CrowdVision

Gianna Mascardo, Wesley Bellin, Dylan Agiman, and Jens Tuyls
Professor Stuart Kleinfelder
Department of Electrical Engineering and Computer Science

***Abstract*—CrowdVision uses computer vision and a Raspberry Pi v3 module with a camera to help students conveniently and quickly find an available study room. The camera on the Raspberry Pi v3 takes pictures of the study room at timed intervals and sends it to our processor using Wi-Fi. The processor uses Facebook's Detectron2 library to analyze each image and extract how many people are in it. After the image has been analyzed, it is removed and the extracted occupancy data is stored in our database. Students can conveniently see our collected data through our website and thus can view which study rooms are available in real time. This allows students to focus less on where to study and more on what to study.**

***Index Terms*—Computer Vision, Study Rooms, University**

I. Introduction

FINDING study spaces is a key consideration for students at UC Irvine and other institutions. Currently, room availability is an important piece of information that is not readily available to students. CrowdVision plays a role in providing that information to students, and helps them make informed choices on where to study. Our goal is to create a system that will detect how many people are in a study room to see if it is at capacity, allowing students to spend less time looking for available rooms and more time studying.

Past work in population-counting includes density calculation instead of by-person counting. [1] shows a possible implementation of one such density-estimation algorithm to count higher densities of people. In this case, the system makes an estimation of the number of people in an area rather than physically counting each one. This allows for faster performance, at the expense of accuracy. Other systems have used a break-beam system, where a person is counted once they cross the threshold of a sensor [2]. Yet also this system sacrifices accuracy for speed performance.

Our implementation is ideal for small to medium-sized rooms, because we would want a high degree of accuracy with our calculations. Since the rooms we are implementing CrowdVision in are of this size range, doing a by-person calculation will not be too resource intensive, and still retain a high level of accuracy. If we were to use our system in a larger room, it might make sense to shift our detection strategy to a more estimate-based approach to make sure that the detection can still happen sufficiently rapidly.

II. Materials Used

CrowdVision uses a combination of hardware and software materials. The Hardware side of our project consists of using a Raspberry Pi 3 Model B+, Raspberry Pi Camera Board v2, 5V 2.5A Switching Power Supply, a Raspberry Pi Case (to protect the Board and camera), and Mounting items. The Software side of our project consists of the Facebook Detectron2 as our Computer Vision library, Flask Server on Heroku, and an Azure SQL database for our database.

III. High-level Hardware and Software System

CrowdVision integrates hardware and software systems together to provide students real time data about the availability of study rooms. Figure 2 depicts the flow of data from the cameras, through the processing server, and to the end-user devices, with a database to server as a data buffer. CrowdVision uses hardware to capture images from a study room, which will be sent to our processing server using Wi-Fi. Our processing server will use Computer Vision to analyze the image and extract data from it, such as the time it was taken and the amount of people detected in the image. The server delete the image after it's been analyzed and will store that extracted data into our database. Students will be able to view the extracted data in real time through our web application.

