RORRIM: A Health-Centric Smart Mirror

Isabelle P. Wang, Johnathan Tang, Carissa A. Sevaston, Kevin S. Zhu, Bryan D. Trinh, Salma Elmalaki

***Abstract*—RORRIM hopes to encourage the use of smart technology to promote a healthier lifestyle. It currently has a functioning home screen implemented through Python Tkinter and multi-threading that takes in touch input and displays the widgets for time, weather, and health data (extracted from a Fitbit Sense). It also has facial recognition, implemented primarily through OpenCV and the facial_recognition library. User information is stored within an internal database implemented as a JSON file. We hope to implement machine learning to make personal inferences based on health data collected from the wearable. Future plans for RORRIM include adding the Oura Ring as an additional wearable device for interaction, uploading data to a cloud, encrypting user data, and skin irregularity detection.**

***Index Terms*—Data Augmentation, Deep Learning, Facial Identification, Fitbit, Heart Rate Variability, Pattern Analysis, Smart Mirror**

I. Introduction

WITH the emergence of COVID-19, the issue of regular health monitoring has become a prominent global and social issue. Since mirrors are ubiquitous and inconspicuous objects of daily use which people use extremely frequently and perhaps even subconsciously, they are an excellent tool for encouraging individuals to monitor their health more closely. The goal for RORRIM is to promote the habit of regularly checking health measurements such as temperature and respiration level through the practice of routinely checking the mirror.

The idea behind a health-monitoring smart mirror is not a new concept. The MIRROR by lululemon is the latest trending smart mirror on the market. It has an emphasis on promoting fitness through its streaming service for workouts and fitness classes displayed directly on the mirror [1]. The MIRROR also allows for pairing with an Apple or Android smartwatch to allow for display of heart rate on the mirror. The Wize Mirror emphasizes non-intrusive symptom analysis and "sustained engagement" to promote the prevention of diseases. It also performs "touchless" analysis through a series of cameras and optical sensors with the exception of the breath analysis device, the Wize Sniffer [2]. Other existing smart mirrors include facial recognition authentication implemented using a LCD monitor, Raspberry Pi 3, and camera [3]. Facial recognition is used in conjunction with web services to further enhance the user experience. Face recognition implementation on the Raspberry Pi is not a new issue, as there have been many libraries built and updated to assist in detecting and identifying faces, such as Open-Computer-Vision (OpenCV) [4]. Facial recognition and identification go hand in hand, as Haar-Cascades are used in detecting, and Eigenfaces, Fisherfaces, and Local double example histograms are used in matching [4] [5].

Supported by The Green Initiative Fund

A. Project Overview

RORRIM takes inspiration from past smart mirror implementations. However, it will also use information from wearable devices and machine learning to make health recommendations to its users, such as suggesting better sleeping habits. By displaying these inferences and health statistics in plain view on the mirror, possible symptoms may be brought to the users' attention and allow them to take early action. Thus, RORRIM hopes to encourage the use of smart technology to promote a healthier lifestyle.

II. Materials

Materials for this project includes:

- Raspberry Pi 3 B+ [1]
- Raspberry Pi NOIR Camera Module V2 [2]
- HC-SR501 PIR Motion Sensor [3]
- 24" Multi-Touch 10 Point Infrared Touch Frame [4]
- Sceptre E248W-19203R 24" LCD Monitor
- Reflective Film
- Acrylic Sheet
- Fitbit Sense [5]
- Oura Ring

RORRIM's central computer is a Raspberry Pi. A Raspberry Pi Camera is used for facial recognition. A passive infrared (PIR) sensor is used for motion detection. A reflective film is applied to an acrylic sheet which produces a reflective surface. This combined component is attached to the surface of a computer monitor to produce the basic smart mirror. The Fitbit and Oura Ring are the smart wearables we plan on integrating with RORRIM as far as exporting end-user's health data.

