COMP7310 GROUP PROJECT

Kang, Zheming; Wu Tsun Ho, Wang, Keqing, Yang, Yulin

The University of Hong Kong

ABSTRACT

This project implements an end-to-end heart rate monitoring system based on the ESP32 camera and Mqtt communication protocol. By performing POS_WANG and FFT on the face data extracted by the camera, this project can calculate the heart rate based on the user's facial information in a certain environment and display it in the user client in real time.

***Index Terms*—** IoT, FFT, MQTT, End to End, ESP32

1. DATA ACQUISITION AND TRANSPORTATION

In this project, the development of ESP32 camera was coded based on Arduino. The code was implemented in Arduino and transmitted to IoT objects so the device can be correctly initialised and process the code locally. The code that was transmitted into the device accomplished the following objectives: camera initialization, WIFI and MQTT(Message Queuing Telemetry Transport) connection, image data transportation, and configuration based on the message it subscribed from MQTT broker.

The initialised code of the camera was constructed based on a simple example of a camera web server in Arduino but the configuration of "jpeg_quality" and "fb_count" were separately changed to 15 and 2. "jpeg_quality" with higher value can compress the fetched image to jpeg format with smaller data size, and "fb_count=2" allows ESP32 camera to capture and transmit image simultaneously.

The ESP32 camera can connect to the WIFI if the SSID and password is presented in its embedded code. The data transmission is based on MQTT protocol, and we used AWS IoT service to build a MQTT server for the data transmission. The image data is continuously published to one of the topics in the MQTT server and the dashboard can subscribe to the image data and visualise it. The sampling rate of image capturing was based on how many times the ESP32 camera captured the image in one second, and the fluency of video depends on the data size of image and the transmission rate of MQTT broker. A potential method to improve the fluency of video streams is to compress them into a video format instead of jpeg format. While the IoT camera keeps publishing images to the MQTT server, it can change its settings such as framesize, contrast, saturation, brightness and so on, based on the message the camera received from the MQTT server. Therefore, we can use a dashboard to publish a certain message to configure the camera via MQTT server.

2. DATA ANALYTICS

The data analysis process allows us to extract the heart rate information from the video frames. This part contains two tasks:

- Make predictions and evaluations on the given data set.
- Perform heart rate prediction on real-time data from the camera and return it to the UI.

As for the first task, we read a set of MATLAB files (.mat) using the scipy.io.loadmat function. The file contains a range of information like video, GT_ppg, light, motion and so on, but as for our prediction and evaluation, we only need to use video and GT_ppg. The two kinds of data are introduced in detail as follows.

The video is stored as an array of frames. we can print it to see what is going on. And the GT_ppg signal is the ground truth Blood Volume Pulse (BVP) of the person. We learned this information by reading rPPG-Toolbox[1]'s treatment of GT_ppg.

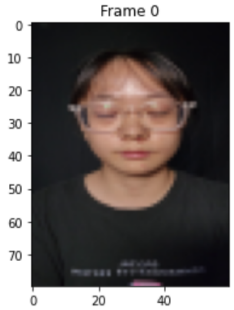

Figure 1. Video frame example of p29_8.mat

After loading the necessary data info, we applied the POS_WANG algorithm to the frames which are extracted from video. This algorithm is written based on the unsupervised method of rPPG-Toolbox, aims to calculate the RGB average values for each frame and processes them to estimate the BVP signal, which involves dividing the

RGB values by their mean, performing matrix operations, and applying filters to the resulting signal.

After implementing the POS_WANG algorithm, we obtained the unprocessed BVP signal. And then, we applied detrending and bandpass filtering using scipy.signal.detrend and scipy.signal.filtfilt functions respectively so that the linear trend in the signal and the frequencies outside the range of 0.75 to 3 Hz would be removed. In the end, we got the processed BVP signal for us to focus on the heart rate-related components.

