ENPH 454 Advanced Engineering Physics Design Project-Final Report

QueueHop: Computer Vision Wait Time Estimation for Clark Hall Pub

Date of submission: Dec 12, 2022

Instructor: Professor Jordan Morelli, P. Eng

Alex: Machine Learning Architect, Data Analyst

Bryn: Electrical Systems

Jonah: Machine Learning Architect, Cloud Specialist

Julia: Machine Learning Architect, Safety Officer

Griffin: Software and App Design

Alex Chase, Bryn Macduffee, Jonah Garmaise, Julia Everitt, Griffin Clark

Department of Physics, Engineering Physics & Astronomy, Queen's University, Kingston, ON, Canada

Abstract

Computer vision is a rapidly growing field that involves the use of machine learning algorithms to analyze visual data from images and videos. One application of computer vision is in crowd counting, which involves using algorithms to automatically estimate the number of people in a given area. In this project, we sought to use computer vision to determine the length of the line at Clark Hall Pub and estimate the wait time for individuals in line. Additionally, time of flight sensors were used to automatically count the pub's capacity. These metrics were displayed to users via a mobile application built using JavaScript and React Native, providing them with information to help them decide about their plans for Friday afternoon. The machine learning model achieved a 7.2% mean average error, the sensor capacity device achieved a 78% accuracy, and the mobile application achieved a frame rate of 60 frames per second. This provided the end user with a user-friendly experience and the ability to display live data. In future work, full integration of these subsystems is an area of focus.

Table of Contents

List of Figures

List of Tables

1.0 Introduction

Computer vision is a rapidly growing field that involves the use of machine learning algorithms to analyze visual data from images and videos. One of the key applications of computer vision is in the area of crowd counting, which involves using algorithms to automatically estimate the number of people in a given area. This technology has a wide range of potential applications, including in public safety, crowd control, retail, and transportation. One extension of crowd counting involves using the computer vision to count the number of people in line at a venue and extend this data to estimate the wait time for individuals in line.

For students on Queen's campus, one venue where individuals often find themselves waiting in long lines is at Clark Hall Pub. Here, students gather on Friday afternoons, and can often be faced with wait times upwards of an hour if the pub is at capacity. As such, the objective of this project is to use computer vision to determine the length of the line at Clark Hall Pub and estimate the time that students will wait when joining the line. Additionally, time of flight sensors will be used to automatically count the capacity of the pub over time. These metrics will be displayed to users live via a mobile application and will help the end user to make a sound decision about their plans for Friday afternoon.

To achieve this, the task was separated into 5 subsystems: the machine learning model, capacity sensor development, data pipeline, app design and data collection and analysis. Upon testing the final product, results found that the machine learning model achieved a 7.2% mean average error (MAE) when passed 500 unique images of students in line at Clark Hall. The sensor capacity device achieved a 78% accuracy during a 30-minute test period and the mobile application achieved a frame rate of 60 frames per second. This provided the end user with a sleek user-friendly experience and was also built with the capability to display live data using a Google Sheets API. A predictive wait time estimation model was built; however, it was not sufficiently tested with real data due to time constraints.

Although all subsystems were effectively developed and saw promising results, a key area for future work involves the full integration of these subsystems. Full integration would offer end users a fully functional and connected mobile experience and allow them to identify the length of the line and other key metrics from any location and at any time.

2.1 Motivation

Queen's students cannot know how long the line for Clark Hall Pub Ritual is before they arrive, and even then, have no means of knowing how long they will have to wait. The QueueHop machine learning model and Mobile Application aims to address this issue. Computer vision and object counting is not new to the machine learning world but commercializing the insights into a user-friendly app is where our project stands out. By developing a line counting model and displaying key capacity and wait time metrics live on a mobile application, users will be presented with all the necessary information to help them make a sound decision about their plans for Friday afternoon. In addition to Queen's students, who will be the primary users of the application, Clark Hall Pub, Queen's Security, and future potential clients are also key stakeholders in this project. In its final iteration, capacity and line data collected will prove invaluable to Clark Hall Pub. This information can help them to identify the busiest times of their service and could help with business problems including staffing, optimizing inventory, improving the experience of their patrons, and maximizing revenue. Apart from Clark Hall, Queen's

Security is also a key stakeholder as this project requires filming on Queen's Campus. As such, all regulation must be followed. Although this project is focused on Clark Hall Pub, the technology can be extended to different clients. Understanding wait times would be a beneficial addition to clients such as busy takeout restaurants, bars, airport security checks, professional sports games and other venues where large groups of people wait in line.

2.2 Project Scope and Goals

Quantifiable objectives are described in Table 1 below.

Table 1: Summary of quantifiable objectives

Objective	Target
Line length estimate accuracy	Crowd count accurate within 1 person or 10%, whichever is larger
Time to process image	1 minute per image
Overall Sensor Accuracy	Detect 80% of people
Speed	Measure individuals walking as fast as 4 km/h
Sensor Range	Detect people up to 2 m away
Wait Time Accuracy	20% error when capacity not reached
App Frame Rate	30 fps

Qualitative objectives include creating an aesthetic and user-friendly app to display information, and a physical device setup that does not get in the way of regular pub activities. Design constraints include storage required by the computer vision model as a smaller model will be more readily deployable, in addition to a time frame of one semester for completion, and the budget of $1000. The budget and timeline will likely make a complete outdoor security camera installation infeasible.

The scope of the project is to design the product for Clark Hall Pub as the client, however the technology could be extended to other venues, and this desire for generalizability will impact design decisions, particularly in app design. Due to time constraints, the model will only be tested for daytime use.

2.0 Design Decisions

The following contains decisions made concerning the design of the 5 elements of the project.

2.1 Machine Learning Model

The machine learning model must provide accurate results for both sparse and dense crowds as the length of the line at Clark Hall Pub changes throughout the day. Two machine learning frameworks were initially considered: a density-based approach, and a density and regression-based approach. It was decided to use the density and regression-based approach for the final design due to superior performance on variable crowd sizes. This will be discussed in further detail in Section 3.1.