III. Progress

A. Home Screen

RORRIM's current functionality includes a basic home screen display. The display is implemented using Python and its built-in graphics library, Tkinter. Our initial prototype display only contained the standard smart mirror widgets—time and weather—in the top left and right corners respectively. Datetime is retrieved directly from the Raspberry Pi. Weather

[1]Raspberry Pi 3 B+ will now be referred to as Raspberry Pi

[2]Raspberry Pi NOIR Camera Module V2 will now be referred to as Pi Camera

[3]HC-SR501 PIR Motion Sensor will now be referred to as PIR Sensor

[4]24" Multi-Touch 10 Point Infrared Touch Frame will now be referred to as IR Frame

[5]Fitbit Sense will now be referred to as Fitbit

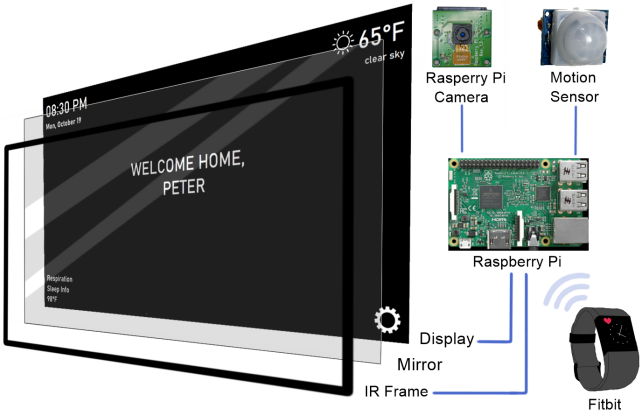

Fig. 1: Conceptual diagram of hardware components

and forecast information is retrieved through an API call to Open Weather Map [6] [7][6].

The second and current version of the home screen, Figure 2, added the step count and heart rate of the user from the Fitbit onto the display. The health statistics are extracted from the wearable via calls to the API and updates every 5 s using similar multi-threading methods as before. This version of the home screen also includes buttons to add and change users. The ”Add User” button prompts the virtual keyboard (implemented using Tkinter) to appear and allows users to input information such as name and location. The ”Change User” button currently triggers the PIR motion sensor and facial recognition code to run and detect a known user.

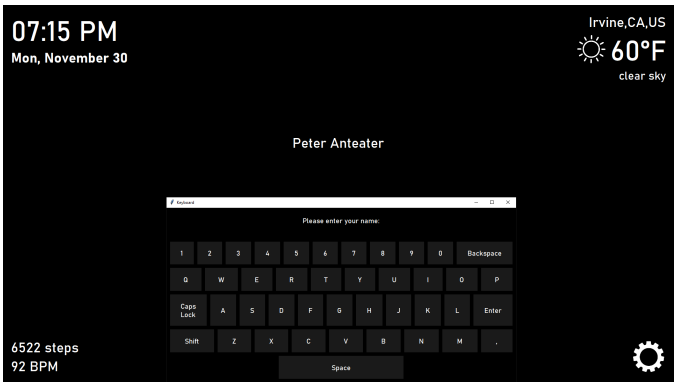

Fig. 2: User interface of the RORRIM’s home screen

B. *Facial Identification*

RORRIM also has facial identification to recognize different users to store each user’s personal health information. The facial detection module uses Python OpenCV to load and manipulate the image, and the face_recognition library in order to recognize faces from a sample encoding database to match faces using deep learning. The facial identification module will check if an updated cache exists. If there is an updated cache, then the module will load the encodings; if there is no updated cache, the encodings will be created from a known face directory, and the cache will be updated. Then the unknown encoding will be compared to the list of known encodings, to see if there is a close enough match, under a particular tolerance level. See Figure 3 below for a graphical interpretation of this process. This process allows the user to log in through facial recognition and access their personal health information in a JSON file, which is used in conjunction with our display module.

