People Counter: Count, Estimate, and Predict Occupancy of Rooms in Hallway

David Feng, Tianzhuo Li, Gary Qin
Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract* — A system capable of estimating and predicting the occupancy of lab spaces at CMU's Hamerschlag 1300 wing. People Counter helps users save travel time by presenting a real-time estimation of room occupancy and an interactive element that predicts ahead of time whether the spaces will be busy. The system combines video feed processing, computer vision, as well as a web application deployed on server; it will be more than 80% accurate for both estimation and prediction to deliver up-to-date occupancy levels and useful predicted data for users.**

***Index Terms* — ArduCAM, classification, computer vision, interactive web application, object detection, prediction, room occupancy, video processing**

I. Introduction

Several travel-related applications such as Google Maps and the New York MTA TrainTime mobile app have evolved in recent years to introduce real-time capacity tracking and estimation features. However, these applications currently do not combine the usage of historical data and real-time video processing techniques to make accurate predictions. Google Maps' prediction algorithm relies on users' historical location data [1], while the TrainTime mobile application is only able to show real-time occupancy data of the train cars [2] and does not have the capability to let users know whether their train is going to be packed ahead of time.

Thus, we propose People Counter, a system that uses both live camera feed to CV processing as well as analysis of historical data to count, estimate, and – most importantly – predict the occupancy levels of the Hamerschlag Hall 1300 hallway at Carnegie Mellon University. The system is an amalgamation of computer software and digital hardware technologies, as the live video feed will be captured by ArduCAM B0205 cameras and fed into the local backend through wireless connection, then the backend program will process the data through SQLite as well as computer vision algorithms such as Yolo and SORT for object detection, and lastly the occupancy and prediction data will be sent to the frontend web user interface.

People Counter's intended users will be students enrolled at Carnegie Mellon University who would like to know about the current and predicted occupancy data of the 1300 hallway at Hamerschlag, which is among the most popular study spots for students in the Electrical & Computer Engineering department. Through our observation over the past several years of students who enter the hallway, many students would struggle to find a seat when the labs are in session or when the rooms are busy. They tend to quickly exit when no seats are available. Over time, the time wasted walking to and from study spaces like the 1300 hallway has become a major pain point for students. That is why we believe People Counter has the capability to solve this user pain point. By interacting with People Counter's web application, users will not only be able to know the accurate current occupancy data of the Hamerschlag 1300 hallway but also obtain the predicted occupancy of the two individual lab spaces in the Hamerschlag 1300 wing at a time of the user's choosing within the same day. With this information, the users will be able to make a better-informed decision about whether they would like to go study at the Hamerschlag labs. The project's ultimate goal is for our users to save valuable time during their day that they would otherwise spend on traveling to and from Hamerschlag without the data from People Counter, only to find a hallway packed with studious individuals at times.

II. Use-Case Requirements

We formulated the use-case requirements for People Counter based on the needs of our intended users. One of the critical use-case requirements for our system is to have a greater than 80% occupancy estimation accuracy for the two enclosed lab spaces on the Hamerschlag 1300 wing. The reason behind the 80% benchmark is due to the fact that the larger of the two lab rooms has a capacity of 50 people, while the smaller one only has a maximum occupancy of 25. The 80% benchmark would allow for a maximum of 5-people margin for error in the smaller lab space which would suffice the needs for our target users, as a difference in occupancy of 2 or 3 individuals would in general not affect a user's decision of whether or not they would want to spend time studying at the 1300 wing. However, if the estimation accuracy metric is set at a much lower mark, such as 50%, then it would not be sufficient for our use case, because the difference between 10 and 20 people in a room with 25 seats in total is significant for users who hope to find a less busy, relatively quiet place to do work. Additionally, the 80% benchmark must be reached consistently for at least 15 hours, which is roughly the amount

of time that the 1300 wing at Hamerschlag will see people movement during a normal day when the university and Hamerschlag Hall is not closed.

Other than real-time estimation, another key feature of the People Counter system is same-day occupancy prediction for the two lab rooms on the Hamerschlag 1300 wing. We plan to implement our prediction feature through categorization, characterizing occupancy levels of the rooms into 4 categories: “almost empty” (up to 19.9% full), “not busy” (20%-39.9% full), “busy” (40%-69.9% full), and “almost full” (70% full or more). When a user selects a same-day time on the web application, the interface will return one of the four categories listed above, the percentage range of each category, as well as the total capacity of the room. The aforementioned categorization and the percentage of capacity associated with each category are not set arbitrarily. Both lab rooms on the 1300 wings at Hamerschlag do not have individual, separate seating areas but rather feature contiguous work benches intended for better collaboration. If there are more than 70% of the total seats taken in one of the lab spaces, this would mean that the room is effectively almost full, because there will rarely be two consecutive occupied seats (having both neighboring seats taken is less ideal for individual studying) [3] when total occupancy is above 70%. On the other hand, when the occupancy levels of a room are below 40%, then it is guaranteed that there will be at least 1 seat in the room where there is at least 1 unoccupied neighboring seat. This would warrant a “not busy” categorization. Similar to the live estimation benchmark, we set the prediction accuracy to 80%, which means that when a user selects any time later during a particular day, the predicted occupancy categorization must exactly match the actual percentage of the rooms at the chosen time more than 80% of the time over a 15-hour period.

Lastly, another use-case requirement that, if not achieved, would impact the usability of our application is latency. Ideally, the latency of our system shall be less than 1 minute (60 seconds) at any given time when the server of People Counter’s web application is running. Given the nature of people movement (moving in and out of a room on foot takes at least several seconds), we do not expect any non-malicious user to be checking People Counter’s web application in an interval less than 60 seconds for the up-to-date information regarding real-time occupancy estimation. However, given that occupancy in a room, especially multi-use spaces like the Hamerschlag 1300 labs, can shift swiftly (such as the start and end of class sessions), it could potentially be beneficial to users if the total latency of our system is consistently kept at under 60 seconds.