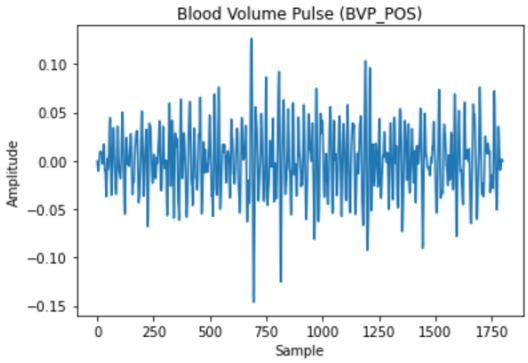

Figure 2. BVP of p29_8.mat calculated by POS

Finally, we chose the Fast Fourier Transform (FFT) algorithm to calculate the heart rate from the calculated BVP signal. The scipy.signal.periodogram function was used to compute the power spectral density of the signal, and the peak within the desired frequency range was selected as the heart rate. It is worth noting that peak detection can also be used to extract heart rate from BVP, but after trying it, we found that for different video data, we need to manually adjust the parameters of peak detection, especially the peak height threshold. This is really poor generalization, so we gave up this plan.

For the given data set, we first select all BVP signals and apply FFT to calculate the heart rate and print out the results; then we select the time window size to be 100 and the sampling rate to be 30, use the sliding window to select a part of the BVP signals and draw the curve.

In the end, we use the same FFT to get the ground truth heart rate from the GT_ppg signal, and compare it with the heart rate calculated from the video.

As can be seen from Figures 3 and 4, taking p29_8.mat as the example, our method has a relatively small error in estimating the overall heart rate within one minute of around 5 bpm.

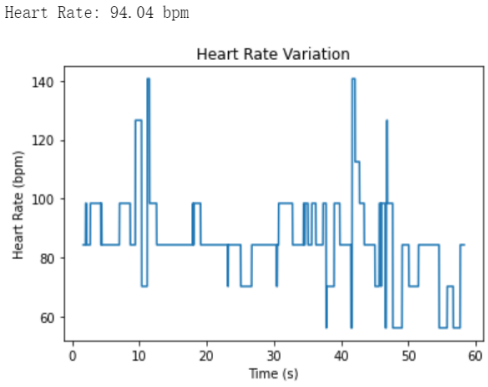

Figure 3. Heart Rate of p29_8.mat calculated by FFT

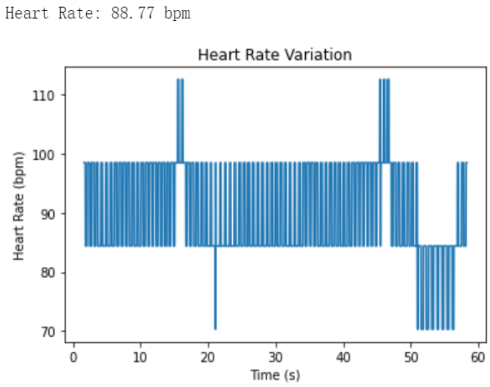

Figure 4. Ground Truth Heart Rate of p29_8.mat

The heart rate calculation part of the evaluation task has been introduced, and the detailed evaluation part would be discussed in 4.(1) Offline evaluation. Complete code is in "Offline evaluation.ipynb".

As for the second task of performing heart rate prediction on real-time data from the camera, the code is implemented and run independently in a separate Python file called "HeartrateToMQTT.py". This part independently subscribes to image data from a MQTT broker and calculates heart rate in real time. The code was implemented with the following logic:

Firstly, a time window was set with a certain size in terms of number of frames. When the code collects enough frames, the real time data analysis starts. In addition, this fixed size time window will change whenever a new frame is gained from MQTT, and the first frame of this window will be removed to maintain the size of the window.

Secondly, the BVP signal is retrieved from this set of image data, and the FFT is used to calculate the heart rate from the BVP signal. And the heart rate data will be plotted into a line graph and saved as a PNG file which will be published into MQTT broker so that the dashboard can subscribe and

visualize it. This part of code is run in a while loop with a time interval of 0.5 secs.

In short, the code continues moving the time window when a new frame is in, and the heart rate will be calculated from it, and a line graph of heart rate will be updated per 0.5 secs.

4. DATA VISUALIZATION

We created a windows client and a backend script for receiving plots from the mqtt server, and plotting the heart rate graph. The client consists of 3 columns, two of whom are control panels and the rest is real-time video and dynamic heart rate plot. Heart rate graphs are plotted by our backend script, which will draw the line graphs and publish them to the same mqtt server in the heart rate topic for the client receiving.

