Where's the Milk?

Project Contributors: Lucky Wavai, Allen Ding, and Takshsheel Goswami

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—For the average shopper who is not able to find the items in their local stores that they desire, the employees that are constantly showing customers the aisles where things belong only to find them out of stock, the businesses losing customers to the ease of online shopping and retail, it seems that the traditional brick and mortar shopping experience needs to be reevaluated. We hope to create a component of the next generation of the in-person experience that will provide real time information to customers and businesses alike. More specifically, we aim to build a platform that incorporates computer vision for inventory management of grocery store aisles.**

***Index Terms*—AJAX (Asynchronous JavaScript and XML), background subtraction, BRISK (Binary Robust Invariant Scalable Keypoints), feature detection, FLANN (Fast Library for Approximate Nearest Neighbors) Matching, Lowe's Ratio Theorem, Brute Force Matching, openCV, RPI (Raspberry pi), SIFT (Scale Invariant Feature Transform), SQLite3, SURF (Speeded Up Robust Features), web application, wireless communication**

I. Introduction

These days, more and more people are turning from in-person shopping to e-commerce alternatives for ease compared to that of traditional brick and mortar stores. According to Fortune, "UBS forecast that some 80,000 retail stores, or roughly 9% of the current total, will shut their doors permanently by 2026. The forecast is based on the assumption that e-commerce as a percentage of total retail sales jumps to 27%—up from 18% now" [2]. There seems to be a trend of local brick and mortar stores closing which also inevitably results in lost jobs nationwide. One potential way to combat this is to provide customers with a reason to do more of their shopping locally.

Current solutions lack real time analysis and/or require heavy infrastructure. For instance, the typical inventory management systems involve scanning products when received at the backdoor, then keeping track of what gets sold at the register. These processes have no comprehensive management of items while in the store such as information about the presence of items on shelves or misplaced products. [3]

Some more advanced solutions involve robotics in grocery stores such as the Marty Robot. Similar technologies maneuver around the store sensing for the desired information such as spills or hazards, but the downsides include high cost of such robotics hardware (i.e a single Marty Robot costs about $35,000), take up aisle space, and lack the ability to perform comprehensive real time analysis as robots do not scan all aisles of the store simultaneously. [4]

We hope to work towards a new generation of brick and mortar stores with real time automated inventory management systems, navigation systems, and more. Specifically, this semester, we will attempt to take steps towards that store of the future by developing a platform which incorporates computer vision technology to transmit real time data of an aisle status to an application for store operators.

II. Use-Case Requirements

Our primary goal is to build a platform that accurately detects and provides real time data on aisle and object states to a web-interface. Therefore, our primary focus is on accuracy and time/latency..

One use case requirement is to provide a system with at minimum 75% accuracy in detecting the presence of all of the expected store items on aisle shelves. The qualifier of "expected" is to differentiate from items that are on the shelf that were displaced from another aisle. For our primary goal, such items will not be a requirement, but we hope to tackle this area in a possible reach scenario. This use case requirement will be analyzed in our testing phase called the Presence Test.

Another use case requirement is to provide a system with at least 80% accuracy in detecting activity within the aisle, specifically the motion / activity of people within the aisle. This use case requirement will be analyzed in our testing phase called the Motion Test.

Another use case requirement is for all of the information to be processed, transmitted, and displayed to the user device within 15 seconds of detected motion in the aisle i.e. someone entering and leaving an aisle.

III. Architecture and/or Principle of Operation

We are building our platform primarily using the Python programming language with some exception i.e. the web application front end. On top of Python, we are using various libraries packaged under the conda-forge openCV requirements package. The primary libraries of note are openCV, numpy. We will use the MOG2 class within openCV for background subtraction and test and select from the SIFT, SURF, and BRIEF algorithms within openCV for object detection. *See Fig 2.* Python with the addition of these libraries provides an efficient development environment for our detection use case requirements as well as backend technology and communication between devices for the rest of our system.