III. Architecture and/or Principle of Operation

Our system consists of three main subsystems: a hardware module running on a Raspberry Pi controller, a backend CV module running on our personal computer, and a UI module. Fig. 1 shows a high-level design of each of our subsystems. Our hardware module consists of a Raspberry Pi connected to an ArduCAM B0205 camera, which will send live video feed to our backend through WIFI. We will place the camera in the entrance of the Hamerschlag 1300 wing entrance, looking down the hallway in Fig. 2.

The camera looks down the hallway at a slight angle, capturing all the doors in the hallway. The camera feed then gets sent to our backend CV pipeline. In our CV module, we will first preprocess the frames by identifying the areas surrounding each of the doors in the hallway and use those as our regions of interest. We will run Yolov5s on each of the frames and identify people in the hallway. The people detected in the video feed will be tracked with DeepSORT tracking algorithm, and we will interpret the tracks of each person to create a heatmap for each of the rooms in Hamerschlag 1300 wing. The interpreted data will be passed through a decision tree model to predict the future occupancies of each of the rooms (we will use the data we collect during testing as a foundation and create a dataset to train the model during the development of our project). Eventually, the interpreted data and prediction results given the interpreted data gets sent to our UI to be displayed on a web application.

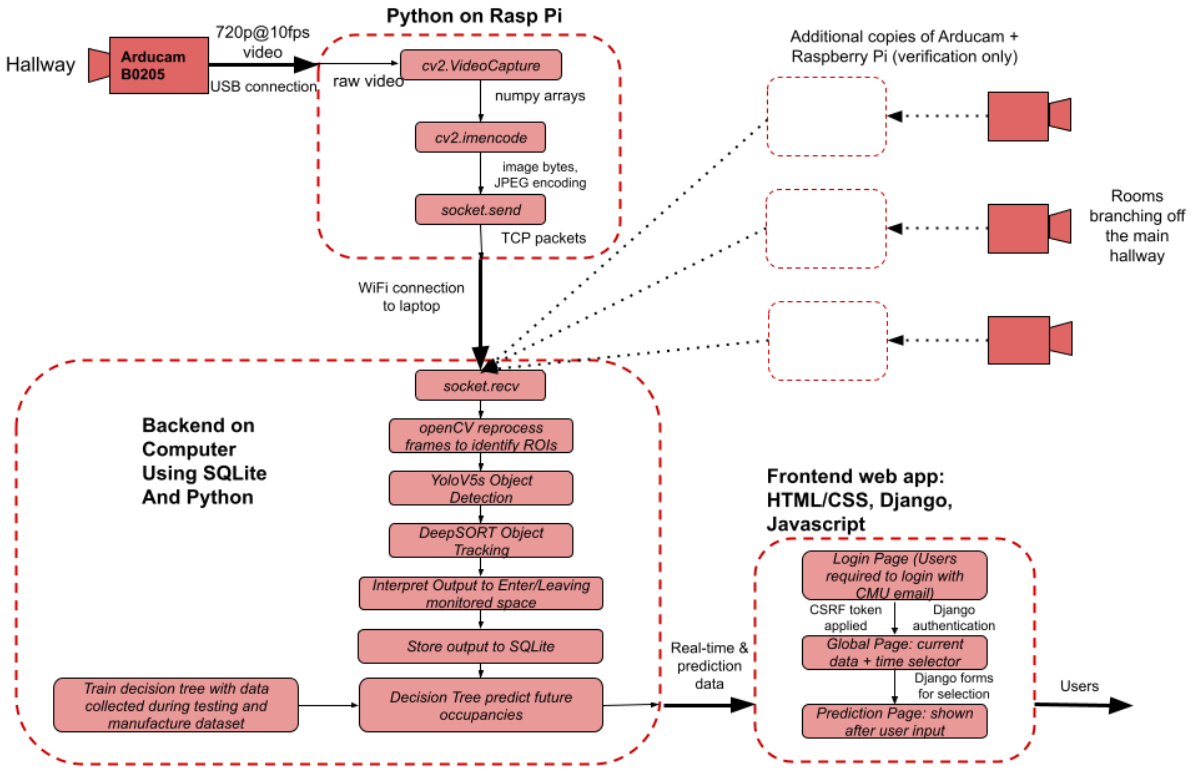

Fig. 1. Overall system block diagram

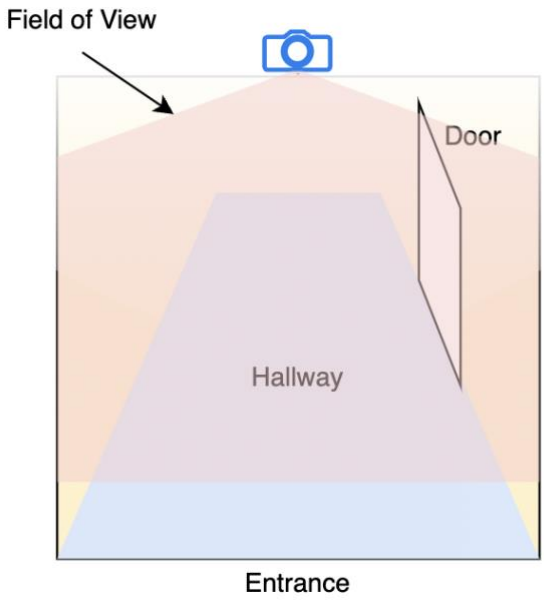

Fig. 2. Camera setup in Hamerschlag Hall 1300 Wing

IV. Design Requirements

The primary goal of our design requirements is to ensure that the system we develop meets the estimation and prediction accuracy as well as the overall latency requirements previously mentioned for our use case of the Hamerschlag 1300 hallway.