Additionally, to accurately count the crowd size it was important to consider how a camera could be set up with a full view of the line. An overhead angle provides the best vantage point and allows for the capture as many people as possible in one frame. Multiple cameras or the rotation of one camera would also be a suitable way to capture a wider view of the line. The frames could then be combined before being run through the model to ensure the entire crowd is considered. It was decided that the minimum viable product (MVP) would be a single camera view for proof of concept, and a stretch goal would be the integration of two cameras for a complete crowd count.

2.2 Capacity Sensor

The capacity sensor must provide an accurate reading of people flowing in and out of the building. This means that the device must be always online and consistently updating and monitoring to catch someone passing by within a matter of a few seconds. There are various ways to achieve this, so the team evaluated three main options before pursuing the decided method.

An extension of the machine learning crowd counting model was proposed first, with an additional camera placed inside the building to monitor the capacity of the pub, and thus the change in capacity. This idea was promising, but upon further consideration, it was decided that the overall cost of an additional camera and setup, as well as the difficulty of capturing a full image of all rooms and people in the pub would be too difficult.

The second proposed design involved setting up two laser tripwires oriented horizontally near the entrance. This would involve using two laser pointers on one wall being aligned to connect with two photoresistors on the opposite wall. A change in the photoresistors voltage input to an Arduino would indicate a trip has been detected, and the order of trips would indicate whether the person was travelling in or out of the building. Though this was a more economically feasible method, the difficulty of aligning lasers to small photoresistors, made the design less versatile in its mobility around different possible building layouts.

The final design proposed was to utilize two Time of Flight sensors (ToF) oriented horizontally to catch those walking by. A ToF sensor uses a light pulse to determine the distance away from an object in front of it. Using a similar program architecture to the second design, where a change in distance readings from a sensor indicates a trip has been detected, and the order of trips would indicate the direction of travel. This design was determined to have a similar cost to the second design, while offering more versatility with its ability to be placed on any wall and be grounded to a base distance of an opposing wall or no wall at all. Because of this, the ToF design was chosen.

2.3 Data Pipeline

The functional purpose of the data pipeline was to move data between different components of the project. A dataflow diagram that was created at the beginning of the project is shown in Figure 1.

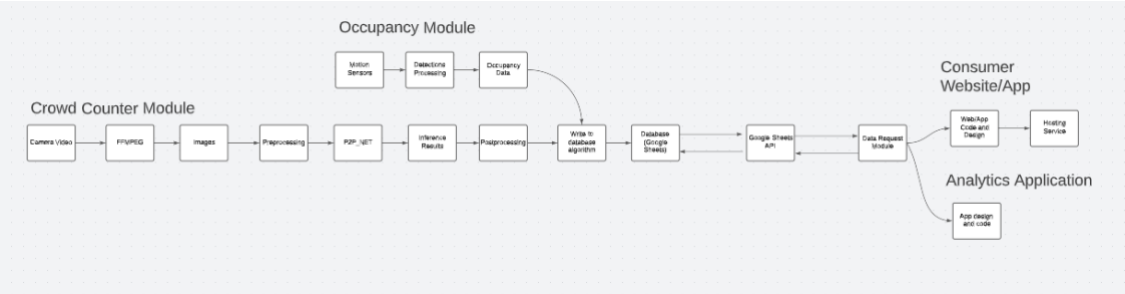

Figure 1: Dataflow diagram of the entire system. Includes the sensor, machine learning and app modules.

The design of the data pipeline depended mostly on the hardware used for the camera and the software and algorithm used for machine learning. The cameras purchased were manufactured by Reolink and thus the image transfer software needed to compatible with the Reolink software. Initially ffmpeg was the tool chosen for image processing and transfer. However, ffmpeg had compatibility issues with Reolink hardware so PyAutoGUI was chosen instead. It would take screenshots of the live feed and those images could then be fed to machine learning algorithm,

To send the data to the machine learning algorithm, the Torch library was used. The reason for this design choice, is that the machine learning algorithm uses the Torch library. The images had to be converted into Torch tensors to be compatible with the machine learning algorithm [1]. The images also had to be preprocessed and shaped to the correct size for a machine learning inference to be performed. The Torch library was tool employed for the preprocessing stage.

To transmit the resultant data to the app, it was decided that using a cloud-based storage medium would be best. The team opted to use the Google Sheets API as it stored data into a spreadsheet that was easy to read from and write to. In addition, the Google Sheets API has a freemium plan that allows for three hundred read and write requests a minute which was more than was required [2]. It allows for two megabytes of transmitted data per request, which would allow for all the necessary data to be transmitted [2]. The Google Sheets API also has JavaScript and Python APIs which made it compatible with both the image processing and application components of the project.

2.4 App Design

For our final project, we decided to use JavaScript and React Native to develop a mobile app. These technologies were chosen because they are well-suited for building cross-platform mobile applications and allow us to leverage existing knowledge of JavaScript as web developers. Additionally, React Native provides a robust framework for implementing user interfaces and handling application state, which was important for our project as it involves displaying and manipulating data in real-time.

In terms of design, we focused on creating an intuitive and user-friendly interface that would be easy for users to navigate and interact with. We used a clean and minimalistic design aesthetic, with a focus on clearly organizing and presenting information to the user. This was achieved using clear

typography, bold colors, and carefully considered layout and spacing. We also included a variety of interactive and auto updating elements to allow users to interact regularly with the app.

2.5 Data Collection and Analysis

The success of a computer vision algorithm can largely depend on the quality, diversity, and size of the dataset that it is trained on. The initial dataset used was the Shanghai Tech dataset containing 1198 annotated crowd images. Each image was stored as a 5-dimensional array, with the first four dimensions containing R,G,B, and A values and the fifth element containing a "points" array to store coordinate data of the location of each person in the. Figure 2(a) below shows an example raw image from the shanghai tech dataset, while Figure 2(b) shows the same image with the points array superimposed on the image.

