Grocery Store Checkout System

Team C3: Brian Chhour, Shubhi Jain, Simon Xu

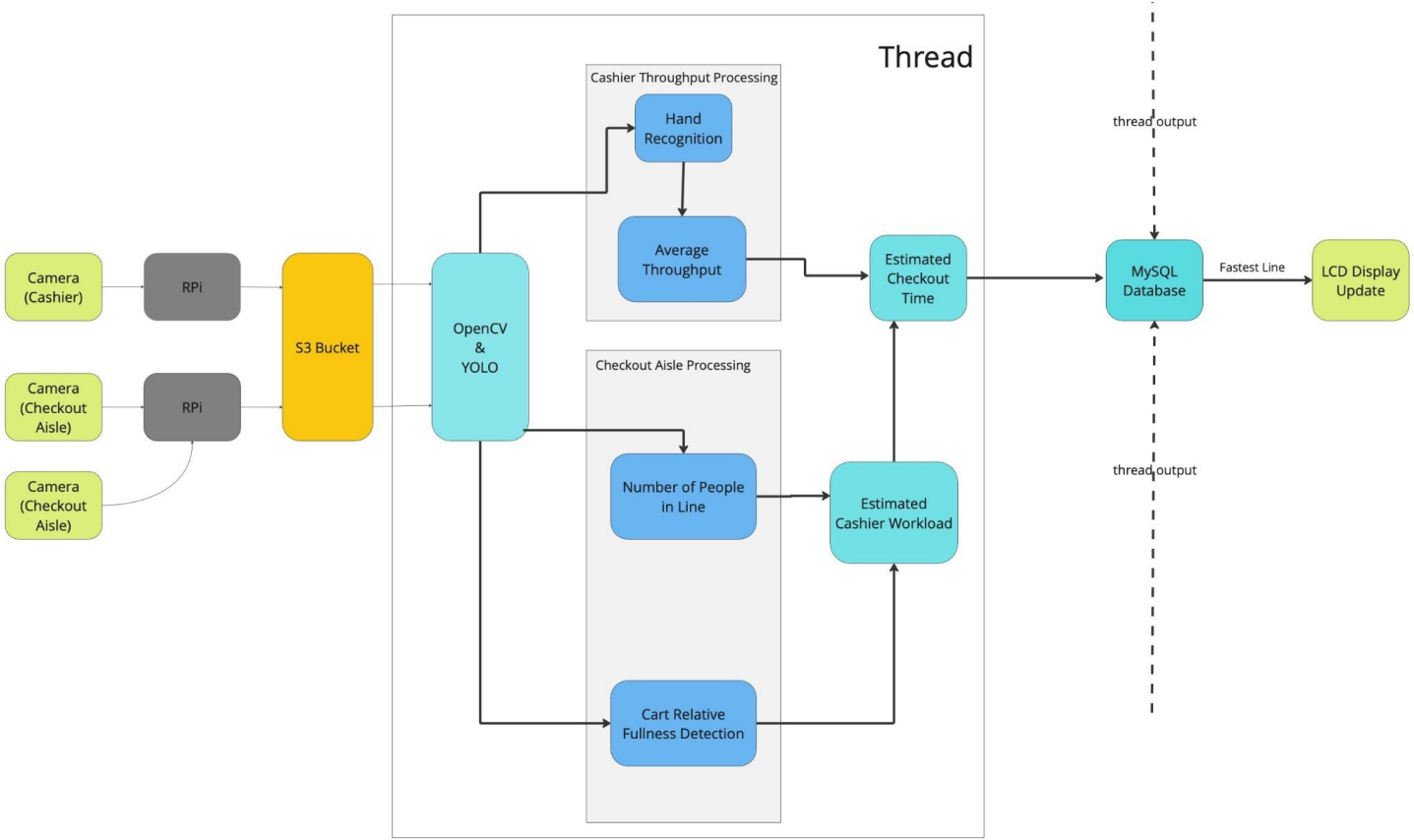
Application / Use Case

- When in a grocery store, we want to check out as soon as possible
  - Just eyeing the number of people seldom works
- We want to build a system that chooses the best checkout line for a given user to go to in order to minimize their checkout time
- Main priority is how accurate the predicted fastest checkout line is