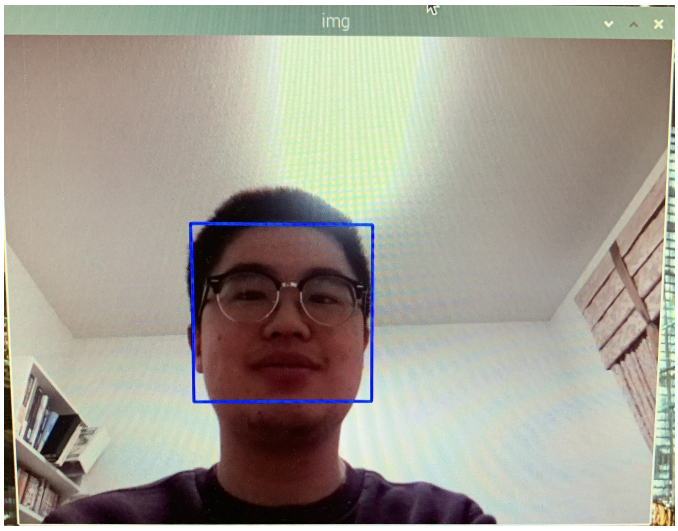

Fig. 3: Face is boxed by RORRIM’s facial recognition

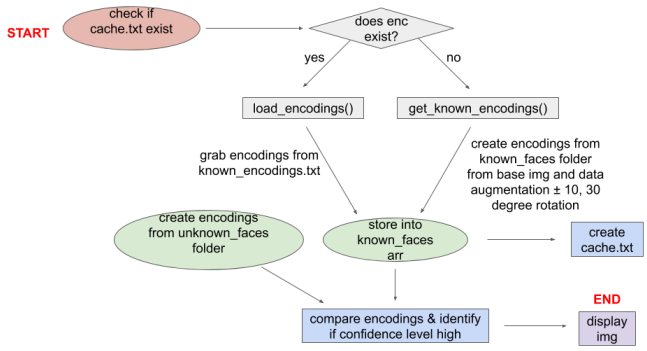

Fig. 4: Steps of the facial recognition module

As seen in Figure 4, to prevent the Raspberry Pi from constantly identifying a face if there is no active user, as well as overheating, RORRIM uses a PIR sensor to alleviate constant identification. The PIR sensor will detect if there is motion and activity whenever there is no active user. If motion is detected, then the facial recognition module will then be activated. Whenever there is no more active user, the Raspberry Pi will be put into a lower power state by occasionally detecting motion to trigger the facial recognition module.

[6]We use two API calls to Open Weather Map. The Current Weather API [6] is used to retrieve the user’s latitude and longitude coordinates. The One Call API [7] is used to retrieve actual weather and forecast information based on the results of the previous API call.

C. Smart Wearable

RORRIM allows for the user to sync their wearable device to interact with the mirror. Currently RORRIM only allows for Fitbit interaction. To export health information from the wearable device, the Fitbit Web API [8] is used to access data collected from the user. Each Fitbit user is assigned a user identification number as well as an access token to export data from the device. Through API calls, RORRIM will make GET requests to access and store personal health information such as heart rate, steps taken, and sleep levels. RORRIM also databases health data in accordance to multiple users. The health information is then used in machine learning and regression models to create health recommendations based on inferences made.

IV. Future Work

For the winter quarter of 2021, we plan on finishing the recommendation feature to make inferences on health data[7] using machine learning. Furthermore, we plan on incorporating an Oura Ring in a similar role as the Fitbit. Since we expect the amount of data collected from both wearable devices to occupy a significant amount of local storage, we are considering the use of cloud scaling to manage data off the Raspberry Pi. Since RORRIM will upload private data to the cloud, we plan to enforce encryption to ensure privacy for all end-users. If time permits, we hope to support additional features such as skin irregularity detection as well as adaptive lighting.

V. Conclusion

RORRIM extracts health information from a Fitbit and displays these statistics alongside time and weather widgets. Users can interact with the mirror using touch input produced from an IR Frame. The mirror also incorporates facial identification and recognition using OpenCV and facial_recognition library. Through machine learning, we hope that RORRIM makes precise recommendations based on health information inferences about its end users.

The Smart Mirror sees a place in the common household in the future. Alongside checking the mirror daily, integration of health components allow for monitoring of health. This data can make recommendations to the users (a feature we implemented), but the data can also be uploaded to the cloud for health providers. Monitored by AI, health organizations can alert doctors about patient health and critical warnings. Reminders, monitoring, and recommendations can greatly improve health coverage. As the future of the smart home comes closer into reality, RORRIM aims to encourage the public to engage in health awareness.