The setting elements of the first two panels are shown in table 1. After changing the values of the panel, our client will publish the setting information to corresponding topics. The ESP32 board will receive the data from specific topics and change the settings.

Elements	**Description**
Client ID	ID for the client
Frame Size	4 options for the size of the video
Sampling rate	Frame rate of the video
Notes	Logs of the system
Brightness	Lighten or darken the video
Contrast	Increase or decrease contrast
Saturation	Increase or decrease saturation
Effects	Multiple options of special effects
White Balance	Switch on/off white balance
WB Mode	Mode of White Balance
Exposure control	Enable/ disable exposure
Heart Rate Plot	Switch on/off heart rate plot

Table 1. Settings of panels

The third column of our client front-end features a real-time camera video from the ESP32 camera and dynamic heart rate information. The data sources of them are the MQTT server. Figure 1 shows the final result of our project. There are two-column setting options, and one-column dynamic videos and graphs. The system will receive the data from the video topic so the real-time video will appear immediately when we open the client script. After we open the backend script, it will get the data from the server and draw the line graphs. Then, it will publish the graphs to the MQTT server again. The client will receive a heart rate plot from the server when there exists data on the server and the user switches on the Heart Rate Plot option.

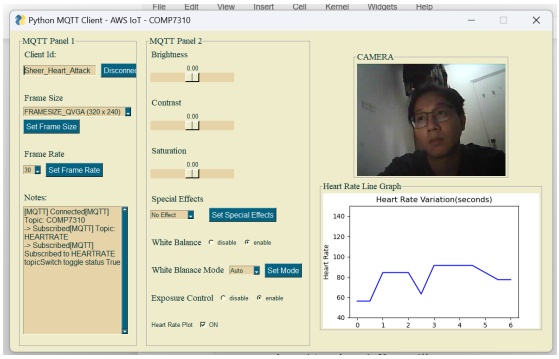

Figure 5. Client front-end

4. EVALUATION

(1) Offline evaluation

We use the given data set to test the algorithm offline. The algorithm process has been introduced in detail in 2. DATA ANALYTICS section. Here we only focus on the analysis of the evaluation results.

For each given data set, first use the POS method to obtain the BVP, and print the total heart rate value calculated by FFT of the entire BVP_POS signal, and the ground truth heart rate value calculated by FFT of the GT_pgg signal. Then, use the sliding window to calculate the heart rate. For each window, extract the window data from BVP_POS and GT_ppg, use FFT to calculate the FFT heart rate, and store the calculation results in heart_rates_pos and heart_rates_sample respectively. Finally, calculate the absolute errors of heart_rates_pos and heart_rates_sample, and print the mean absolute error (MAE).

As for the sliding window algorithm, we set the window size as 100 and the sampling rate as 30. For every frame that passes our window slides, a value is calculated and recorded. Here in the given data set, the video is 60 seconds long, 30 frames per second. So the detection frequency here is 0.033s.

As shown in the below figure, our algorithm has an average MAE of around 20.

```
p29_0.mat :
BVP_POS:  65.0390625  BVP_SAMPLE:  95.80078125
FFT Mean Absolute Error: 18.339154411764707
p29_16.mat :
BVP_POS:  60.64453125  BVP_SAMPLE:  97.55859375
FFT Mean Absolute Error: 22.549632352941178
p29_4.mat :
BVP_POS:  85.25390625  BVP_SAMPLE:  94.921875
FFT Mean Absolute Error: 20.754595588235293
p29_8.mat :
BVP_POS:  94.04296875  BVP_SAMPLE:  88.76953125
FFT Mean Absolute Error: 10.927389705882353
p6_0.mat :
BVP_POS:  59.765625  BVP_SAMPLE:  53.61328125
FFT Mean Absolute Error: 25.395220588235293
p6_16.mat :
BVP_POS:  76.46484375  BVP_SAMPLE:  60.64453125
FFT Mean Absolute Error: 24.245404411764707
p6_4.mat :
BVP_POS:  75.5859375  BVP_SAMPLE:  53.61328125
FFT Mean Absolute Error: 23.34375
p6_8.mat :
BVP_POS:  59.765625  BVP_SAMPLE:  52.734375
FFT Mean Absolute Error: 19.6875
SUM FFT Mean Absolute Error: 165.24264705882354
```