Quantitative Design Requirements

Requirement	Metric
Time between obtaining video data to computing fastest checkout line and displaying it	< 5 seconds
System determining the fastest checkout line correctly	>= 85%
Margin of error on determining relative fullness of someone’s cart	< 20% error
Margin of error on average throughput of cashier	< 20% error
Margin of error on determining number of people in a line	< 10% error

Solution Approach - Algorithmic Layer (TODO)

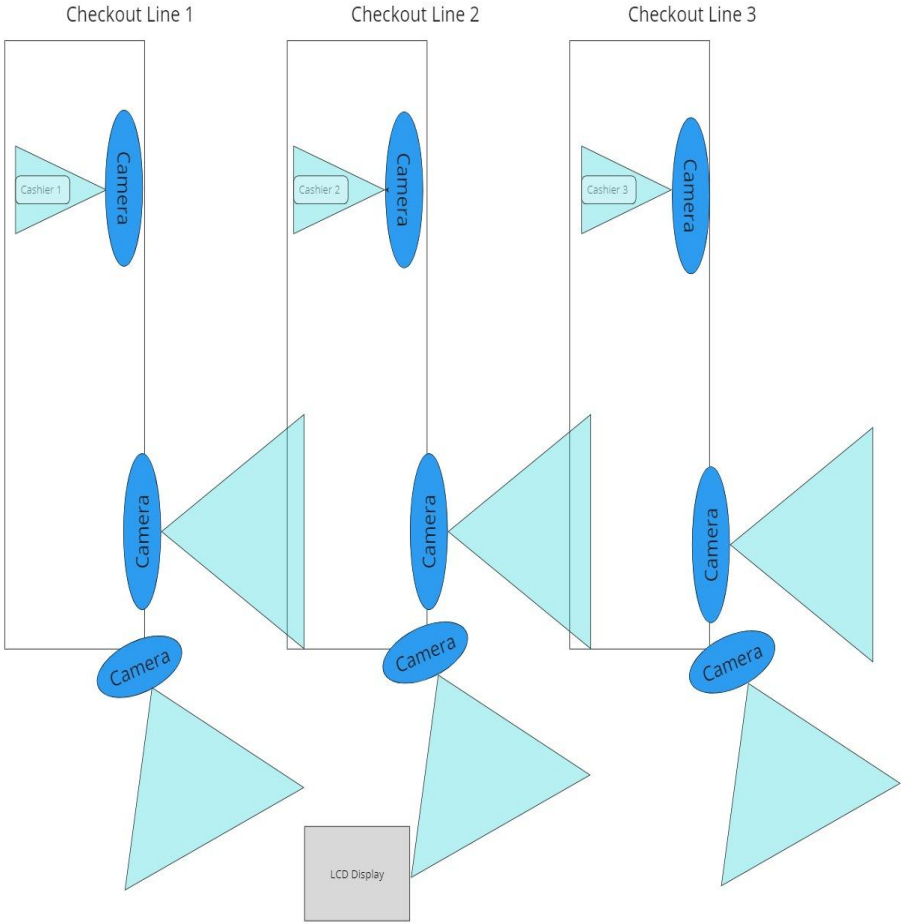


Solution Approach

- After we download 3 separate streams of footage from S3, we have 3 separate models for detection:
    1. Throughput, where we detect the cashiers hands to see how fast items are being scanned
    2. Shopping carts, where we are detecting shopping carts passing into and leaving the line, and computing fullness when we detect a cart
    3. Line detection, where we are trying to find the number of people in the line
- We convert the number of people and the percent fullness into an estimated time based on cashier throughput

Complete Solution

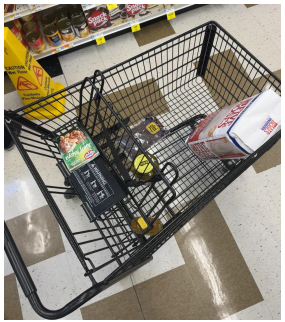
- Have footage taken from RPis running in Salem's while we are demoing in person
- Have two lane setup for demo with some carts + items
- Show how system changes when we join the line with varying amount of items
- Display what our models are predicting