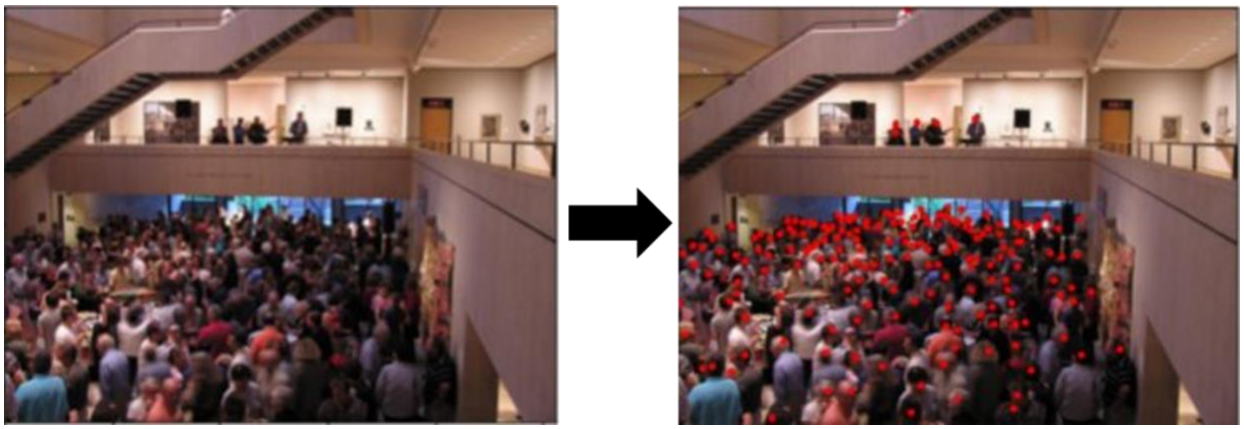

Figure 2: (a) Original image from Shanghai Tech Dataset (left), (b) beside same image with annotations overlayed (right).

One of the limitations in training the QueueHop model on the shanghai tech dataset alone was that most images consist of densely populated scenes, usually containing 100+ individuals. For the use case at Clark Hall, we typically observed far fewer people at any given time, up to a maximum of 30 individuals in the frame. As such, it was imperative to collect image data of students in line at Clark Hall to train the QueueHop model. Over the course of 3 weeks, 3000 raw images were collected ranging from images with 0 people to images of a full line containing 32 individuals. The data set was manually filtered and cleaned to remove duplicate images and a final dataset of 1498 images were collected. To generate a points array for each of these images, VGG image annotator, an open-source online software, was used and each image was manually annotated to match the data structure of the shanghai tech dataset. Figure 3(a) below shows an example of the raw image and Figure 3(b) shows the annotated image from VGG image annotator.

Figure 3: (a) Original image of line at Clark Hall pub (left) compared to the same image (b) with annotations generated manually using VGG Image Annotator Tool.

In addition to image data, capacity and processing time data was also collected. Capacity data was tracked by manually monitoring the inflow and outflow of each individual into the pub. This was used to evaluate the functionality of the time-of-flight sensors. Processing time was defined as the amount of time it took for an individual reaching the front of the line and being processed by the student constable. This data was used in the final design.

3.0 Safety, Environmental Concerns, Ethical Concerns, Equity

3.1 Safety

Secure camera set up was required to ensure the camera would not fall and injure team members or patrons at Clark Hall Pub. When setting up the camera, 4 team members were present to help select an ideal location and setup method, as well as to watch for any unsafe placement. Heights were avoided whenever possible. The camera was attached to a tripod using the manufacturer provided strap and mount and reinforced with duct tape. The tripod was placed on a large, flat surface out of the way of patrons and reinforced with duct tape.

Electrical safety was also considered in the design. The sensor device was placed indoors to avoid electrical hazard. A wireless camera was selected to eliminate risk of tripping over long wires. This also mitigated the risk posed by having wires exposed to the elements if the camera was placed outside.

3.2 Environmental Concerns

Training a machine learning model requires significant compute power, and the ones used in this project took several hours to train on Google Collaboratory GPU. Complexity should be reduced wherever possible to minimize the power requirements to train and run the model.

Waste produced by the project also poses environmental concern. It was therefore a priority to minimize the purchase of new items whenever possible. A tripod already available in Stirling Hall was used to mount the camera rather than purchasing a new one. However, one of the cameras purchased was not used for the MVP. It is important to reuse and properly recycle the materials at the end of the project.

3.3 Privacy Concerns

Privacy is a major concern in this project as it involves recording video footage in public places. Canada's Video recording laws state that it is permitted to record video in any public place where there is no expectation of privacy [3]. Queen's Campus is subject to additional legislation: the Freedom of Information and Protection of Privacy Act [4]. Under this act, it is required to provide notice of security surveillance that will come to the attention of individuals describing what personal information is being collected with a telephone number and/or email contact. To satisfy privacy requirements, the team posted the sign shown in Figure 4 below to show that there is a video recording in progress. Audio was not recorded, and footage was deleted at the end of the project.

Figure 4: Notice of video recording with a team member's phone number as contact information. This was posted outside of Clark Hall Pub during data collection.

4.0 Methodology

The following contains details on the methods used to implement each project component.

4.1 Machine Learning Architecture

Two machine learning techniques were implemented, and their performance was compared to determine the best architecture for the purpose of accurately counting line length at Clark Hall Pub. The Shanghai Tech dataset was used to train each of the networks. Recall from Section 2.4 that each element of the "points" matrix represents one person, meaning that the total number of points corresponds to the number of people in the crowd and represents the total crowd count for a given image.