Figure 6. Evaluation result

(2) Online evaluation

We performed a heart rate test of about a minute and a half for each of the four members of the group, and the result is shown in Figure. 4. The image frame window that was used for online evaluation was 150 frames, and the backend scripts calculated the heart rate in the unit of BPM(beat per minute). The heart rate trends of four people fluctuate because our heart rate detection algorithm detects the heart rate based on the variation of pixels in the image. If some of the pixel values change during the detection, then the detected heart rate may change. However, the detection maintains an acceptable range from about 50 bpm to 140 bpm, which is a normal range heart rate for an adult.

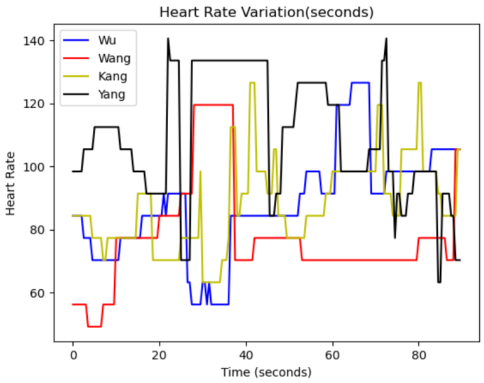

Figure 7. The trend of heart rates

In addition, the average heart rate for each group member and the overall heart rate for all members in the group were calculated as shown in Figure. 5. The average heart rate of all members are in a normal range from 77 bpm to 110 bpm.

```
Heart rate for Wu = 88.2421875 BPM
Heart rate for Wang = 77.890625 BPM
Heart rate for Kang = 87.6171875 BPM
Heart rate for Yang = 109.140625 BPM

In [3]: print('Overall heart rate=',(sum(data1)+sum(data2)+sum

Overall heart rate= 90.72265625 BPM
```

Figure 8. The average heart rate

5. CONCLUSION

This project implements an end-to-end heart rate monitoring system based on the MQTT protocol. Just open the ESP32 camera and connect to the corresponding aws server. Users can get their current heart rate through this project. We used the pos_wang algorithm to help achieve this. The offline evaluation resulted in an average error of about 20, and the online evaluation indicated an acceptable real-time heart rate detection. The heart rate detection algorithm can be more robust if the function of face recognition is adapted into our system. The detection result will be more stable and reliable if only the face features are captured by the system, and the detection result should be independent of the image background.

6. CONTRIBUTION STATEMENT

Tsun Ho: Implemented scripts on ESP32 board and Client Backend Scripts. Helped the system to transmit data from the IoT equipment to the PC. Plotted heart rate line graphs and publish to the server.

Zheming: Implemented the Client frontend and set up the AWS MQTT server. Helped the system to show the video and the heart rate graphs. Provided GUI for users to change the settings of the camera.

Keqing: Implemented the heart rate detection algorithm. According to the video frames, the corresponding heart rate value is returned after POS and FFT calculations. Completer of the offline evaluation “Offline evaluation.ipynb”. Demo presenter.

Yulin: Helped to implement the real-time heart rate algorithms, tested the feasibility and usability of the system. Helped beautify the interface.Integrate everyone's work and write reports.

7. ACKNOWLEDGEMENTS

We hereby acknowledge that the provided data from the MMPD dataset is used for and within this project only. All the data that we have downloaded and used will be removed after the completion of this course. [Kang, Zheming; Wu Tsun Ho, Wang, Keqing, Yang, Yulin].

8. REFERENCES

[1] Liu X, Narayanswamy G, Paruchuri A, et al. rPPG-Toolbox: Deep Remote PPG Toolbox[J]. arXiv preprint arXiv:2210.00716, 2022, 6: 13.

[2] Tang J, Chen K, Wang Y, et al. MMPD: Multi-Domain Mobile Video Physiology Dataset[J]. arXiv preprint arXiv:2302.03840, 2023.