Testing - Fullness Detection

Testing Method:

- Ran module on carts with different fullness
- Expect the difference in output to reflect the difference in actual fullness
- Outcome:
  - 5 different tests with 3-4 carts each
  - 60% of the time carts were within our margin of error



13%

28%

36%

Testing - Cashier Throughput

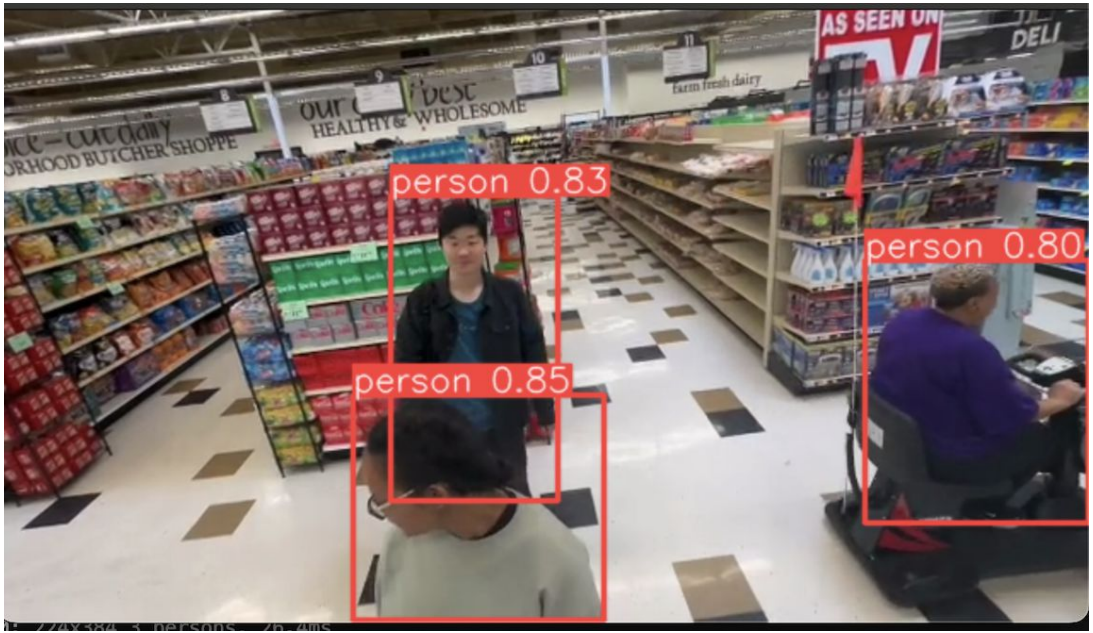
Testing Method:

- How often that the cashiers hand is detected as having put an item on the other side of scanner (which indicates a completed scan)
- 3 different videos around 20-30 seconds each
- Accuracy: 58%

Testing - Line Detection

Testing Method:

- How often the number of people in line is actually the correct number



Testing - Full System Latency

	Worst Case	Average
Video Length	1 second	1 second
Upload	0.9 seconds	0.5 seconds
Download	0.7 seconds	0.6 seconds
System Computation	2 seconds	2 seconds
Total	4.6 seconds	4.1 seconds

Design Tradeoffs

3 Cameras vs. 2 Cameras

- Determine fullness of each cart as well as number of people in line - initially relying on same camera for both

Cashier Throughput Observation Method

- Initially relied on CV to track items being scanned, now tracking hand movement

USB Connected Cameras vs RPis uploading to S3

- Simplicity and easier installation versus ability to run system remotely from another location, less clutter from cables

FPS of the Cameras

- reduced FPS of cameras so that uploads/downloads are faster and model inference takes less time

No longer have FPGA

- Our latency requirement is 5 seconds, so it’s easier to simply reduce camera FPS