The system will process in a manner similar to a finite state process. Initially, the system is in an idle state awaiting aisle activity. After a person is detected within the aisle the system enters the motion state. Once there is no longer any motion detected, the system enters the item detection state where the presence of all items will be updated. Once processing is complete, the system resets to the idle state. *See Fig 1*

On the user-facing side, the application presents aisle and shelf information that will continuously update as changes are detected. *See Fig 2*

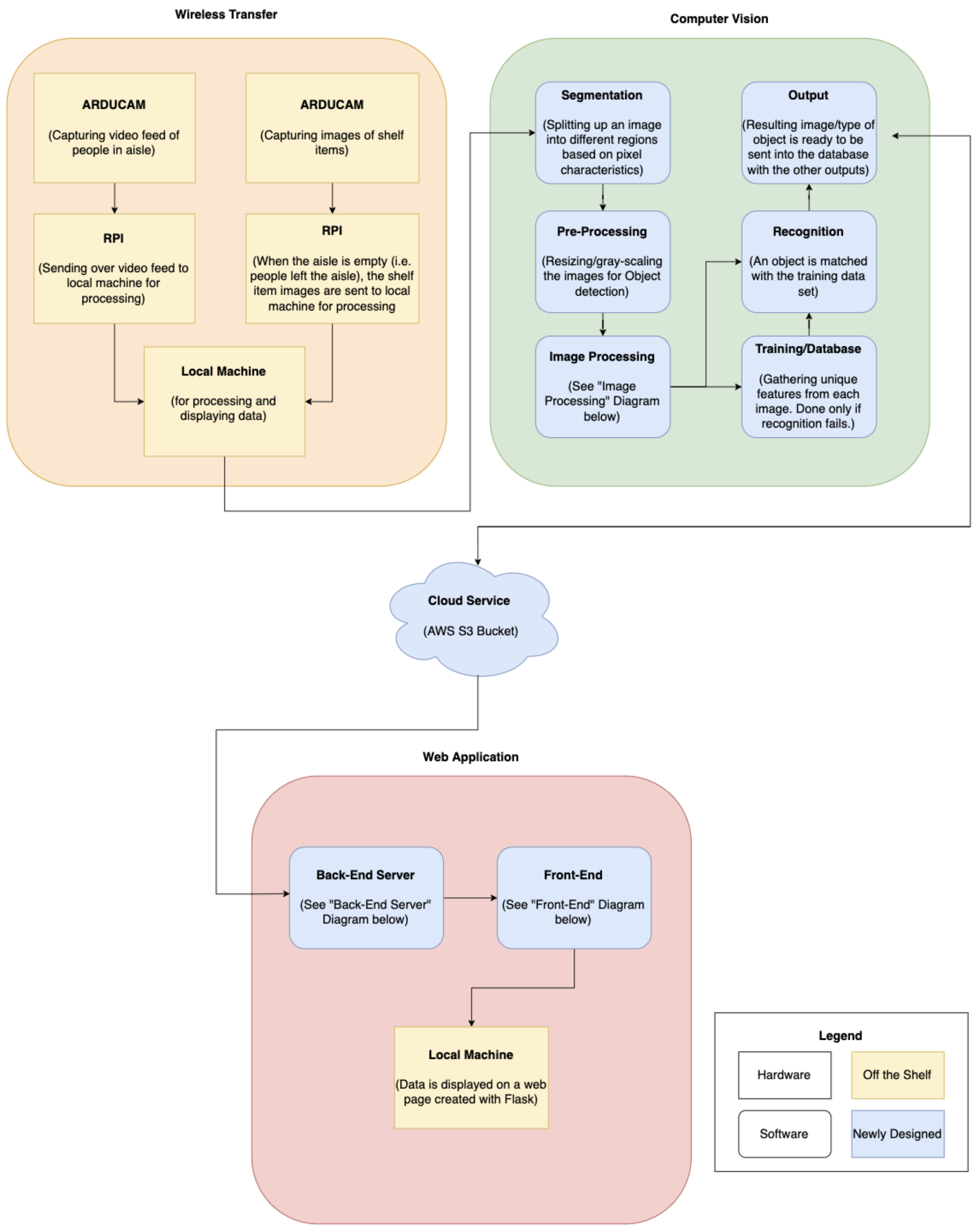

Wireless Transfer
ARDUCAM
(Capturing video feed of people in aisle)
ARDUCAM
(Capturing images of shelf items)
RPI
(Sending over video feed to local machine for processing)
RPI
(When the aisle is empty (i.e. people left the aisle), the shelf item images are sent to local machine for processing
Local Machine
(for processing and displaying data)
Computer Vision
Segmentation
(Splitting up an image into different regions based on pixel characteristics)
Output
(Resulting image/type of object is ready to be sent into the database with the other outputs)
Pre-Processing
(Resizing/gray-scaling the images for Object detection)
Recognition
(An object is matched with the training data set)
Image Processing
(See "Image Processing" Diagram below)
Training/Database
(Gathering unique features from each image. Done only if recognition fails.)
Cloud Service
(AWS S3 Bucket)
Web Application
Back-End Server
(See "Back-End Server" Diagram below)
Front-End
(See "Front-End" Diagram below)
Local Machine
(Data is displayed on a web page created with Flask)
Legend
Hardware
Off the Shelf
Software
Newly Designed