Mean absolute error (MAE) is the metric used to quantify the success of each model. The formula for MAE is given in Equation 3 below.

$$MAE = \frac{|predicted\ \#\ people - true\ \#\ people|}{true\ \#\ people} * 100\% \quad (3)$$

4.1.1 ECAN Model and CSR-Net: Density-Based Method

The density-based crowd counting method uses the ground-truth annotations to produce a density map for each image, using a k-nearest gaussian density function [5]. Mathematically, the density map is produced using Equation 1 below.

$$F(x) = \sum_{i=1}^{N} \delta(x - x_i) \times G_{\sigma_i}(x), with\ \sigma_i = \beta \bar{d}_i \quad (1)$$

Where $\bar{d}_i$ represents the average distance of the "k nearest neighbors" and G_{σ_i} is a gaussian kernel with standard deviation σ_i, and x is the position of the pixel in the image containing an individual's head.

A convolutional neural network is then trained to reproduce these density maps without the ground-truth, allowing for crowd count estimation of new images. The numeric crowd count is obtained by summing over the values of the density map. The networks are trained with a loss function defined in Equation 2 below.

$$L(\theta) = \frac{1}{2B} \sum_{i=1}^{B} \left| \left| D_i^{gt} - D_i^{est} \right| \right|^2 \quad (2)$$

Where D_i^{gt} denotes the ground truth density map, D_i^{est} denotes the density map estimated by the neural network, and B is the batch size which refers to the number of images under consideration at a time.

The ECAN and CSR-Net density-based architectures were successfully implemented and trained on the Shanghai Tech Dataset [5] [6]. Initial results showed MAE scores greater than 30% after 10 epochs, with qualitatively weak performance on images with sparse crowds. It was therefore decided to pause work on this method and test a density and regression-based architecture. Approximately 4 weeks were spent implementing the ECAN and CSR-Net models, and it would have been more efficient to fully understand the strengths and weaknesses of density-based architecture prior to this.

4.2.2 P2P-Net: Density and Regression-Based Method

P2P-NET [7] is a neural network that utilizes both regression and detection methods to count the number of people in the image. The neural network is supplied a source image. The regression module estimates the number of people in each segment of the image. The detection module estimates where in each segment of the image person is located. For instance, if the top quadrant of the image had an estimated 10 people in it, the detection module would estimate where in the image each person was. P2P-NET is built upon a large convolutional neural network architecture. P2P-NET specifically uses the convolutional neural network VGG -16 as a backbone. The architecture of the neural network is shown in Figure 5.

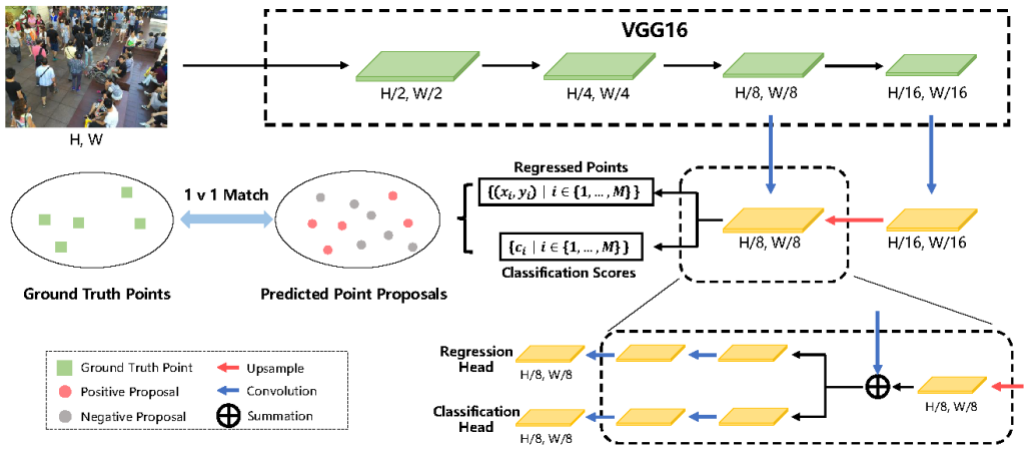

Figure 5: Architecture diagram of P2P-NET. There is a VGG-16 backbone and both a regression and classification module.

P2P-NET [7] was chosen as the algorithm for counting the number of people in a crowd. This algorithm was chosen as it was extremely effective. It had an error of 7.2 percent in terms of mean absolute error.

This error was calculated using data recorded at Clark Hall. P2P-NET performed well in situations with dense and sparse crowds because it uses both regression and density calculation modules. Sample results of the algorithm are shown in Figure 6.

Figure 6: Sample annotations of an image labelled by P2P-NET.

To implement the algorithm, the GitHub repository [8] was cloned. Then all the necessary dependencies were cloned, and the baseline model was downloaded. Then some of the pathing in the repository was changed to suit the new environment. Once this was complete, inferences could be run. Afterwards the model and some of its parameters were tuned based off data collected at Clark Hall.

4.2 Capacity Sensor

The capacity sensor was completed in three main phases. The first phase was the circuit construction and sensor validation. Before implementing the full tripwire program, the team wanted to ensure the sensors would work as desired. This led to the decision to not use the WIFI module the team purchased because of its difficult integration. Each ToF sensor was tested individually to ensure proper measurement readings, followed by testing simultaneous readings when both were connected. Once the electronic components were confirmed, the casing was developed to fit the exact measurements of these components. The SolidWorks model was then 3D printed and pieces were confirmed to fit inside properly and secured using screws. Once pieces were confirmed to fit and the physical design was complete, the tripwire program was developed and tested through many iterations. Once completed to

a point of validation of the program, the device was taken to have further optimization through parameter adjustments within the code.

4.3 Data Pipeline

The data pipeline made use of a variety of tools to move data throughout different components of the project. To move data between the camera and computer processing unit, the Python library PyAutoGUI was used [9] [9]. This library would take frames of video recordings by taking screenshots of the camera feed. These screenshots would then be sent to be processed by a specialized python script where the image would be preprocessed for machine learning.