Lessons Learned

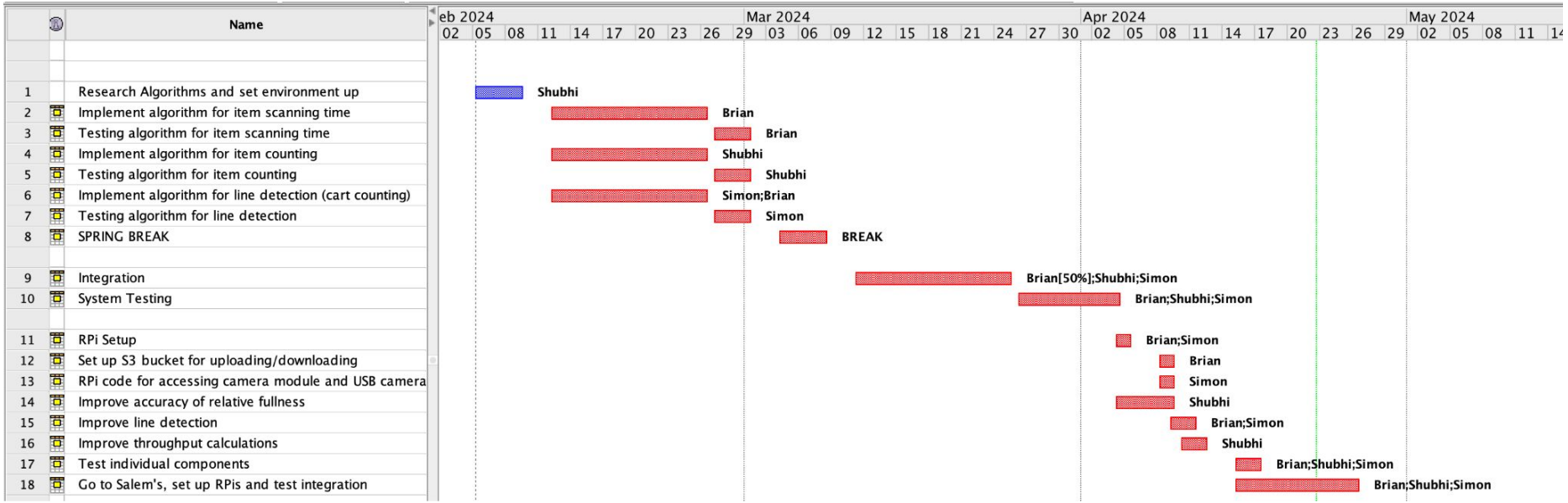
Ethics:

- Employees and customers concerned with cameras
  - Build personal relationships with the employees and customers so less afraid
  - Work with manager and employees to figure out workarounds

Implementation

- Always be ready to adapt to new situations and changes
  - Camera angle changed a lot as we figured out how to setup the cameras at Salem’s
  - Led to some of our models and implementations being very inaccurate - had to adapt

Schedule


Name
eb 2024
Mar 2024
Apr 2024
May 2024
02 05 08 11 14 17 20 23 26 29 03 06 09 12 15 18 21 24 27 30 02 05 08 11 14 17 20 23 26 29 02 05 08 11 14
1 Research Algorithms and set environment up
2 Implement algorithm for item scanning time
3 Testing algorithm for item scanning time
4 Implement algorithm for item counting
5 Testing algorithm for item counting
6 Implement algorithm for line detection (cart counting)
7 Testing algorithm for line detection
8 SPRING BREAK
9 Integration
10 System Testing
11 RPi Setup
12 Set up S3 bucket for uploading/downloading
13 RPi code for accessing camera module and USB camera
14 Improve accuracy of relative fullness
15 Improve line detection
16 Improve throughput calculations
17 Test individual components
18 Go to Salem's, set up RPis and test integration
Shubhi
Brian
Brian
Shubhi
Shubhi
Simon;Brian
Simon
BREAK
Brian[50%];Shubhi;Simon
Brian;Shubhi;Simon
Brian;Simon
Brian
Simon
Shubhi
Brian;Simon
Shubhi
Brian;Shubhi;Simon
Brian;Shubhi;Simon