A. Location of Cameras

To image the Hamerschlag corridor completely, we will utilize a vision system that comprises a single camera placed directly above the main entrance way of the hallway. The camera will face the main hallway, with a slightly angled downward view. The purpose of positioning the camera at this location is to capture a single image that encompasses all room entrances, enabling our computer vision systems to receive as much information as possible. This approach aims to maximize the accuracy of our estimation and prediction systems though this maximized coverage.

B. Camera Resolution

We have determined that the minimum resolution necessary for our computer vision algorithms to accurately estimate occupancy is 1280 by 720p. To arrive at this conclusion, we captured sample images at various resolutions using the ArduCAM camera, from the angle at which we plan to mount the camera during testing. We then utilized our Yolov5 detection algorithm to evaluate whether people were correctly detected in the varying resolution images. We observed that the accuracy of our detection algorithm began to decrease below a resolution of 720p, which led us to select this resolution for our computer vision pipeline. We also decided against using a higher resolution in order to reduce the Wi-Fi bandwidth requirements between the Raspberry Pi and the laptop running our CV system, so that we can consistently meet our latency requirement of 60 seconds.

C. Camera Frame Rate

To minimize the burden on our computer vision pipeline, we have decided to transmit our live video feed at a frame rate of 10 frames per second. If the frame rate were too high, it would be difficult for our local computation of the object detection and tracking on our laptop to keep up with the incoming frames. We believe that a frame rate of 10 fps is sufficient for our use case of the Hamerschlag 1300 hallway, as this is an indoor environment where pedestrians are unlikely to run at high speeds in a narrow hallway. We have determined that 10 frames per second is the minimum required frame rate for our vision pipeline to accurately estimate occupancy, as it is a low framerate where our DeepSORT object tracking algorithm can still follow walking people between frames, in order to meet our requirement of 80% estimation accuracy.

V. Design Trade Studies

During the design phase of our project, we took into account the implementation complexity, the fulfillment of our use-case and design requirements, and the practicality for our system when making design decisions. Our aim was to strike a balance between these factors with our design choices.

A. Number of Cameras

Our product's imaging system is designed with only a single camera, and we believe that adding additional camera angles would provide limited imaging benefits due to two primary reasons. Firstly, the primary camera is already capable of covering all doorways in the Hamerschlag hallways. Therefore, additional camera angles would offer minimal value in additional information. Secondly, the incorporation of more cameras would require the development of another subsystem to synchronize the incoming video feeds from multiple variable Wi-Fi connections. Besides, there would also be a need for additional measures to match detected individuals across the various video feeds, to ensure that our estimation system does not double count pedestrians. This would put significant pressure on not only our development process, but also our computer vision system itself, potentially leading to latency issues. Thus, we have decided to limit our design to just one camera, ensuring that we can still meet our use-case requirements without putting undue strain on our system.

B. Another Design Specification or Subsystem

We selected the ArduCAM B0205 as the camera model for our imaging system. The specific camera model used in the imaging system is not crucial, as long as it fulfills our design requirements of covering the vertical span of the Hamerschlag hallway and exporting video at a minimum resolution of 1280 x 720p and a framerate of 10 fps. We ultimately opted for the ArduCAM B0205 since it met these design requirements and had a small form factor that made it easy to mount during testing. Additionally, it could connect via USB to the Raspberry Pi, simplifying integration of the camera and the microcontroller.

C. Microcontroller

When it came to choosing a microcontroller for our imaging system, we weighed up the options of the Raspberry Pi and Jetson Nano. Ultimately, we opted for the Raspberry Pi due to several reasons. Firstly, we felt that the extra capabilities provided by the GPU on the Jetson Nano would be unnecessary for our imaging system. The main function of the microcontroller in our design is to transmit live video from a connected camera to a local laptop over Wi-Fi, where the actual computer vision algorithms are executed. The video processing done on the microcontroller is minimal, involving only the division of the video feed into TCP packets, thus negating the need for the processing capabilities of a GPU. Furthermore, the design requirements of our video pipeline are relatively low,

with only a 720p resolution and a 10fps framerate, further reducing the processing requirements of the microcontroller. Lastly, our team members were more familiar with the Raspberry Pi, having used the microcontroller before in other projects. As such, continuing to use the Raspberry Pi for our project would significantly reduce implementation time for our imaging system. For all these reasons, we then decided to select the Raspberry Pi for our microcontroller.

D. Figure and Table Formatting

During the selection of object detection algorithms, we researched about different methods such as fast RCNN and SSNMobileNet as well as the YOLOv5 model we selected. Based on our research, we found that YOLOv5s offered a good balance between inference speed and accuracy. See references [5][8] for detailed comparison. Through initial local testing, YOLOv5s looks like a promising model to meet the latency requirements of 10 frames per second (FPS), while delivering accurate multiple object detection. During our initial testing on pictures of people in Hamerschlag 1300 wing hallway, we found that the YOLOv5s model was able to detect and accurately draw bounding boxes around people in the hallway with high accuracy (around 90 percent confidence rate for people that is not colluded, and around 60 percent confidence rate for people that are partially colluded).