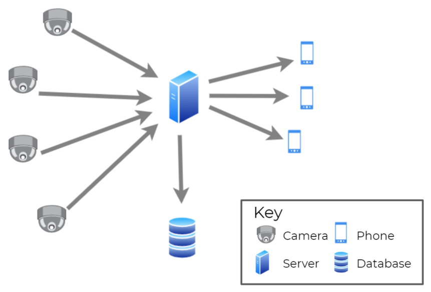


Fig. 1. High-Level Diagram of our Integrated Hardware and Software

IV. Methods - Hardware

While our project is software heavy, we needed a device to capture the state of the study rooms we are analyzing. In order to do this we made use of a Raspberry Pi 3B+ and a camera module to take and send photos to our server for processing. We chose the Raspberry Pi 3B+ due to its internet connectability, remote access, and support available for troubleshooting. Our Raspberry pi operates on the default provided Raspberry Pi operating system called Rasbian. Rasbian allows us to work with the Pi as if it were a normal desktop computer and allows easy remote access. However, despite the ease of remote access, we still had numerous issues on this front. In order to remote into the Pi, the IP address must be known. Whenever we connect the Pi to the university Wi-Fi, its IP changes due to there being several routers to connect to. To overcome this, we decided to use a direct ethernet connection. We did this by purchasing an ethernet to USB converter so that we can set up an SSH from a laptop. Next, we found that Rasbian has supported camera functionality so we were able to take photos and gather data using just a simple command in the command prompt. From here we plan to make a script that will control the camera to take a photo every few minutes in order to accurately gather data from the rooms. In our final project design, we plan to immediately delete any image taken after analysis of the photo is completed.

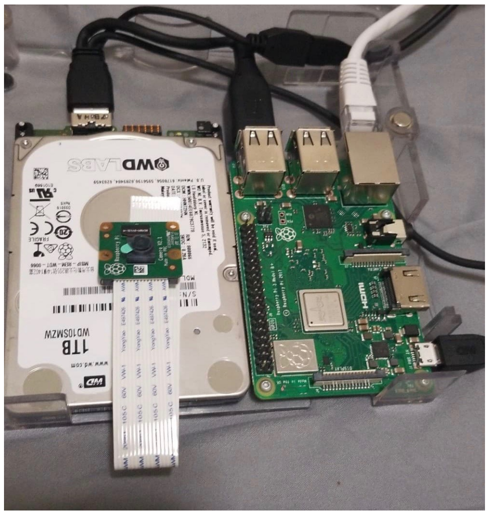


Fig. 2. Our Raspberry Pi 3 Module with the Serial Camera

V. Methods - Software

A. Client-Server Interaction

In order to send the images from the Raspberry Pi to the server it was important to establish a protocol to ensure that the server would be able to receive the information and make sense of it. Thus, we decided to go with an API style of communication with the image being sent as an encoded parameter. This architecture allows the client side to be very flexible. The same software can be run on a variety of different platforms. In addition, the decision to split the software into a client and server side allows the information to be abstracted where it is irrelevant. The client does not need to know the specifics of how the image processing takes place, and the server does not need to know the specifics of how the image was taken. Figure 2 shows a high level overview of how our software and hardware components interact in CrowdVision.

B. Image Processing Strategy

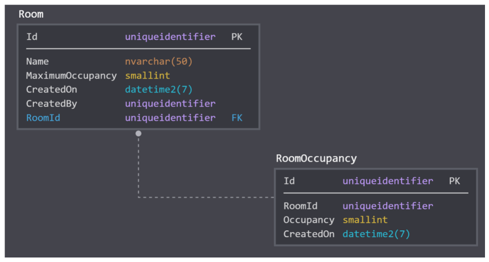
Once the image has been received by the server, the majority of the computation takes place. In order to process the images and count the number of people present, it was necessary to choose an appropriate computer vision implementation. Recent breakthroughs have shown deep learning approaches outperform more traditional computer vision algorithms [3], leading us to look at a variety of libraries that implement these new techniques. We considered OpenCV, Facebook's Detectron2 [4], and SimpleCV. Our need for accurate detection at a variety of angles and distances made Detectron2 a clear winner. The underlying model that Detectron2 uses is a Mask R-CNN [5], a state-of-the-art model for object instance segmentation. Results using models in OpenCV and SimpleCV were inconsistent and had many false negatives. Since accuracy was a key factor for the project, we decided to go with Detectron2.

However, there were still issues with Detectron2 that took time to resolve. First, since it is a relatively new library, there were not many resources that we could refer to, leaving us on our own for the most part with any troubleshooting issues we had. This was problematic, as our system exceeded the free plan memory limits that Heroku offers, forcing us to explore other hosting options. We decided to move to Microsoft's Azure, yet are still having issues getting our model deployed because of compatibility issues with the library dependencies needed for Detectron2. As a current workaround, we are using a local server that we publicly expose using ngrok.