Appendix A
Technical Standards

Technical standards for RORRIM include the use of Bluetooth, WiFi, USB 2.0, and MicroSD. These standards are necessary for the functionality of RORRIM, which includes connecting to a user's wearable and connecting the Raspberry Pi to other peripherals such as the Pi camera and PIR sensor. By using the devices listed in Section 2 in a plug-and-play fashion (as opposed to altering any hardware), RORRIM strictly follows standards that are in compliance with regulations such as FAA and FCC. As the main driver for RORRIM, the Raspberry Pi is widely used and conventional as far as abiding to official governance. Furthermore, RORRIM uses JSON format for databasing due to the fact that JSON is a conventional format for most database platforms. In regards to assembly, there are concerns with the Pi processor overheating due to the runtime and infrequently large loads. To address this issue, cooling solutions are being considered for the end product: including heat sinks, fans, and coolant. In order to address OSHA regulations, we plan on using zip-ties and cables to make sure that the wires are fastened in place [19].

[7]The health information we plan to extract and analyze includes sleep duration, sleep levels (wake, light, deep, rapid eye movement (REM)), body temperature, heart rate, and respiration rate.

Appendix B
Challenges

A. Working remotely due to COVID-19

Due to the COVID-19 pandemic, the team members of RORRIM had to adjust to working remotely and make some compromises to their original plans. Hardware and materials were purchased and shipped to only two team members. Testing of hardware and new functionality thus became dependent on the single member who physically had all the hardware components. All team members also had to adjust to pair programming and working on shared documents virtually, through platforms such as Github, Discord, Zoom, and Google Drive.

B. Limitations to Data Export

Limitations on the specific data that could be exported from the Fitbit postponed our work. First, we were unable to extract any data from the wearable for the first five days. Secondly, after contacting Fitbit support, we learned that oxygen saturation levels (SpO2) is not available to extract for public use. Lastly, wrist temperature measurements can only be exported through the Fitbit app, as opposed to a direct API call. Since we are not able to extract information relating to respiration rate or body temperature yet, we decided to shift our focus onto the sleep, heart rate, and steps data that we do have in order to start implementing machine learning algorithms. Furthermore, we are currently following up with our funding contributor to obtain an Oura Ring, since we will be able to extract the temperature and respiration information through its API calls.

C. Refactoring for Integration

During Week 8 of our project, the team members realized that some miscommunication about the user flow caused reintegration errors between code for facial recognition, Fitbit, and display. Thus, major refactoring occured. All team members attended an emergency meeting to create new classes for better code organization and a new user flow, Figure 5.

To prevent future re-integration errors, we plan to add more documentation to our software and to have different teams touch base more often.

