Customer Protection Utilities

GitHub Repository: https://github.com/SBriguglio/OccupancyTracker

Group One:
Spencer Briguglio
Mohamad Elchami
Yashvi Shah
Jashanpreet Singh

Table of Contents

Introduction

During the past year and some, the occurrence of COVID-19 has reshaped the way society menovers and interacts. With this shift, society as a whole has become much more conscious of their health and hygiene. As a subsequent result and common nature of technology, new solutions have been developed to maintain and tailor towards these new societal norms.

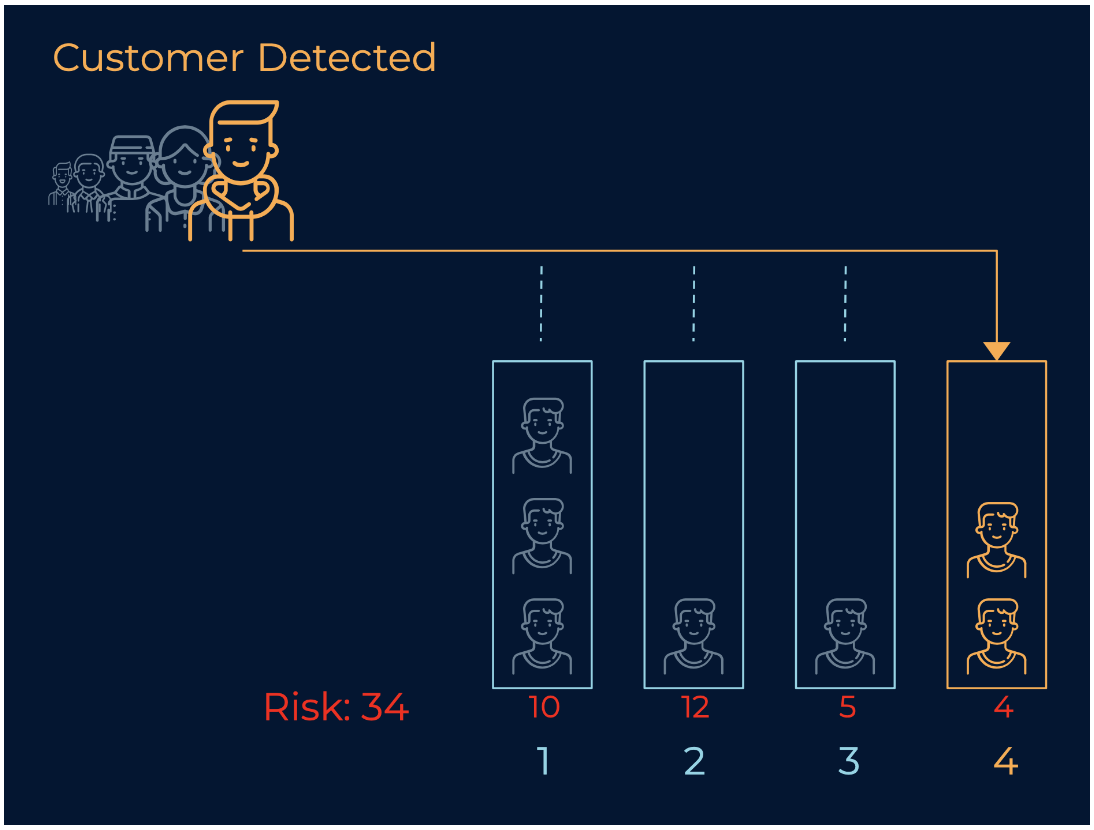
In this project, we aim to examine these situations and develop tools to help maintain and abide by these newly developed societal procedures. The overall target of the project is to reduce the spread of COVID-19 in commercial environments. We seek to utilize visual image processing and artificial intelligence processing to develop a plethora of tools which can be useful in developing metrics and facilitate optimal operating conditions for all parties involved.