E. Object Tracking Algorithm

We decided to choose DeepSORT tracking algorithm due to its ability to handle occlusion of objects and prevention of re-id of tracked objects by running feature extraction of tracked person [8], as well as the multiple-object tracking capabilities, as we expect to track multiple people walking in the hallway. Having an accurate tracking algorithm is very important to achieving our accuracy requirement of 80% prediction accuracy. This is because if our tracker cannot track each person appearing in the video with high accuracy, there would be compounding error when we try to interpret the results of our CV system and result in compounding error in our estimations. Therefore, we have chosen to use DeepSORT as our object tracking algorithm. However, one drawback is that in order to perform feature matching on tracked objects, DeepSORT runs a pretrained CNN, which could lead to increasing computational complexity, potentially leading to failing our latency requirements. If we run into latency issues with DeepSORT, we will experiment with other methods such as the MedianFlow tracking algorithm, which is a Lucas-Kanade based algorithm that would run much faster than DeepSORT as it does not involve running a CNN.

F. Location of Computer Vision Processing

During our design phase for the project, we explored the possibility of performing object detection and tracking on the cloud. However, after careful deliberation, we decided against this option. Although moving the computer vision algorithms to the cloud would have alleviated the computational load concerns of the CV pipeline, it would have raised additional concerns about the latency and reliability of our cloud services. In the event that our cloud provider experiences performance issues during testing, it could result in the computer vision pipeline exceeding our 60s latency design requirement or failing completely. Therefore, we concluded that it would be more prudent to keep the computation of our computer vision algorithms on local hardware to ensure the reliability and latency of our system.

In particular, the Google Cloud Platform, which is covered by the CMU budget, was found to be lacking in modules related to video processing on the cloud and proved to be difficult to access video feeds during testing. Furthermore, funding was not available to deploy the computer vision system to alternatives such as AWS. As a result, we decided against performing object detection and tracking on the cloud, and instead opted for local computation. Despite the more computationally intensive requirements of our Yolov5 and DeepSORT algorithms, we believe that a local laptop will be able to handle the workload within our 60 second latency target, especially given our lowered design requirements in video resolution and framerate.

G. Web Interface Design

One of the emphasis for the design of the web application is letting logged-in users have access to the most up-to-date information. Therefore, instead of employing a simple mechanism in which users manually click refresh to have the main page updated, we have decided to use Ajax so that data can be sent and retrieved from the server asynchronously. This allows data to be delivered to the front-end through JSON, and users will not have to manually refresh their page to gain the most recent information regarding occupancy levels at the lab spaces on the 1300 wing.

VI. System Implementation

A. Hardware Configuration

Fig. 3 displays the architecture of our hardware components, mainly consisting of the ArduCAM camera module and our Raspberry Pi microcontroller. The configuration of the video stream from the camera will be done in by the Python program, with OpenCV setting the video connection to have a resolution of 1280 x 720p. OpenCV will also be responsible for controlling the framerate of the video pipeline, pulling new frames from the camera at a rate 10 frames per second.

After capturing the individual frames using Python, OpenCV encodes them into JPEG format before transmitting them to the Wi-Fi socket. The original raw format of the images stored by OpenCV is not suitable for transmitting over a standard Wi-Fi connection due to its storage inefficiency. Therefore, the images are converted into JPEG format to reduce the bandwidth requirements between the Raspberry Pi and the receiver laptop. The conversion of video into

individual images also prepares the data for transfer via socket. The images are transmitted sequentially over the Wi-Fi socket and are reassembled in the same order on the receiving end.

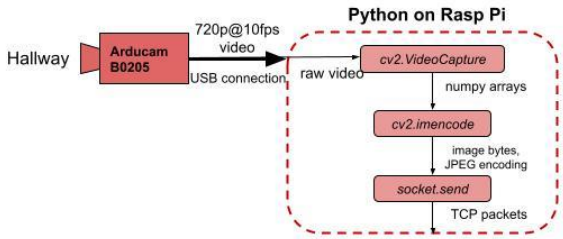

Fig. 3. Hardware (camera) configuration and architecture

B. Backend Processing & Computer Vision

The backend processing and CV module could be separated into several subsystems: Object Detection and Tracking, Interpreting Outputs, and predicting the future occupancy levels.

Object Detection and Tracking:

Our object detection and tracking subsystem will be implemented on a personal computer in python. The video feed gets received through WIFI through python's built-in socket interface. We will manually inspect the camera feed and expand bounding boxes around each of the entrances in the Hamerschlag 1300 wing hallway to define ROIs using OpenCV and python. The ROIs will be defined as a rectangular box that starts at the bottom of each entrance door and expands into the middle of the hallway.

For object detection, we will run pretrained YOLOv5s model to perform object detection. The model is pretrained on the "COCO" dataset. The algorithm will return bounding boxes for each of the object detected, and we will select only the bounding boxes identified as "person" We will match bounding boxes with the ROIs, if the bounding boxes are within the ROIs, we will pass these bounding boxes into our tracker. During initial local testing on a M2 MacBook pro with 720p input video feed, we were able to get around 9 FPS, which is slightly below our 10FPS requirement. We will try to improve YOLOv5s performance through pruning as suggested in [6] and by using half precision FP16 inference as suggested in [7].

Object tracking with DeepSORT will be done in conjunction with the object detection algorithm. The tracking algorithm will take the detection outputs from YOLOv5s and create tracks for each of the bounding boxes it identifies as unique. If the bounding box sent to the detection algorithm is similar to one of the previous tracks, it will be used to update that track. Else, the tracking algorithm will predict the future position of the track to continue tracking the object. This allows for handling occlusion of an object and cases where an object does not get detected in a given frame. For each of the tracks, we can extract the bounding boxes of the objects in the given track and a unique id for the track.

