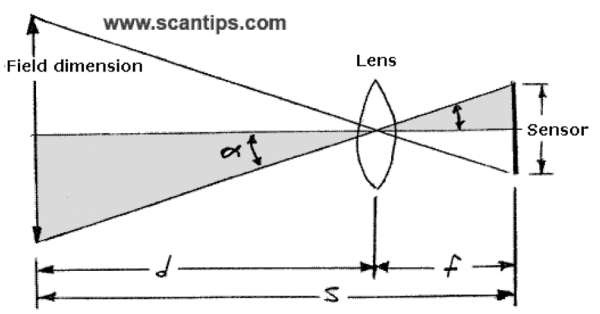


# 3 Implementation

### 3-1 Distance Measuring

The idea of measuring distance is very easy to implement. After we study the theory of how does camera get the image from real object, we found out it works just like projection in Pinhole theory.






We used trigonometry to calculate the distance between the camera and user. Since we knew all parameters we need, we can get the distance by this formula.

$$\text{Distance to object}(mm) = \frac{f(mm) \times \text{real height}(mm) \times \text{image height}(pixels)}{\text{object height}(pixels) \times \text{sensor height}(mm)}$$

*f(focal length)= 35(mm)*
*real_height = 240(mm) => average head size*
*image_height = 240(pixels)*
*object_height = h(pixels) => in faces object*
*sensor_height = 40(mm)*

By using this method, we can get the distance from a 2D image. We used a measuring tape to measure the real distance to evaluate the result, the distance we got is significantly accurate. Although it is restricted by the fixed sensor height, it still perform well on our project.

### 3-2 Face Recognition

We used Python and OpenCV for our face recognition implementation. Due to we only have one user for a device to recognize, so we choose not to use other machine learning methods for face recognition implementation. Additionally, using OpenCV for face recognition can save some CPU usage of the Raspberry Pi. First of all, we used Haar Cascade classifier for face detection. Secondly, gathering the photo and process them for training the OpenCV recognizer model. Finally, implemented the model we trained for the face recognition.

### 3-3 SMS and Email

We used smtplib library in Python to implement the functionality of sending messages to end-user device. The library can not only send emails but also SMS. Every carrier has their own sms-gateways. You can send an SMS through these gateways by internet as long as you know which gateway you are going to send. There are three scenarios that the Raspberry Pi will send the messages. First of all, when the screen time is more than 30 minutes, it'll send a message to the user and remind her is time to take a rest. Secondly, when the user is too close to the computer, it'll alert the user to keep the distance to the computer. Finally, when someone is trying to use the computer that the Raspberry Pi can't recognize him/she. It'll take a picture and send it through the email, also send an SMS to alert the user.

### 3-4 Data Collection

We used the Springboot framework to complete the Web pages design and write an API for ourselves to call. The API is the interface between the Raspberry Pi and the database. The Raspberry Pi requests a post to the interface in the API, it then stores the data and picture in the database.

### 3-5 End-User Interaction

We read the data from the database to the web pages and processed the data, such as calculating the average distance everyday. Our web pages also provide functions for users to retrieve information based on the time. It demonstrate the chart of the distance changing at different times of a day. Additionally, the percentage of days that user in the adequate distance of the month is visible in pie chart.

# 4 Evaluation

In this project, we've implemented several functionalities and successfully evaluated the project. The Raspberry Pi works brilliant most of the time. But they are still problems we need to solve in the future. First of all, it seems a little bit overload for the Raspberry Pi 3B+ to do so many tasks at the same time. For example, when you watch the demo video, you'll probably notice that there are lags in the window. Especially when the Raspberry Pi detects there is other person in front of the computer, the screen will stuck when it takes a picture and send it through the email .The Raspberry Pi uses almost 100% of CPU every time when running the project. I had

tried to use the thread to solve this scenario. But the problem is that the Raspberry Pi does not have enough power to do these tasks at the same time. It still cost almost 100% of CPU usage. Additionally, messages sometimes disappear after sending SMS. I think it’s the gateways on the carrier miss the messages. Overall, in my opinion, the project still has room to improve in the future. But it’ll be a useful product for people who use computer frequently.