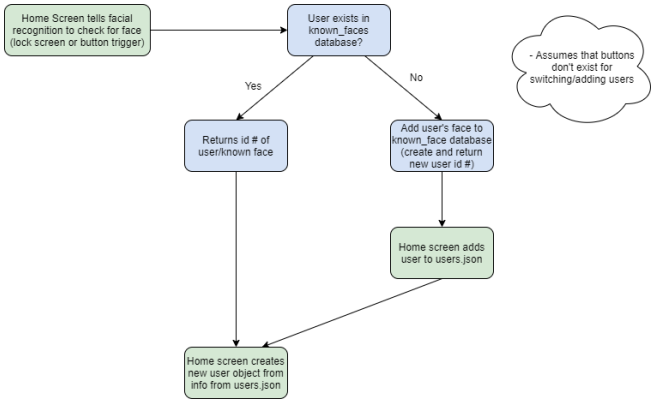

Fig. 5: Flow chart between Facial Identification and Home Screen

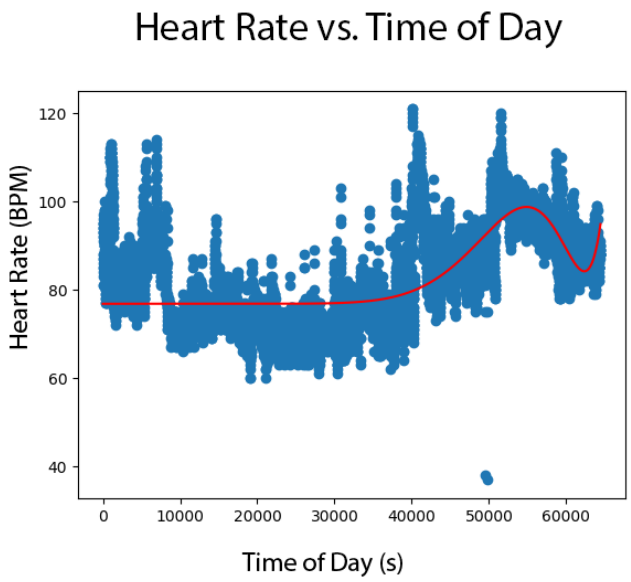

Fig. 6: Polynomial Regression (degree 15) graphing duration of sleep (sec) against heart beat (bpm)

D. Learning AI/ML

All team members had no prior experience in artificial intelligence or machine learning going into this project. Thus one of the biggest challenges this quarter was learning how to implement machine learning in preparation for the recommendation feature. Although our adviser gave us some initial direction[8], the team members learned how to implement simple linear regression models and machine learning primarily by reading online articles [20] and following along with Kaggle notebooks [9]. The team members are currently focusing on recommendations using sleep duration, sleep level, and sleep quality as features and target values. We plan to look more into Hamming loss and multi-label classification in the near future. Although we have not finished researching and fully implementing machine learning, we have succeeded in graphing Fitbit-extracted information with polynomial regression, Figure 6.

Although these exact values will not be used to make inferences within our recommendation feature, this is our first successful attempt at graphing polynomial regression using Fitbit data.

Appendix C
Security Issues

RORRIM will store personal information such as name, location, and health data. Thus, we plan to implement encryption next quarter to protect the privacy of the end users. In the meantime, our current workaround is to reduce the amount of data uploaded into the Internet and instead store it locally on the Raspberry Pi. RORRIM pulls data from the Internet through established web API's (i.e. Open Weather Map and Fitbit). Information that is pulled will only be accessible to the mirror's back-end, with certain downloaded data being displayed for the user, such as location and health information.

We eventually plan to simplify the user interface to be more linear and streamlined and aim for code coverage. In addition to encryption, to prevent reverse-engineering, we also plan to also encrypt all the files and the data in the Pi itself, in addition to compressing files into DLL files to run them dynamically to further safety.

[8]Our adviser introduced us to Kaggle. In addition, she told us to research more about linear regression, hamming loss, and multi-label classification to get us started.

Acknowledgements

We thank our team's adviser, Assistant Professor Salma Elmalaki of the Department of Electrical Engineering and Computer Science at the University of California, Irvine (UCI), for guiding us in our research and project direction. In addition, we would like to thank the EECS 159A/B Senior Design Project teaching staff— Professor Stuart A. Kleinfelder, Dmitry V. Oshmarin, and Sahar Nikbakht Aali— for their guidance and advice throughout the quarter. We are grateful to The Green Initiative Fund (TGIF) at UCI for financially supporting our project.

We appreciate OpenWeatherMap for their One Call [6] and Current Weather [7] API's, which we used for RORRIM's weather widget. We appreciate the Fitbit company for providing us with API function calls through Swagger UI [8]. We used the Kaggle "Sleep Data" dataset, published by Dana Diotte [9] to provide initial data for our machine learning sleep analysis, since new users have no information to start.

We would also like to provide proper attribution to the online resources we used to educate ourselves and allow us to implement RORRIM. As team members had no prior experience using Tkinter, we used David Amos's article [10] as an introduction to Tkinter. Our virtual keyboard is also heavily influenced by the keyboard tutorials of Suraj Singh [11] and Danish Ali [12].

We would like to provide proper attribution to Python's OpenCV library, maintained by Olli-Pekka Heinisuo and user skvark [13], and Python's Facial Recognition library, by Adam Geitgey [14], as the libraries were used for the basis of RORRIM's facial recognition code. The face recognition code is also heavily influenced by the tutorials of Adarsh Menon [15] [16] and the data augmentation code is influenced by Vardan Agarwal [17] and Thomas Himblot [18].