Interpreting Outputs:

For each of the entrances, we will define a range of x and y coordinates that identifies the bottom of the door. For instance, if there is a door at the top left corner of an image, and points on the image are represented with coordinates x (in range 0 to image_width starting on 0 from the left), and y (in range 0 to image_height starting on 0 from the top), then we will model the bottom of the door as line from (x1, y1) representing one end of the line, and (x2, y2) as the other end of the line. We will also maintain a dictionary of unique track_ids and their corresponding movements, done by comparing coordinates bounding boxes between the past several frames. In a given frame, if we detect that the bounding box coordinates of a tracked object crosses the line of the entrance, we will check the movement of that track_id stored in our dictionary. If the movement of the track is away from the center of the image (assuming that the Hamerschlag 1300 wing hallway is placed in the middle of the image, and doors are on the side of the hallway as shown in Fig.2 on page 2), then we can say that the person is entering the room, and we would increase the count for the room. Whereas if the movement of the person across a given number of tracked frames is towards the center of the image, and away from the line that indicates the entrance to a door. We will treat that as a person exiting the room. For cases where a person appears in the region surrounding the line that defines a door, if we do not have historical data associated with the person, we will treat that as a person leaving the given room as well, since a person appearing at the door without previous detection is likely to be out of the field of vision of the camera to start with (meaning that the person was not in the hallway hence not entering the room).

Prediction:

We will be training a decision tree classifier in order to predict the future capacity. To train the decision tree, we will collect testing data for the capacity of the rooms in Hamerschlag 1300 wing at different times of the day and on different days of the week. The collected data will be used as a reference point to build a dataset manually. To train the decision tree, we will label existing data based on features such as time of the day, weekend vs. weekday, if there is class currently being held in the space, capacity level in the room during previous hours, etc. We plan on using the ID3 algorithm to build the decision tree and split on the attribute that gives the most information gain. We plan on creating 100 data points for training our model and will tune the decision tree using cross validation to get the optimal depth. The trained tree will be stored in the backend and could be used to make predictions of future occupancy levels in almost real-time, since the prediction only requires labeling the data and traversing down the trained decision tree to make a prediction. This would allow us to meet the latency requirements for our

system, where the user could get updated data every 60 seconds.

C. *Interactive Web Application*

The web application will provide a secure, usable, and interactive platform for users to view and interact with the data that is processed by our backend database. The computed data output from the backend will be integrated with the frontend using Python as well as the Django framework so that the data can be deployed on the server. The basic user interface and frontend styling for the web application is done on HTML/CSS; we will also be using Ajax and JavaScript frameworks to create an asynchronous web application and ensure the main page will be able to automatically update without having to manually refresh the page for updated information.

For the login page, the web application will employ the Django authentication package and customized decorators to allow new users to register and old users to login. It will protect the system against malicious users by ensuring that all users will need to register using an Andrew email address provided by CMU. After a user logs in, they will be directed to a home page which shows real-time information regarding occupancy at the Hamerschlag 1300 wing. There will also be a form and a submit button on the home page that allows users to interact with the web application by selecting a same-day time for a predicted category of occupancy levels. The form will be done using Django's Model-View-Controller (MVC) system architecture and the relevant data will be transmitted and constantly updated over 60-second intervals through JSON. Fig. 4 details the web application's user experience from registering to getting the occupancy prediction, while Fig. 5 demonstrates the home page user interface upon logging in.

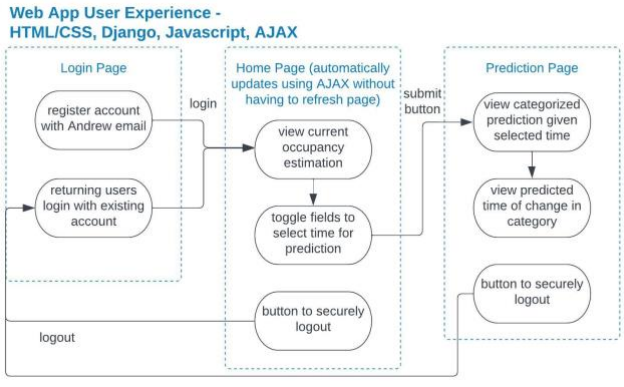

Fig. 4. Web application user experience across all pages

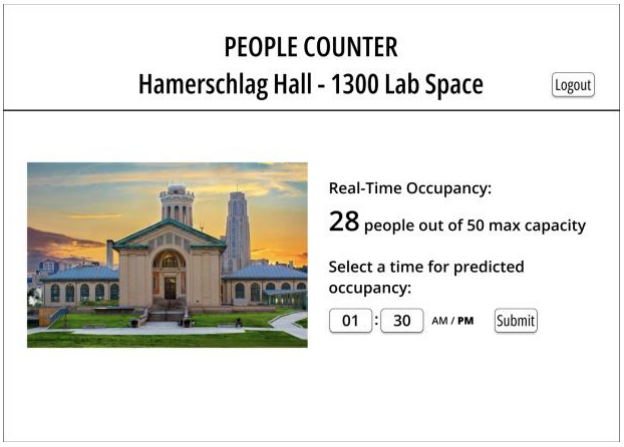

Fig. 5. Web application home page wireframes

VII. Test, Verification and Validation

The testing and verification of our system will be done in three main phases.

A. *Tests on Pre-recorded Footage*

To commence the testing process, we will first use pre-recorded footage. The Raspberry Pi-based vision system, along with ArduCAM, will be installed on top of the Hamerschlag 1300 hallway doorway, and the camera will be set to record for a maximum of one hour. The recorded footage will be saved on an external storage device that is plugged into the Raspberry Pi. In order to replicate our future testing environment, we will capture the initial footage as a sequence of 640p hallway videos at a 10-frame-per-second rate.

Once the recording is complete, we will retrieve the video from the storage device and input it into our computer vision system. The system will then attempt to estimate the occupancy heatmap based on the recorded video. The central aim of this test will be to observe if our computer vision algorithms can suitably detect and track pedestrians inside our testing environment, the Hamerschlag hallways.