Fig. 1.Overall System Block Diagram

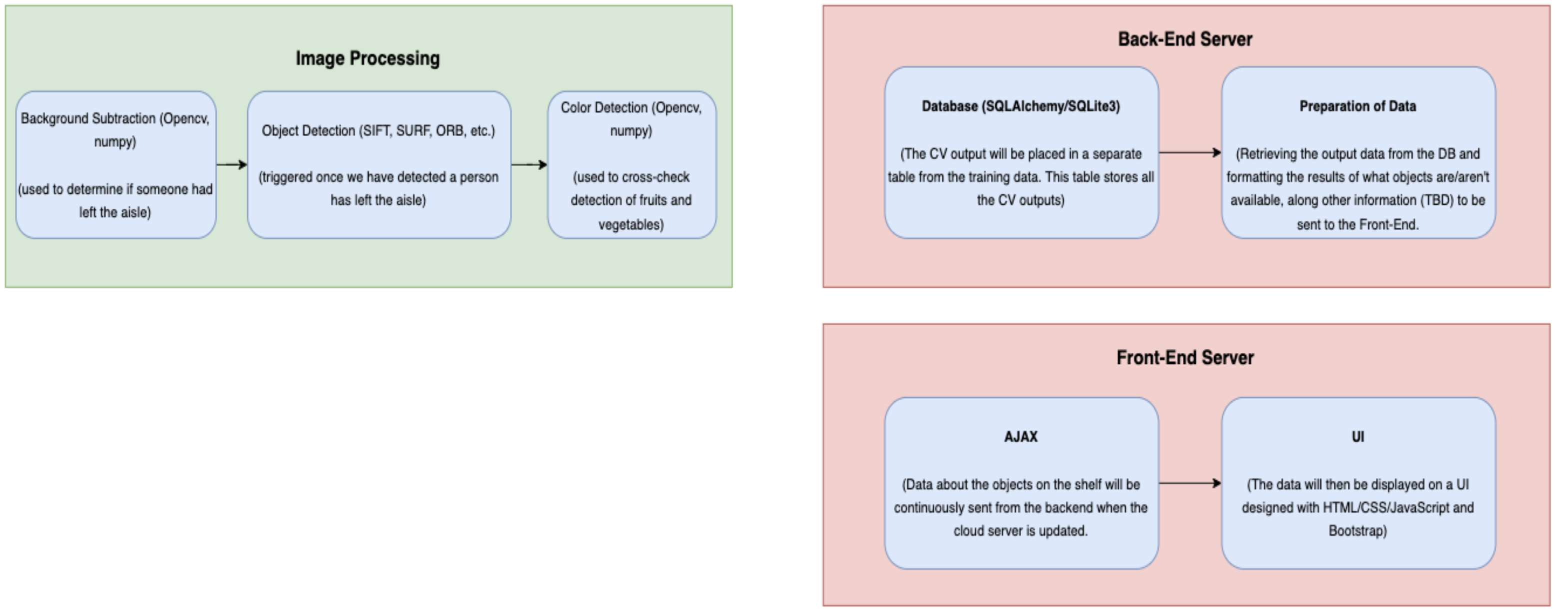


Image Processing
Background Subtraction (Opencv, numpy)
(used to determine if someone had left the aisle)
Object Detection (SIFT, SURF, ORB, etc.)
(triggered once we have detected a person has left the aisle)
Color Detection (Opencv, numpy)
(used to cross-check detection of fruits and vegetables)
Back-End Server
Database (SQLAlchemy/SQLite3)
(The CV output will be placed in a separate table from the training data. This table stores all the CV outputs)
Preparation of Data
(Retrieving the output data from the DB and formatting the results of what objects are/aren't available, along other information (TBD) to be sent to the Front-End.
Front-End Server
AJAX
(Data about the objects on the shelf will be continuously sent from the backend when the cloud server is updated.
UI
(The data will then be displayed on a UI designed with HTML/CSS/JavaScript and Bootstrap)

Fig. 2. Sub-modules within Computer Vision and Web Application blocks

IV. Design Requirements

For this system, we have 3 primary subsystems, namely the wireless data transfer, the image processing, and the web application. Through our use case requirements, we have our speed and accuracy measurements for our whole system.To achieve these overall, we'd need higher accuracy and speed subsystems to integrate it all into our initial use case requirements. Initially, our wireless transfer system would receive images from our camera and transfer it to a local machine for processing. Our camera should be able to capture clear images of people and things 5 feet away, and must be compatible for use with a raspberry pi which we shall use for a wireless image data transfer. This transfer should not take longer than 1-2 seconds (preferably near instantaneous) so we have some time prior to our allocated 15 seconds through the use-case when we can work on our image processing.

For our image processing, we have 3 primary processes/algorithms that we would need to execute at any point. Namely, motion detection (background subtraction), feature extraction (SIFT), and color detection. Through research, we have been able to determine that our feature detection algorithm could take roughly 3-6 seconds, so we would like to not have to spend very much time on the 2 other image processing algorithms. Motion detection/background subtraction tends to be near instantaneous (1 second at most), and so does color detection, so those should not add a lot of time to our initial timing requirements. Besides the timing requirements, we need to have a high degree of accuracy as well. Our motion detection algorithm must have over 80%