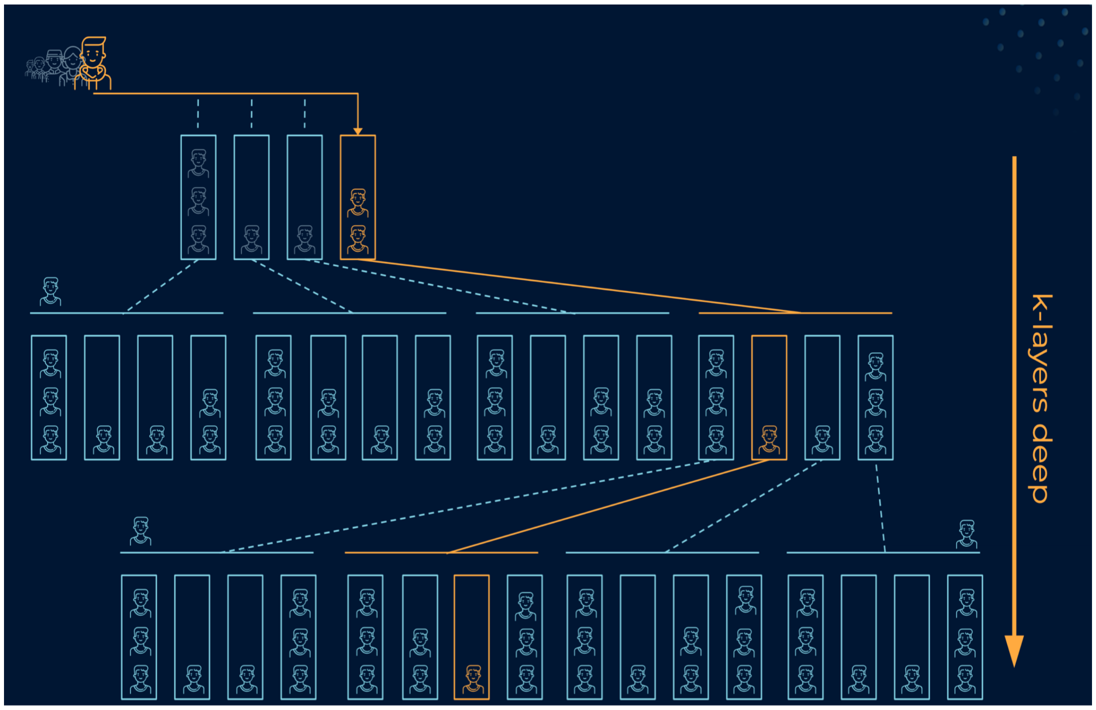
Background & Related Works

Generally, human-tracking is accomplished by, first, recognizing individuals that are visible in each frame and, thereafter, following them across different casings using multiple frames utilizing either extracted features (like size, shading histograms, edge direction, and so on) (Owens et al., 2001)[3], or movement correspondence utilizing a Kalman filter, a particle filter, or different techniques (Veenman et al., 2001)[4].

Other research in counting untagged individuals includes the utilization of various sorts of sensors and procedures. People count systems using multi-sensing applications (K. Hashimoto et al., 1997)[5] utilizes a variety of PIR sensors in masterminded in a line to recognize the quantity of individuals through a stairway. In A real-time people counter (Conrad & Johnsonbaugh, 2004)[6], the middle scanlines of a picture outline are utilized for a similar effect. These methodologies are utilized to identify individuals that go through a bound territory, and don't adjust well to open spaces. Also, since every individual is just identified at doors and ways out, any mistake at location time will propagate indefinitely.

Figures





Simulation Details

The project was completed in incremental steps, beginning with the basic core elements and building to the desired applications. Breaking down the projects we can see a few main aspects that stand out, one being object detection, another being implementation of object detention and data extraction, and the last being the utilization of gathered data in artificial intelligence techniques.

We will begin by examining the object detection step. As mentioned, object detection is a major component of this project, if not the most important. Given that our main source of monitoring is video feed, object detection was determined to be the optimal method of determining important variables. To explore and build this initial step we examined types of object detection and tools utilized for this. Building from a tutorial we learned to detect objects from a trained set, in this case a face. Further implementing the program we explored the use of centroids in tracking that object between each frame of the video feed using euclidean distance. As a result a face detection application was developed which was included in the final project simply as an additional tool to the overall final program. (Rosebrock, *Simple object tracking with OpenCV* 2020)

Moving to the next step we began to explore the uses of centroids and different training sets. Following a similar tutorial an occupancy counting program was developed. Using the same technique of centroid determination, the program is able to monitor a single point of entrance/exit to determine the current occupancy of a residence by determining crossing direction. This iteration explored a different style of training sets as well as a different object, the human body. This program was also included in the final project for execution as it aid to the idea of the project being a set of utilities. From here we began to explore the idea of the centroids more and further explored how we implement them in additional use cases when pertaining to very specific coordinates. (Rosebrock , *OpenCV people counter* 2020)

The last step was to create a register assignment program. In this step we bring together concepts explored and learned in the previously mentioned procedures. Key components carried into this final program include: unique object detection, centroid determination, centroid association for tracking between frames, and image processing techniques.