Since our initial testing phase will take place early in the development of our computer vision and camera systems, we will conduct this stage of testing for a much shorter duration than our final requirement. We will limit our pre-recorded video to a maximum of one hour to allow for easy manual verification of occupancy levels and quick retesting of new developments. Therefore, during this phase of testing, we will not be evaluating our prediction model, but solely focusing on verifying the estimation system. Our initial test's targeted accuracy for each room in the estimation system is 80%. After we have verified that the estimated occupancy in each room remains within our 80% threshold for the entirety of the recording, we will move on to the next phase of testing.

B. Tests on Live Footage

Similar to the first stage of testing, the vision system, which includes the ArduCAM and the Raspberry Pi microcontroller, will be installed in the Hamerschlag narrow hallway. The system will transmit a live video feed of the hallway, which will be analyzed by the computer vision algorithm on a laptop. The video data will be transferred wirelessly between the Raspberry Pi source and the laptop through a Wi-Fi socket interface.

In the second testing phase, we plan to simulate the typical activity duration of the Hamerschlag environment by conducting tests for up to 15 hours at a stretch. However, due to the extended duration, it won't be feasible to manually verify occupancy levels throughout the entire testing period. Instead, we will evaluate the accuracy of our estimation algorithms by comparing the occupancy estimates generated by them with the readings obtained from other cameras installed in the individual rooms branching off from the Hamerschlag hallway. The cameras will be placed in the upper corners of the rooms to capture a complete view of the space. Each camera is connected to a Raspberry Pi that transmits the video feed over Wi-Fi to our primary processing laptop. On the laptop, a basic detection algorithm will run to determine the current number of individuals in the room, providing a basis for comparison.

To simplify matters, we will limit the placement of our verification camera systems to a certain number of rooms that are connected to the hallway. We anticipate that the occupancy patterns in each of these rooms will be relatively similar to each other, so if our estimation algorithm can accurately count the occupancy in one part of the Hamerschlag rooms, it should be able to do so for the others. Additionally, we will select camera locations that account for as much variation and challenges for our computer vision algorithm as possible. As such, we plan to install cameras in the meeting room closest to the hallway entrance, the meeting room furthest from it, and one of the larger lab spaces. Our goal for this testing phase is to ensure that our primary computer vision system can estimate occupancy within 80% accuracy of the occupancy detected by the 3 auxiliary cameras throughout the testing period.

During this phase of testing, our main priority is to ensure that our computer vision pipeline can maintain consistent performance that matches the speed of the live video feed. If our system is unable to do so, we will need to make any necessary adjustments to our system immediately, as each round of testing is very time-intensive. In addition, we will evaluate the connection between the Raspberry Pi and the laptop, as well as the ability of our pipeline to operate continuously over an extended period of time. Another objective during this phase will be to collect training data for our predictive module. We will record occupancy levels from the auxiliary cameras during our testing sessions and store them in a database for future use in building our decision tree model. Therefore, we will not yet test the performance of our predictive module during this phase until it has been appropriately trained and developed with the data we collect. Another objective at this development stage is to have the web application operational, which will display the estimated count of individuals in each of the branching rooms to users. We will observe whether the website maintains its responsiveness to the occupancy changes in the surroundings by observing it at the start and end of the testing runs. Our validation target is to ensure that the system operates within 60 seconds of latency.

C. Tests with predictive module

During this phase, we will begin testing on our prediction component. The testing setup will remain the same as the previous phase, where we will install the ArduCAM and Raspberry Pi microcontroller to face the Hamerschlag narrow hallway. The system will send a live video feed of the hallway from the Raspberry Pi over Wi-Fi, which will be analyzed by the computer vision algorithm on a laptop. We will also place our verification camera in the same three rooms as in the previous phase and continue running detection algorithms to ensure that we maintain our estimation target of 80% throughout the testing period. In addition, we will continue to observe the latency of our web application, in order to maintain our 60 second latency target.

At the beginning of each hour of the testing process, the algorithm will also predict which of the four capacity categories each room will fall into one hour in the future. At the start of the following hour, this prediction will be compared to the occupancy detected by the auxiliary camera setup in the rooms. Our validation target for the predictive model during this testing phase is to accurately predict the occupancy categories of the three rooms with auxiliary cameras, one hour into the future, at an 80% success rate throughout the testing process.

VIII. Project Management

A. Schedule

The detailed schedule for the project is presented at the second to last page of the document. Up to the submission of this report, the team has already completed the purchase of required hardware, explored the computer vision libraries, tested possible deployment methods, and created the wireframes and initial prototype for the web application that users will be able to interact with. We plan to integrate the backend estimation data with the local web application on Week 10 (March 20-26), and we plan to have the full integration of video feed, backend software, and web application for the estimation feature before the interim demo on Week 12. We will use the time after the interim demo to conduct extensive testing of our system over an extended period of time (15 hours continuously), and the full integration of the entire system including our prediction feature will take

place on Week 13 (April 10-16). We plan to have our entire system tested, validated, and integrated by Friday, April 21, 2023, at the latest.

B. Team Member Responsibilities

David Feng is working on configuring the camera modules with the Raspberry Pi 4, calibrating the camera configuration to the testing environment of Hamerschlag Hall 1300 wing, and connecting the video feed to the backend software through wireless internet.

Tianzhuo Li is in charge of developing and running the CV-based algorithm for object detection. Additionally, he will work on translating the CV output to estimation data and implementing the prediction algorithm with machine learning models in the backend.

Gary Qin is responsible for connecting backend data output to the local frontend as well as the server-side web application powered by Django. In addition, he will be working on developing the entire web application user interface using HTML/CSS.