C. Database Schema

In order to allow our data to be queried by our web page (see below), it was necessary to introduce a middleman to handle the data. In our case, we decided to introduce a database to manage the data and organize it coherently. To host our database, we decided to use an Azure SQL database that will allow our database to be accessible from anywhere. An advantage of using Azure for this is that it handles scaling issues, as well as any security issues that could arise from hosting data in the cloud.

Once the web server has finished processing an image, it aggregates relevant information about that image, such as the room name, occupancy of the room, date of creation, etc. The database schema is illustrated in more detail in Figure 3.


Room

Id	uniqueidentifier	PK
Name	nvarchar(50)	
MaximumOccupancy	smallint	
CreatedOn	datetime2(7)	
CreatedBy	uniqueidentifier	
RoomId	uniqueidentifier	FK

RoomOccupancy

Id	uniqueidentifier	PK
RoomId	uniqueidentifier	
Occupancy	smallint	
CreatedOn	datetime2(7)	

Fig. 3. Database schema used to store room information such as occupancy

D. Front-end Web Page

A key component of CrowdVision is allowing the processed data to be easily seen by the end consumer. To achieve this, we developed a web application that would query the data available in the database and display it in a convenient manner. There were multiple options for front-end development, including React, Angular, and .NET Core. However, due to our team being more experienced in .NET Core, we decided to go with that option to reduce the learning time needed to begin development.

In addition, due to the split in front-end and backend, there is no need for the web application to know anything about the image processing or the hardware components. This allows the code to be relatively simple, with the majority of the code being dedicated to the UI components rather than calculations. Since the data is already preprocessed, the web application can simply connect to the database and query it, with the data being appropriately organized by the server.

For future work, we would also like to use collected data over time to make predictions about occupancy of popular study rooms throughout the day.

VI. Results and Performance

First, we were able to successfully take a JPEG image, send it to an exposed local web server, and give a response back that includes an accurate number of people in the image. With respect to performance, the POST request to the server happens very fast, however the image processing is currently the main bottleneck, taking several seconds to perform the instance segmentation. We do however note that this is not necessarily a problem in our final application, as we only plan on taking a picture of the room about every minute. Finally, with regards to the front-end web page, we have developed an initial UI that users will be able to browse to find available study rooms. Performance is currently very fast since data is limited and we currently do not have any real users, but once we acquire more data and gain more users, we may have to look into scalable options for our web app. In addition, once we need to use multiple systems to report on multiple different rooms, we need to make sure that our architecture can support this growth.

VII. Summary

Students currently don't have a way to find information about the occupancy of study rooms on campus. CrowdVision tackles this problem by detecting the number students in study rooms using a Raspberry Pi and Facebook's Detectron2. We are currently able to take the picture and get an accurate count of the number of people in it. Furthermore, we also have the Raspberry Pi setup, as well as a mock web page that shows the front-end of how our fully functioning web app will look in the future.

VIII. Conclusion

Even though we have achieved some major milestones, our project has some important future work to be done. Firstly, we would like to integrate the Pi, server, database, and front-end into one fully functioning system. Once this is done, we can start testing our system and add additional features to the web page, such as showing plots of when study rooms are typically available. We hope that, once finished, CrowdVision can serve as a useful tool for students to conveniently find study rooms on campus.

IX. Acknowledgements

This project was funded from the University of California, Irvine senior design course (EECS 159A) which is an ABET approved course. The equipment purchased for crowdvision was sources from Adafruit and Facebook's Detectron2 library. CrowdVision would like to acknowledge Professor Stuart Kleinfelder for advising this project throughout the quarter. CrowdVision would also like to acknowledge the senior design group CSEEE, Gianna Mascardo, Wesley Bellin, Dylan Agiman, and Jens Tuyls, for all of their dedication and continuing efforts towards this project.

Appendix A
Technical Standards

CrowdVision follows many technical standards, including the Ethernet standard, micro SD card standard, USB standard, Wi-Fi standard, and JPEG file format standard. We chose these standards because it made it easier to gather compatible materials and ensuring connectivity between our software (processor) and our hardware (Raspberry Pi 3 Model B+). The Raspberry Pi 3 Model B+ was designed to be compliant with these common standards to account for universal usability withing various types of projects - such as the Ethernet standard, micro SD card standard, USB standard, Wi-Fi standard.