The preprocessing script would reshape the image into the correct shape for the machine learning model with the Torch library. The resultant data needed to be a Torch Tensor and have the dimensionality 3x512x512. Once the image was preprocessed, it was run through the machine learning model. The machine learning model uses a postprocessing unit to turn the results of the machine learning model into a count of the number of people in the image.

The results of the machine learning algorithm were then uploaded to the cloud using the Google Sheets API to automate the process. The results [9]of the machine learning algorithm were then uploaded to the cloud using the Google Sheets API to automate the process [2]. This API would store all the data in a spreadsheet in a normalized format. This spreadsheet is accessible from anywhere with the correct credentials. This allows for the app subsystem to access and use this data to display how long users will be waiting in line.

4.4 App Design

The app design aspect of this project was completed in several steps. First, a YouTube tutorial was followed to build the backbone of the app using React Native and JavaScript. This provided a basic understanding of the framework, its similarities and differences between other component creation frameworks and allowed development to get started on the project.

The next step comes through creating a pipeline for the app to access data from the google sheets API. Connecting the API to the app involved setting up the necessary credentials and integrating the API client library. This allows the developer to make requests to the API and retrieve data to be displayed in the app. These requests are called using fetch request. The fetch request is part of the Fetch API, which is a modern way of making network requests in web applications. A fetch request is typically made using the '**fetch()'** function, which takes a URL as an argument and returns a promise that resolves to a response object. This was chosen as fetch requests are a powerful and flexible way to request data from a server in a web application.

After the API was connected, the focus shifted to the user interface (UI) of the app. This involved designing the layout and structure of the app, as well as creating the visual elements that would be used to display the metrics from the API. Once the basic UI was in place, the final development became customizing the app to match the look and feel of Queue Hop. This included using the same color scheme and font, as well as incorporating similar design elements.

4.5 Data Collection and Analysis

Processing time data was collected for 221 people over the span of 3 hours and estimated wait time was calculated using Equation 4 below by multiplying the output of the P2PNet model (estimated

number of people) by the average processing time. The key limitation of this model is that it is unable to predict the wait time when the pub reaches capacity. To extend this to include wait time estimation at capacity, a statistical approach could be taken to determine the expected duration of a patron's stay; however, for a reliable model to be built, more data would need to be collected.

$$Wait\ Time = number\ of\ people\ in\ line\ \times \frac{1}{N}\sum(processing\ time) \quad (4)$$

5.0 Final Design

The final design for each element of the project is discussed below.

5.1 Machine Learning Architecture

The P2P-Net was chosen as the final computer vision model due to its superior performance. The camera was set up on a tripod in Clark Hall with an overhead view of the line through a window, shown in Figure 7 below for collection of images which were run through the P2P-Net on a team member's laptop.

Figure 7: Camera set up on tripod in Clark Hall Pub for data collection. The video stream is viewed live on a laptop, where a snapshot was saved every 10 seconds.

Images collected exclusively during the daytime were used due to significant glare at night. The final design achieved a MAE score of 7.2%.

5.2 Capacity Sensor

The final design of the capacity sensor consists of two VL53L0X ToF sensors, which are designed to measure absolute distances of up to 2 meters away from the sensor. The programmable I2C address of the sensor makes it an easy integration to get multiple sensor readings in parallel without computational cost. The sensors are connected to an Arduino Uno, which proved to have enough dynamic memory to efficiently run the tripwire program, but any additional sensors or actuators would most likely be too much for the controller to handle. A casing for both the sensors and the Arduino was modelled using SolidWorks CAD, as seen in Figure 8,with both sensors being approximately 25 centimetres apart from one another with the Arduino in between. The casing was printed out and the components were installed using small 3/8-inch screws to fit with the holes built into the CAD design.

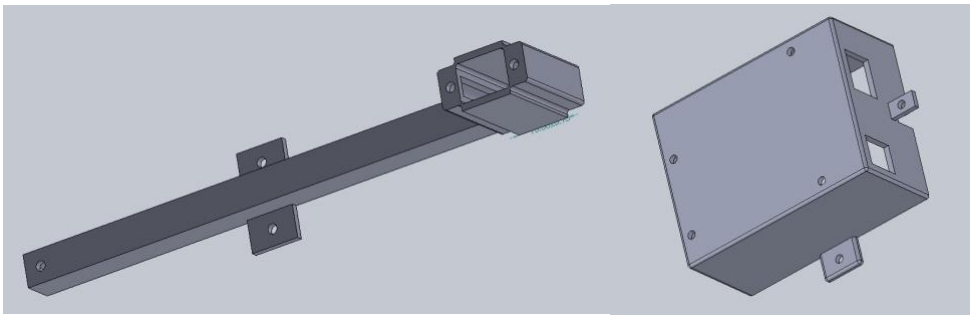

Figure 8: Shows CAD design for ToF sensor casing (Left) and Arduino casing (Right). Designed in SolidWorks, the pieces are designed to bind together using screws

The tripwire program involves setting a base distance for the ToF sensors, which is used to detect a person if a new measurement is closer than the base distance. Once this trip has been detected, a while loop is called to wait for the second sensor to be tripped, or for nothing to happen and to return to the original measurement loop. The main adjustable parameters of the program are the delay time between sensor measurements, the wait time between detecting a second trip, and the base distance of the trip function. The program updates the serial monitor only when the capacity has changed, displaying the total capacity count and the time stamp. A python script was written to also take in the serial monitor information so that this data can be logged using the Google Sheets API.

5.3 Data Pipeline

The final design of the data pipeline used the Python Library, PyAutoGUI, to transfer images between the camera and the computer processing unit. Before being sent to the machine learning algorithm, images would be preprocessed using the Python library Torch. After the results of the machine learning algorithm were processed, they would be sent into a cloud database using the Google Sheets API. The pipeline met the functional requirements laid out as it would take 10.2 seconds to go through the entire pipeline. The final data pipeline was both robust and quick. It met all the necessary functional requirements.