C. Bill of Materials and Budget

The bill of materials used for this project and the budget allocation is displayed in detail in Table 1 at the last page of this document.

D. Risk Mitigation Plans

The team has identified 3 critical risk factors for this project: the speed and efficiency of the backend in processing live video feed, the potential of algorithmic bias being present in the computer vision algorithm, and the privacy concerns of people present in the camera frame and the users who interact with the web application.

We are committed to improving the performance of our system and optimizing the latency of individual subsystems. If, during early-stage testing of the object detection algorithm, the performance stats are way below the 10 frames per second (FPS) design requirement, we will switch to Jetson Nano controller with a GPU, which will increase the frame rate of our system based on previous experience working with GPUs.

During the testing and validation stage of our project, we will be observing the existence of any algorithmic bias present in the CV algorithm doing object detection. We will strive to eliminate any sort of algorithmic bias, including discrimination of race, gender, and height, that causes the algorithm to be incapable of detecting a person under various confidence levels. We will tweak the parameters of the CV algorithm so that we ultimately find no evidence of bias when testing and validation our integrated system.

We care about the privacy of students, staff, and faculty members whom the system will be interacting with one way or another. Therefore, we will not be storing any live camera feed in a database – the video will be sent to the software backend for processing only. We will also mount the camera in a way so that it does not intrude into the normal movement of people in the Hamerschlag 1300 wing.

IX. Related Work

Even though we believe that the estimation and prediction of usage and occupancy in enclosed spaces have a high demand and a variety of potential use cases, there does not seem to be a considerable number of existing applications or systems in the market. Specifically, we found no similar products currently being used on Carnegie Mellon's campus that help estimate and predict the occupancy of lab spaces and lecture rooms. Nevertheless, there exist systems and products that are slightly different from People Counter in both use cases and implementation methods.

Zensors, a spin-off company from CMU's School of Computer Science, uses existing surveillance cameras near airport security checkpoints to estimate the wait time needed for passengers to go through security. It uses computer vision algorithms deployed on AWS cloud to process live information captured on monitoring cameras, and the computed output is displayed on a big TV screen in front of the security lines at Pittsburgh International Airport that gives travelers a real-time estimation of the expected security wait times. To our best knowledge, in contrast to People Counter's prediction feature, Zensors' airport security product does not predict the wait time of security lines in advance and does not have an interactive web application that allows users to select a future time for wait time prediction.

In addition, the paper by Saralegui et al. describes a case study of using IoT sensors and energy consumption of rooms to predict the rooms' future usage [4]. The researchers utilize a variety of parameters and historical data of an IoT smart home system, namely thermal behavior, energy usage, and humidity, to monitor and predict room occupancy data. The results show the prediction accuracy maintaining at around an 79% clip across multiple tests. This study shows that there are more ways to estimate and predict occupancy in rooms, but the accuracy is somewhat limited given that certain people movement can be unpredictable.

X. Summary

In summary, People Counter's design leverages the combination of live camera feed, a backend that computes occupancy data with CV and machine learning algorithms, and an interactive web-based application to deliver both real-time estimation and future predictions of the occupancy levels at Hamerschlag Hall's 1300 lab spaces at CMU. On a compact college campus like that of Carnegie Mellon, it can be difficult for students to find an ideal place to do some work or relax. For them, it would be a huge waste of time and energy if they come to their favorite spot to study, like the Hamerschlag 1300 wing for ECE students, only to find a packed room.

People Counter can help resolve this issue, as students will have remote access to real-time occupancy data and will even be able to choose a time (same day) to view the predicted occupancy of the lab spaces. With the help from People Counter, students can now make a calculated decision with

better confidence on whether they would like to travel to Hamerschlag 1300 wing to spend their time, and thereafter they will be able to find more time doing what they truly enjoy, instead of wasting much of their valuable time on the road.

REFERENCES

[1] "Popular Times, wait times, and visit duration," *Google Business Profile Help*. [Online]. Available: https://support.google.com/business/answer/6263531?hl=en#:~:text=To%20determine%20popular%20times%2C%20wait,enough%20visits%20from%20these%20users. [Accessed: 03-Mar-2023].

[2] "MTA unveils new capacity tracking and real-time location features in Metro-North Traintime App," *MTA*. [Online]. Available: https://new.mta.info/press-release/mta-unveils-new-capacity-tracking-and-real-time-location-features-in-metro-north-traintime-app. [Accessed: 03-Mar-2023].

[3] L. Lim, M. Kim, J. Choi, and C. Zimring, "Seat-choosing behaviors and visibility," *JSTOR*. [Online]. Available: https://www.jstor.org/stable/26893773. [Accessed: 04-Mar-2023].

[4] U. Saralegui, M. Antón, O. Arbelaitz, and J. Muguerza, "Smart meeting room usage information and prediction by modelling occupancy profiles," *Sensors*, vol. 19, no. 2, Jan. 2019.

[5] M. G. Naftali, J. S. Sulistyawan, and K. Julian, "Comparison of Object Detection Algorithms for Street-level Objects," Aug. 2022.

[6] "YOLOv5 on CPUs: Sparsifying to Achieve GPU-Level Performance and a Smaller Footprint," *Neural Magic - Software-Delivered AI*, 07-Sep-2022. [Online]. Available: https://neuralmagic.com/blog/benchmark-yolov5-on-cpus-with-deepsparse/. [Accessed: 03-Mar-2023].

[7] Ahmed, Khaled R. *Smart Pothole Detection Using Deep Learning Based on Dilated Convolution*. https://www.researchgate.net/publication/357093620_Smart_Pothole_Detection_Using_Deep_Learning_Based_on_Dilated_Convolution.