accuracy in determining motion in the aisle, and should not provide false positives as far as possible (ideally 0). We would also like to have at least a 75% accuracy with the feature detection algorithms, although correct execution of algorithms that are well documented, eg: SIFT should do the trick.. Once this processed data has been collected, we need to present it in a viable format. That is where our web application comes in.

For our web application we need a speedy framework specifically for display of data. We aren't performing much computation in this step, so our framework need not be extremely comprehensive (eg: django) and we may veer towards others that are lightweight and faster for the specific purposes of receiving and displaying data. For this web application, we'd also need a database that could store processed image data and also work in conjecture with the python image processing that we shall perform. With the current timing, this should leave us about 2-5 seconds to spare from our use-case requirements, while meeting and exceeding the accuracy standards involved.

V. Design Trade Studies

A. Raspberry Pi 4

Raspberry Pi is an affordable microprocessor / computer device that is about the size of a wallet or credit card. The pi module is programmable, and can be used as a controller for other devices. Due to its form factor it can serve as a gateway to IOT device like applications. These specifications make it an ideal candidate for the camera subsystem of our project.

Other options could include using our own laptops or computers, devices from the arduino, or a device such as the jetson nano. One reason laptops and jetsons are not reasonable to meet our project goals is in part because they are expensive devices that provide more computing power than we need. Therefore, a cheaper alternative is desirable. Another reason laptops are not desirable is that it is too large to incorporate on store shelves. The arduino is a cheaper option, but it is also a limited option. The arduino is essentially just a controller for other devices. We want to attempt some onboard processing using the raspberry pi's and transmission that would otherwise require additional complexities if we used an arduino.