REFERENCES

[1] "Shop |The Mirror," MIRROR. https://www.mirror.co/shop/mirror#product(accessed Dec. 01, 2020).

[2] S. Colantonio et al., "A smart mirror to promote a healthy lifestyle," Biosystems Engineering, vol. 138, pp. 33–43, Oct. 2015, doi: 10.1016/j.biosystemseng.2015.06.008.

[3] I. Garcia, E. Salmon, R. Riega, and A. Barrientos, "Implementation and Customization of a Smart Mirror through a Facial Recognition Authentication and a Personalized News Recommendation Algorithm," Dec. 2017, pp. 35–39, doi: 10.1109/SITIS.2017.17.

[4] M. Bansal, "Face Recognition Implementation on Raspberrypi Using Opencv and Python," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3557027, 2019. Accessed: Oct. 21, 2020. [Online]. Available: https://papers.ssrn.com/abstract=3557027.

[5] T. Mantoro, M. A. Ayu, and Suhendi, "Multi-Faces Recognition Process Using Haar Cascades and Eigenface Methods," in 2018 6th International Conference on Multimedia Computing and Systems (ICMCS), May 2018, pp. 1–5, doi: 10.1109/ICMCS.2018.8525935.

[6] "One Call API," OpenWeatherMap. https://openweathermap.org/api/one-call-api (accessed Dec. 01, 2020).

[7] "Current weather API," OpenWeatherMap. https://openweathermap.org/current (accessed Dec. 01, 2020).

[8] "Swagger UI: Fitbit Web API," Fitbit. https://dev.fitbit.com/build/reference/web-api/explore/ (accessed Dec. 01, 2020).

[9] D. Diotte, "Sleep Data: Personal Sleep Data from Sleep Cycle iOS App." https://kaggle.com/danagerous/sleep-data (accessed Dec. 01, 2020).

[10] D. Amos, "Python GUI Programming With Tkinter," Real Python, Jan. 22, 2020. https://realpython.com/python-gui-tkinter/ (accessed Dec. 01, 2020).

[11] S. Singh, "how to create virtual keyboard using python and tkinter - part - 1," Mar. 18, 2017. https://www.bitforestinfo.com/2017/03/how-to-create-virtual-keyboard-using.html (accessed Dec. 01, 2020).

[12] D. Ali, "How To Create Virtual Onscreen Keyboard Using Python And Tkinter," Master Programming, Aug. 13, 2019. https://masterprograming.com/how-to-create-virtual-onscreen-keyboard-using-python-and-tkinter/ (accessed Dec. 01, 2020).

[13] O. Heinisuo and skvark, opencv-python: Wrapper package for OpenCV python bindings. https://github.com/skvark/opencv-python.

[14] A. Geitgey, face-recognition: Recognize faces from Python or from the command line. https://github.com/ageitgey/face_recognition.

[15] A. Menon, "Face Detection in 2 Minutes using OpenCV & Python," Towards Data Science, Apr. 22, 2019. https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81 (accessed Dec. 01, 2020).

[16] A. Menon, "Face Recognition in Python using face_recognition Library (in Google Colab)," May 07, 2020. https://www.youtube.com/watch?v=987QtKPZ-P0 (accessed Dec. 01, 2020).

[17] V. Agarwal, "Complete Image Augmentation in OpenCV," Medium, May 16, 2020. https://towardsdatascience.com/complete-image-augmentation-in-opencv-31a6b02694f5 (accessed Dec. 01, 2020).

[18] T. Himblot, "Data augmentation: boost your image dataset with few lines of Python," Medium, Mar. 05, 2018. https://medium.com/@thimblot/data-augmentation-boost-your-image-dataset-with-few-lines-of-python-155c2dc1baec (accessed Dec. 01, 2020).

[19] Occupational Safety and Health Standards, 1910.305, 2007.

[20] D. Fumo, "Linear Regression — Intro To Machine Learning #6," Medium, Mar. 05, 2017. https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06 (accessed Dec. 01, 2020).