After our body detection implementation detects a customer waiting to be assigned a queue, it is intended to detect the number of people queued for each cash register in the checkout. In this respect, there are two types of queues, one queue of customers waiting to be assigned a register at the checkout and *n* register queues where customers are in line at a register ready to pay (*n* is the number of registers in the checkout). The face/object detection implementation passes the number of people in *n* register queues to the AI component. The AI records this initial state as an array with *n* elements to construct a search tree using the A* algorithm and predicts which register assignment would be the safest for the customer by assigning a cost to each state equivalent to the risk associated with said state. Each element, *i,* in the array is the number of people queued at register *i* in the checkout.

Risk is assessed in two steps. First, a risk associated with each register is calculated. This risk is determined using the formula:

$$register\, risk(x) \ = p_x^{p_x - 1}$$

where *p* is the number of people queued for register *x*. Once the risk for each register is determined, the risk of all queues adjacent to the *i*-th queue is assessed using an inverse-squared-distance formula where *n* is the number of registers in the checkout:

$$vicinity\, risk(i) \ = \sum_{x=0,\, x \neq i}^{n} \frac{register\, risk(x)}{|x-i|^2}$$

The total risk for each queue/register is simply the summation of the register risk and the vicinity risk. Finally the total risk of a state is determined by the following formula:

$$state\, risk \ = \sum_{i=0}^{n} vicinity\, risk(i) \ + \ register\, risk(i)$$

Table 1: shows a demonstration of how risk is assessed for a checkout containing n=10 cash registers.

n=10	**Registers**									
i	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
people	3	2	1	1	4	1	3	0	2	1
R. risk	9	2	1	1	64	1	9	0	2	1
V. risk	6.69	13.09	22.00	67.86	5.45	74.13	18.11	18.95	7.61	5.81
Q. risk	15.7	15.1	23.0	68.9	69.4	75.1	27.1	19.0	9.6	6.8
State risk	329.7									

We believe that this model accurately represents the risk inherent in checkout systems based on the value we obtain from our risk calculations. We see that the formula used to determine register risk also incentives short line lengths as risk is increased exponentially with every addition to the queue for a particular register. Actually, the total risk of a state with all 18 individuals in a single line would be at least 20 orders of magnitude larger than the current state risk of the state in table 1. Additionally, the inverse squared method for determining risk associated with adjacent isles seems to ensure that the shortest length lines are not automatically chosen by the algorithm as the best possible choice (something that a human agent might naively assume). In table 1, for this state, we see that the lowest risk register is actually register 9, largely due to its distance from register 4 which has relatively high risk. Interestingly, even though register 4 has the most amount of people within it, register 5 is the most risky because it is between two high risk registers which contribute greatly to its calculated vicinity risk. By calculating state risks like this for each possible state, the AI can accurately model the real world changes in risk as it assigns people to different registers in the checkout.

In addition to risk assessment in the short term, we wanted to predict how risk would evolve in the long run. Low risk decisions made now may lead to high-risk situations in the future. This should be avoided. In order to simulate this, the algorithm is passed a throughput rate which represents the number of registers in the checkout which are expected to process a customer while customers are added. Throughput rate is a number from 0-1 which is determined by empirical observation of the checkout. For now it needs to be set manually, but we would like to make this an automatic function in the future by expanding our object detection capabilities. If the checkout contains 10 registers and is expected to process 1 customer every time a customer is to be assigned a register, then the throughput rate would be 0.1 or 1/10. Using the throughput rate, the algorithm randomly reduces the people in a queue for subsequent states while also adding another person to the *safest* queue in that state to represent the next customer waiting to be assigned a register. It should be noted that when generating these subsequent (leaf) states, each state has the same randomly chosen register queue decremented. This makes it so states in each level in the tree only vary by where a customer is assigned to go. The optimum choice is made by finding a goal state in the tree.

A goal state is selected by the A* algorithm in a slight different manner than what we have done previously in class. Instead of searching only for a specific state, the algorithm also stops when the *k*-th level of the search tree is filled with states and the cheapest state is chosen. Because cost is used to model risk, the cheapest state is the same as the state with the least risk. Once this state (or a goal state of an empty checkout) is found, the exact traversal path to that state is mapped back to the start state. The leaf that directly leads to the goal state is the one that is chosen for the customer to be allocated to. The algorithm stops at the *k*-th level (*k* is decided by the user) because it may take infinitely many steps to reach an empty state as customers are added to the queues as each leaf is created.

Tools Appendix

We will utilize a wide variety of tools and libraries to build this project. All tools and libraries are open source accessible.