[8] Wojke, Nicolai, et al. "Simple Online and Realtime Tracking with a Deep Association Metric." *ArXiv.org*, 21 Mar. 2017, https://arxiv.org/abs/1703.07402.

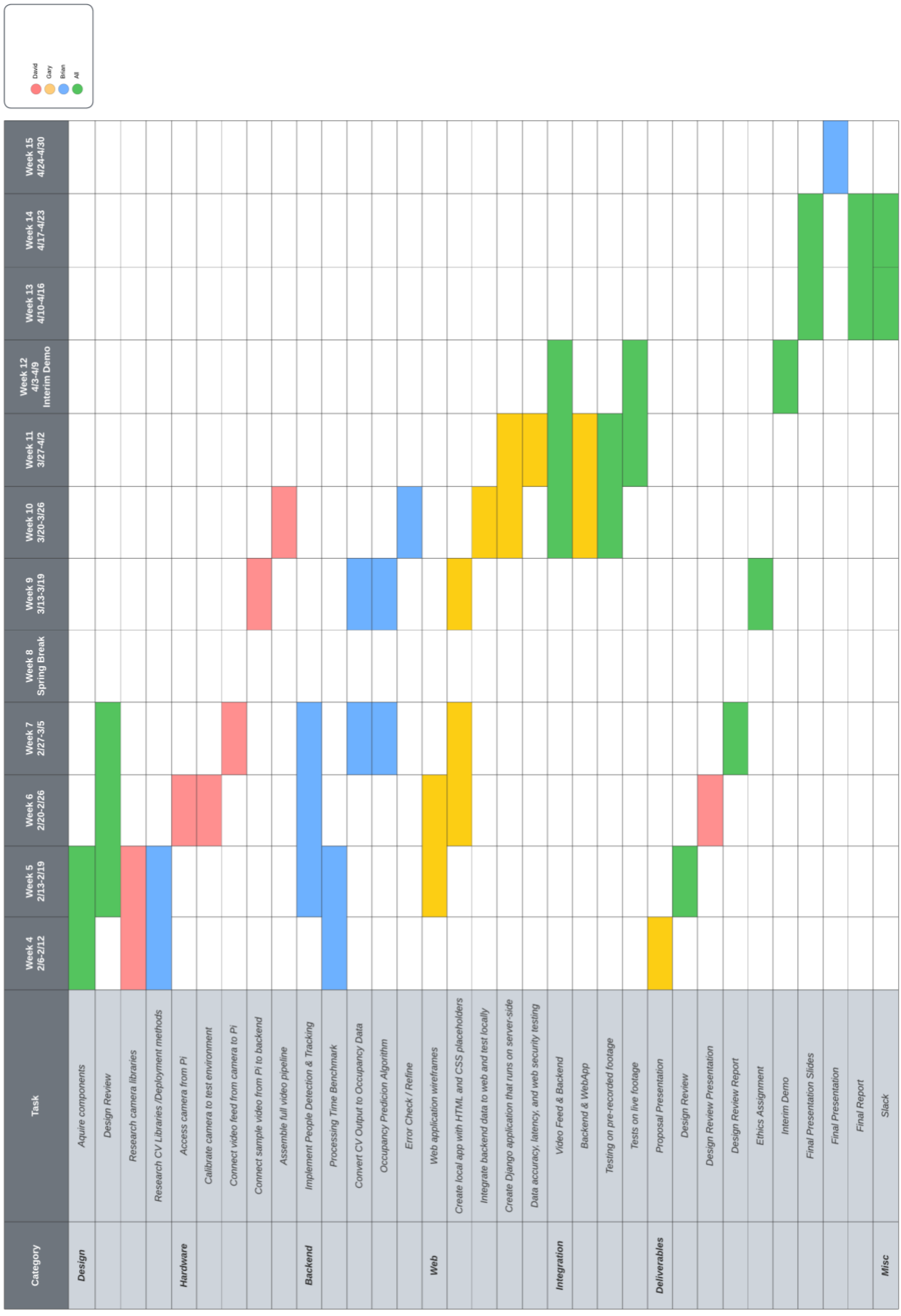
David
Gary
Brian
All
Category
Task
Week 4 2/6-2/12
Week 5 2/13-2/19
Week 6 2/20-2/26
Week 7 2/27-3/5
Week 8 Spring Break
Week 9 3/13-3/19
Week 10 3/20-3/26
Week 11 3/27-4/2
Week 12 4/3-4/9 Interim Demo
Week 13 4/10-4/16
Week 14 4/17-4/23
Week 15 4/24-4/30
Design
Aquire components
Design Review
Research camera libraries
Research CV Libraries /Deployment methods
Hardware
Access camera from Pi
Calibrate camera to test environment
Connect video feed from camera to Pi
Connect sample video from Pi to backend
Assemble full video pipeline
Backend
Implement People Detection & Tracking
Processing Time Benchmark
Convert CV Output to Occupancy Data
Occupancy Prediction Algorithm
Error Check / Refine
Web
Web application wireframes
Create local app with HTML and CSS placeholders
Integrate backend data to web and test locally
Create Django application that runs on server-side
Data accuracy, latency, and web security testing
Integration
Video Feed & Backend
Backend & WebApp
Testing on pre-recorded footage
Tests on live footage
Deliverables
Proposal Presentation
Design Review
Design Review Presentation
Design Review Report
Ethics Assignment
Interim Demo
Final Presentation Slides
Final Presentation
Final Report
Misc
Slack

TABLE I. BILL OF MATERIALS

Description	Model #	Manufacturer	Quantity	Cost @	Total
ArduCAM	B0205	Arducam	2	$34.99	$69.98
Raspberry Pi 4	4 Model B	Raspberry Pi	1	$0	$0
				Grand Total	**$69.98**