5.4 App Design

The final design of the app has 3 pages. A landing/home page, a description page, and a profile page. Using few pages for the app makes it easier for users to quickly move find what they are looking for and allowed for the full app to be designed and built by one person over the course of a semester. This app was made primarily to display data obtained from the other aspects of the project, yet there are also many features that are part of what would be the Beta launch. These features, while having no direct functionality or relation to the rest of the project, were added to make the app look more professional.

The profile page is a trivial page that shows displays your name and profile picture. Clicking on the profile picture will allow the user to change that picture for display. This page is displayed in Figure 9.

The landing page is much more involved as it is the first page that a user will see when they open the app. The primary focus of the page is the list of cards that contain different bars in the Queen's area. The first in the list and only one that has true data attached to it is Clark Hall Pub. Each card contains information about the bar including wait time, and number of people in line, and the address.

The pictures of teammates are also added in as a perspective social aspect of the app. Seven other Kingston bars are added to the app with their own filler metrics and information. There is a search bar at the top of the page that allows the user to search for a specific bar by name, address, and description. This page is shown in the center of Figure 9.

The details page can be accessed by clicking the Get Info button. This page is used to display more information about the bar, including a grid of boxes that display four calculated metrics from the data: the current queue count, the estimated wait time, the average wait time for today and the number of people inside. It also includes a description of the bar with an expandable read more button. At the bottom of the page, there is a button reading 'I'm in line'. Once this button is pushed, a timer will be activated to display the amount of time that the user has waited in line. Two buttons will be displayed reading 'I got in' and 'I left the line' which will end the timer and return the display to the original single button. This screen is shown on the right side of Figure 9.

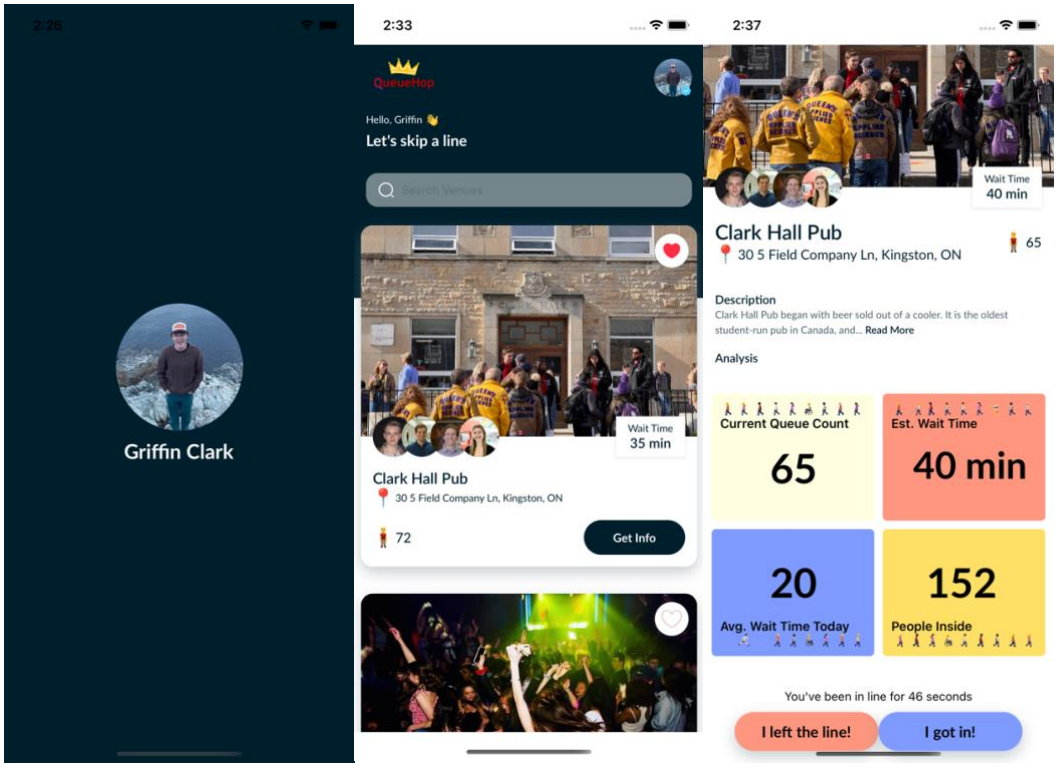

Figure 9: The three main screens of the user interface. From left to right we have the profile, home, and details pages.

5.5 Data Collection and Analysis

The final image dataset consisted of 1498 annotated images from Clark Hall were reserved to test the accuracy of the model. In addition to the Shanghai Tech dataset, 998 of the annotated Clark images were used to train the model, while the remaining 500 images were reserved to test the error of the model. Average processing time was calculated as the average of 221 datapoints, and was determined using Equation 4 to be 72 seconds:

$$Wait\ Time = \#\ of\ people\ in\ line\ \times 72s$$

Capacity data was collected by tracking the inflow and outflow of patrons by manually tracking the number of people entering the pub and saving this. Figure 10 shows the final capacity data that was recorded over the span of 3 hours.