- **Python** (3.9.4 or similar): Python was utilized for its large quantity of data, image processing, and artificial intelligence libraries available.
    - Additional Information: https://www.python.org
- **Numpy**: A numerical computation module which is additionally used in many other modules and libraries.
    - Additional Information: https://numpy.org
- **OpenCV**: OpenCV is built off of numpy. It is a large and powerful library for image processing.
    - Additional Information: https://docs.python.org/3/library/argparse.html
    - For Python: https://pypi.org/project/opencv-python/
- **collections**: Collections is a python library consisting of data structures.
    - Additional Information: https://docs.python.org/3/library/collections.html
- **argparse**: A module(library) for parsing command line arguments within a program.
    - Additional Information: https://docs.python.org/3/library/argparse.html
- **imutils**: A convenient module for image processing.
    - Additional Information: https://pypi.org/project/imutils/
- **dlib**: A powerful c++ based library used for a plethora of use cases.
    - Additional Information: http://dlib.net
    - Installation for Python: https://www.pyimagesearch.com/2018/01/22/install-dlib-easy-complete-guide/
- **TKinter**: A standard GUI toolkit for python
    - Additional Information: https://docs.python.org/3/library/tkinter.html

Discussion

Throughout the past year, SARS-CoV-2, the virus that causes COVID-19, has wreaked havoc on the global economy, supply chains, travel and the health of millions of people. As of the time of writing, nearly 135 million people have been diagnosed with COVID-19 globally with nearly 3 million people dead (Johns Hopkins University & Medicine, 2021). Governments around the world started initiating lockdowns and exposure limiting measures in the months following the declaration of COVID-19 as a pandemic by the World Health Organization with much of the burden of adopting these measures being placed on individuals and businesses. Essential services like grocery stores remained open, however, and present a particularly high exposure risk for customers, even at reduced store capacity. In order to limit risk in stores, customers are required to wear masks, social distance and more. Even though customers may be able to limit exposure while shopping, this is much more difficult to do while queuing for the checkout where people often stand for a prolonged amount of time without strictly adhering to social distancing guidelines. Infact, many big box locations have an employee paid to direct customers to the appropriate cash register. This also presents a risk to the employee because they need to risk prolonged/close contact with customers in order to direct them properly. Furthermore, human error may result in excess risk and inefficiencies in processing customers.

In light of these issues, we decided to attempt to create a program which can use face/object detection to direct customers to the best checkout in terms of risk reduction. The program was intended to detect customers waiting to queue at a register while also counting the number of people waiting in a register queue. The program does this by detecting customers needing to be assigned to cash register while also counting the number of customers queued for each register in the checkout. This information is stored as a state which is an array of *n* elements where *n* is the number of cash registers in the checkout. Each element stores the number of people queued at a register. As described in the simulation section, the A* algorithm uses these states to generate new child states and find the lowest risk assignment.

Though this seems at first glance like a feasible way to predict possible states and model the randomness of a checkout system, we think we could improve the model by assessing individual throughput times of each register and cart fullness. By tracking the time a customer is present at the register, we could better predict the next queue to be emptied and then construct a much more accurate tree of states with the A* algorithm. This is one of the first things we wish to address in the future because it would likely have the highest impact on the efficacy of this system.

Other than this, the model does a decent job at creating predictions. What is needed is a better method of passing information from the face/object detection components to the AI. As of right now it is manual, which is not acceptable for wide-spread commercial use of the product, especially when a simple, user-friendly solution is needed. However, as a first iteration, we believe the program has great potential now that the foundation has been built.

References

[1]Rosebrock, A. (2020, April 18). OpenCV people counter. Retrieved April 10, 2021, from https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/

[2]Rosebrock, A. (2020, April 18). Simple object tracking with OpenCV. Retrieved April 10, 2021, from https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

[3]J. Owens, A. Hunter, and E. Fletcher, “A fast model free morphology-based object tracking algorithm,” in Proceedings of the British Machine Vision Conference, 2002, from https://www.researchgate.net/publication/221259805_A_Fast_Model-Free_Morphology-Based_Object_Tracking_Algorithm

[4]C. J. Veenman, M. Reinders, and E. Backer, “,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, March 2001, from https://ieeexplore.ieee.org/document/899946/authors#authors

[5]K. Hashimoto, K. Morinaka, N. Yoshiike, C. Kawaguchi, and S. Matsueda, “People count system using multi-sensing application,” in IEEE International Conference on Solid-State Sensors and Actuators, June 1997, from https://ieeexplore.ieee.org/document/635472

[6]Gary Conrad and Richard Johnsonbaugh, “A real-time people counter,” in SAC ’94: Proceedings of the 1994 ACM symposium on Applied computing, New York, NY, USA, 1994, pp. 20–24, ACM Press, from https://dl.acm.org/doi/10.1145/326619.326649