Appendix B
Constraints

CrowdVision faces many constraints, including ergonomic difficulties, legal considerations, and policy and regulatory issues.

Since CrowdVision is meant to be mounted in a room in order to gather images, it is important to design the device in a way that would allow it to be mounted with no damage to the device itself. To accomplish this, we purchased a case for our Pi that will allow it to be mounted with ease. Since the actual PCB and ports are not externally exposed, we should be able to mount the Pi by any means necessary, while mitigating damage to the device.

In addition, CrowdVision's application is in university environment, meaning that one of our responsibilities and constraints is to abide by UCI's public filming policies. The UC Irvine Administrative Policy Sec. 900-30: Policy on Filming and Photography on the UC Irvine Campus states that, "filming and photography are permitted on the UC Irvine campus and on property leased by the campus" [6]. While we are allowed to film on campus, there are some constraints for legal considerations, as well as policy and regulatory issues.

CrowdVision is restricted in the placement of our project by the university's public filming policy. We must ensure that our project "does not interfere with regular educational, research or outreach functions or previously scheduled events of the University and does not pose a security, health or safety risk" [6]. This constraint deals with ergonomic difficulties since we must ensure that our project does not interfere with an individual's normal activity. Our current workaround to this constraint is placing our Raspberry Pi v3 module and camera in an imperceptible area of a study room to minimize the possible distraction it may cause.

APPENDIX C
HARDWARE AND SOFTWARE SECURITY

Security was a key consideration for our project, as it is mostly internet-based, which introduces greater risks. In order to minimize the amount of personal information that is stored on our servers, we do not store any of the images that are processed by CrowdVision. Once the image has been analyzed and the number of people has been extracted, the image is immediately deleted. CrowdVision never stores images in the cloud. It only stores numbers, which is extracted processed data. In addition, since our project is set up in a modular manner, there is a minimal amount of information that is stored at any one location. As such, even if our web page is compromised, our server will still stay secure, and vice versa.

In the future, there are other steps we can take to ensure that our project is even less vulnerable to hacking or loss of personal information. For example, when sending the image over the internet, we can employ an encryption scheme to ensure that even if the image is intercepted, none of the information will be compromised. Another option for greater security is to blur out the faces of individuals locally before we send the image to the server. However, for this option we would have to run tests to ensure that this will not make our image processing less accurate. If blurring faces is possible, it will add another layer of security, ensuring that even if the image is intercepted and decoded, the individuals will not be able to be identified by the hacker.

ACKNOWLEDGMENT

This project was funded from the University of California, Irvine senior design course (EECS 159A) which is an ABET approved course. CrowdVision would like to thank and acknowledge Professor Stuart Kleinfelder for advising this project throughout the quarter.

REFERENCES

[1] Morerio, Pietro, et al. *People Count Estimation In Small Crowds.* 2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance, 2012, doi:10.1109/avss.2012.88.

[2] J.W. Choi, X. Quan, and S.H. Cho, *Bi-Directional Passing People Counting System Based on IR-UWB Radar Sensors*, IEEE Internet of Things Journal, vol. 5, no. 2, pp. 512–522, 2018.

[3] A. Krizhevsky, I. Sutskever, G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, Advances in neural information processing systems, 2012.

[4] Yuxin Wu, Alexander Kirillov Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. *Detectron2.* https://github.com/facebookresearch/detectron2

[5] K. He, G. Gkioxari, P. Dollar, R. Girshick. *Mask R-CNN.* CoRR, 2017.

[6] *Sec. 900-30: Policy on Filming and Photography on the UC Irvine Campus: Policies & Procedures: UCI*, Policies & Procedures. [Online]. Available: http://www.policies.uci.edu/policies/pols/900-30.php. [Accessed: 14-Nov-2019].