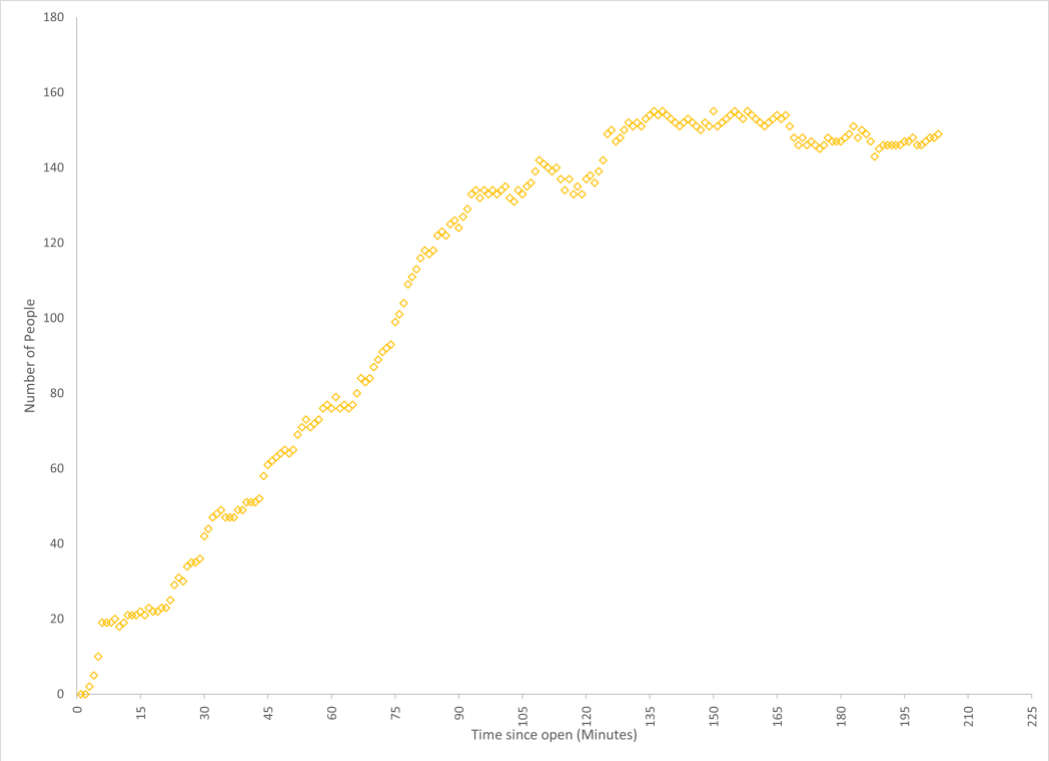

Figure 10: Plot of the capacity of Clark Hall Pub over the span of 3 hours.

6.0 Testing, Validation, and Iteration

Overall, six out of seven quantifiable goals spread across the different project elements were achieved. A summary of project goals against results is displayed in Table 2 below.

Table 2: Comparison of target and measured values for 7 quantifiable objectives.

Objective	Target	Measured	Pass or Fail
Line length estimate accuracy	Crowd count accurate within 1 person or 10%, whichever is larger	7.2 % error	Pass
Time to process image	1 minute per image	7.75 seconds	Pass
Overall Sensor Accuracy	Detect 80% of people	30-minute trial – 78% accuracy	Pass
Speed	Measure individuals walking as fast as 4 km/h	Measure individuals at a running pace exceeding 12 km/h	Pass
Sensor Range	Detect people up to 2 m away	Detect people 1.4 m away	Pass
Wait Time Accuracy	20% error when capacity not reached	Model built but not tested	Fail
App Frame Rate	30 fps	60 fps	Pass

The machine learning model achieved a 7.2% MAE score and can therefore predict line length within 7.2% error, which satisfies the 10% or one person goal. It occupies 527 MB of storage and has a runtime of 7.75 seconds. However, the wait time accuracy goal was not achieved; a predictive model was built, but not sufficiently tested with real data.

The sensor device achieved 78% accuracy for the total capacity count in a 30-minute trial that is taken to be representative of overall performance. Accuracy was calculated by dividing the number of individuals detected passing by in the correct direction by the number obtained by counting manually. Individuals can be detected up to 1.4 meters away, which was determined by walking back and forth in front of the sensor at 10 cm increments, where the distance was measured using a ruler. The sensor can detect individuals running at speeds greater than 12 km per hour, which was determined by measuring a runner's speed 1m away from the sensor, using a stopwatch on a fixed 20m long path. The speed was gradually increased from 5 km per hour to 12 km per hour, and it was assumed that no individual would pass the sensor at a pace greater than 12 km per hour.

The app has a frame rate of 60 frames per second which provides a user-friendly experience. It can also retrieve and display data from the Google Sheets API.

7.0 Discussion

Overall, the project was a success. Each of the project's subsystems met their functional requirements and performed as designed. Nearly every quantitative and qualitative objective for the project was met during the semester. Any failure to meet a predetermined objective was the result of lack of time to work on that objective, not faulty design. With more time to integrate and implement all the project components, the system would work effectively to provide information on the capacity count of Clark Hall to both workforce and patrons.

While there are some limitations of the system, these limitations are addressable with more iteration on the project. Addressing these limitations was considered during the semester but they were considered out of the scope of this project.

The main limitation for the person counting sensor, was that it could not detect two people walking past it at the same time. To address this, the sensor needs to be put in a doorway where only one person can walk through at a time.

The key limitations of the crowd counting portion of the project was that it could not handle image data from the nighttime and that it was not built to support multiple cameras running at once. The lack of ability to handle nighttime data mainly stemmed from the placement of the camera. It was placed in front of a window which produced glare. If the camera was moved outdoors, it would be able to get much clearer images at night and get a better view of the crowd. To improve the app, future development goals would involve adding in more visual components to improve the user experience.

The next steps for this project would be to fully integrate and implement all components of the project. If all the components were combined, the result would be a hardware and software system that would generate information about the capacity and line count for Clark Hall. The distribution of this information would have benefits for both Clark Hall patrons and management. For patrons, it would allow them to make decisions about whether to go to Clark Hall. The line count and capacity information would allow management to determine how to deploy scaling information.

8.0 Economic Analysis

Through many iterations of the project design, there were various items and materials that were purchased, though some were not implemented in the final product. The budget break down can be seen in **Error! Reference source not found.**with an additional category describing whether the piece purchased was taken from lab inventory or ordered in. Both the Arduino WIFI module and the second Reolink camera were not used in the final design of the project, meaning to recreate the project to its current state, it would cost approximately 310 dollars. However, in future plans for the design, the second camera would be implemented to expand the view of the line up around the corner of the pub.

The economic potential of wait time and line length analytics can be estimated by examining the implications of such statistics on a restaurants profit. If one considers how consumers having accurate expectations for wait times can increase revenue, there exists some quantitative research in the area. These expectations being met or even exceeded will cause the users to have a more favourable and satisfactory experience inside [10]. The TempKin Group estimates that a small increase in customer

satisfaction can 'generate an average revenue increase of $823 million over three years for a company with $1 billion dollars in annual revenues [11].

If the analytics were to be sold as a business only interface, there would still exist many opportunities for increasing revenue. If common restaurant rushes and lulls can be determined to a higher degree of accuracy, it will aid restaurant management in properly allocating resources, most notably the wait and kitchen staff scheduled. The average restaurant uses approximately 30% of its revenue on staffing [12]. An opportunity to reduce this by cutting workers when business is predicted to be low, would be a significant advantage to any restaurant. Further action with these metrics can be taken by introducing promotions to counteract consistently slow business times, and therefore adding more business and revenue. Other industries would also benefit from such metrics, but the scope of QueueHop has been limited to the restaurant and bar industry.

9.0 Conclusion

Overall, this project was a success in almost all metrics measured. The team leveraged P2P-Net, an advanced machine learning model to count the number of people in the Clark Hall Pub line with an error of 7.2%. Data was collected by the team and manually labeled to train and test this model. Images were able to be processed ten times faster than the goal set for the project which allows for real time data aggregation and analyzation. Two time of flight sensors were combined with an Arduino and encased in a CAD model to measure the flow rate of people entering and exiting the pub. The sensor was able to measure people from a distance of 1.4 meters at speeds topping 12 km/h all while maintaining a 78% accuracy. These metrics are then visualized on a mobile application built with JavaScript and React Native. This multiplatform app was designed with usability as a priority and surpasses a frame rate of 60 fps matching the standards of many of the App Store's top apps. The app was able to display live metrics as a pipeline between the machine learning model was made through a google sheets API.

While this project achieved many successes, there are some goals that were not attained. A goal of accurately estimating the time a user will wait in line was not realized as time became a constraint on testing the accuracy of this metric. Furthermore, the full integration of the data analyzed by the model and the data obtained by the sensors being transmitted through the API to be displayed in the app was not fully implemented. Achieving these goals, along with increased user forward elements and better data visualizations in the mobile application rank the highest in priority if this project is to continue.

At the moment, this project would not be able to be used for students in the Queen's community. Through the implementation of the goals in the previous paragraph, as well as the induction of the app to the App Store and the Google Play Store would allow for a beta version of the project to be released. In the long-term, the goal becomes collaboration with other bars in the Kingston area and monetization of our services into a software as a service (SaaS) business model.

References

[1] Pytorch, "PyTorch Vision Transforms," Torch, [Online]. Available: https://pytorch.org/vision/stable/transforms.html. [Accessed 11 12 2022].

[2] Google, "Google Sheets for Developers," Google, 2022. [Online]. Available: https://developers.google.com/sheets/api/limits. [Accessed 11 12 2022].

[3] "Canada Audio and Video Recording Laws," [Online]. Available: https://recordinglaw.com/canada-recording-laws/. [Accessed December 2022].

[4] "Freedom of Information and Protection of Privacy Act, R.S.O. 1990, c. F.31," 2022.

[5] L. e. Al, "CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly," 2018.

[6] L. e. Al, "Context-Aware Crowd Counting," 2019.

[7] C. W. Z. J. Y. W. Y. T. C. W. J. L. F. H. Y. W. Qingyu Song, "Rethinking Counting and Localization in Crowds:A Purely Point-Based Framework," *Computer Vision and Pattern Recognition,* vol. 2022, 2022.

[8] Q. a. W. C. a. J. Z. a. W. Y. a. T. Y. a. W. C. a. L. J. a. H. F. a. W. Y. Song, "Crowd Counting - P2P-NET," GitHub and Tencent, 31 10 2021. [Online]. Available: https://github.com/TencentYoutuResearch/CrowdCounting-P2PNet. [Accessed 11 12 2022].

[9] PyAutoGUI, "PyAutoGUI - Screenshot Function," PyPi, 2022. [Online]. Available: https://pyautogui.readthedocs.io/en/latest/screenshot.html. [Accessed 11 12 2022].

[10] R. W. S. N.-M. T. G. B. D. Holly Chu, "The psychology of the wait time experience – what clinics can do to manage the waiting experience for patients: a longitudinal, qualitative study," *BMC Health Services Research,* vol. 19, no. 459, 2019.

[11] A. Graves, "Why Restaurant Customer Satisfaction is So Important," 2022.

[12] A. Chegini, "What Percentage Should Labor Cost be in a Restaurant," 7 April 2021. [Online]. Available: https://www.eposnow.com/ca/resources/what-percentage-should-labor-cost-be-in-a-restaurant/. [Accessed 11 December 2022].

[13] Reolink, "Reolink Argus 3," Reolink, [Online]. Available: https://reolink.com/product/argus-3/#specifications. [Accessed 11 12 2022].

Appendix for:

QueueHop: Computer Vision Wait Time Estimation for Clark Hall Pub

Alex Chase, Bryn Macduffee, Jonah Garmaise, Julia Everitt, Griffin Clark

Department of Physics, Engineering Physics & Astronomy, Queen's University, Kingston, ON, Canada

Submitted December 12, 2022

Appendix A – Bill of Materials

Table 3: Shows full list of parts used in project, along with prices and column showing whether pieces were ordered in or taken from the physics inventory

ITEM	PURCHASED / FROM INVENTORY	PRICE (CAD)
3D PRINT FOR CAPACITY COUNTER	Purchased	$31.50
VL53L0X TIME OF FLIGHT SENSORS (X2)	Purchased	$11.99
ARDUINO UNO	From Inventory	$35.99
ESP8266 WIFI MODULE - **NOT USED**	Purchased	$10.99
REOLINK CAMERA WITH MAGNETIC MOUNT (X2) - **ONE NOT USED**	Purchased	$368.33
TRIPOD	From Inventory	$50.00
TOTAL		$508.80
TOTAL PURCHASED		$422.81