One drawback is the limited supply of raspberry pi's in the market. We currently only have one raspberry pi. This is addressed in the risk mitigation portion of this report.t

B. Raspberry Pi FB2 Camera - Arducam

Raspberry Pi cameras can take good quality images and are fully programmable. Another benefit with using the raspberry pi camera is integration with the raspberry pi device itself.

We will avoid the risk of hardware incompatibility that may come with certain webcam or other camera devices. We will also avoid the risk of cameras that require more complexity to control or may be limited in functionality altogether for the purposes of our project.

One drawback of the Arducam is that the current lens we have is a fisheye lens. Fisheye lenses cause some distortion in footage. This is addressed in the risk mitigation portion of this report.

C. Raspberry Pi Power Supply

One goal is to make the system as unobtrusive as possible within the use case environment. Therefore, we are using a separate power supply for the camera subsystem.

The alternatives were battery power or connecting to a laptop or computer device. The laptop or computer device connection defeats the purpose of using a raspberry pi in place of a traditional large computational device to begin with i.e. bulk and cost. The battery powered option would require constant replacements. We want our system to be as set and forget as possible to increase the ease of installation and maintenance

D. SIFT, SURF, or BRISK - Object Detection

SIFT stands for Scale Invariant Feature Transform. That means SIFT can be used to detect objects with a degree of invariance or degree of tolerance to objects that are scaled differently or in different orientations. For our use case, that could meen items that are further back in the shelves, items that are toppled over or replaced in an orderly fashion by people. Therefore, to meet accuracies of above 75% for object presence detection, we would benefit from such an algorithm

One drawback to SIFT is that it is an algorithm that incurs a lot of latency. From research paper [9], it is evident that among similarly peer reviewed alternatives, SIFT is orders of magnitudes slower in total processing time. For instance it averages about double the processing time of SURF.

However, among the alternatives, SIFT time and again out performs in terms of matching accuracy. Since these algorithms run up to about 3-5 seconds in research papers [5], [6], and [9] , and our use case requirement is 15 seconds of total processing time we hope to remain compliant to user expectation and deliver a high accuracy robust system.

Nevertheless, SURF is twice as fast as SIFT, and BRISK is on average faster than SURF according to research paper [9]. Thus, if the latency does become a problem, we will make a pivot after our benchmarking and testing phase as described in subsection title Presence Test in the Testing section of this report

E. MOG Background Subtraction

We decided to use background subtraction for detecting motion within the aisles, more specifically MOG2 background subtraction. Background subtraction is a simpler process of object detection than other methods such as HOG or YOLO because it is essentially just a difference of two images rather than an algorithm to generate and match keypoints and descriptors.

In addition, compared to other background subtraction alternatives such as MOG or GMG, MOG2 performs 3 times faster than MOG and 10 times faster than GMG according to research paper [7]. MOG2 is also capable of upwards of 89% accuracy which exceeds our use case requirement of 80%. [7]

VI. Color Detection

The object detection algorithms we are testing to incorporate into our shelf detection portion - SIFT, SURF, BRISK - primarily use corer and blob detection. Therefore, it may prove difficult to differentiate between items such as oranges and apples especially since those initial algorithms will be processing gray scaled images for speed requirements. Thus, we will use a color detection algorithm rather than an alternative such as building a neural network from scratch. A full NN approach for this edge case may be unnecessarily complex and introduce other risk factors into our system.

VII. Flask

Flask is a lightweight web framework that is developed using python. Since the rest of our system is primarily built using python this reduces complexity. [10]

An alternative we initially considered was the Django Web framework. Django has more configuration requirements and features that go beyond the scope of what we need for the user interface of our project. The focus of our system is not in the web application portion, therefore a lightweight option with configurable features / components such as Flask is desirable. [10]

One drawback however is that Flask does not have a built-in administrative tool which comes with frameworks such as Django. Therefore the development portion will lack in the ease of creating and managing instances of models and dummy data for presentation and testing [10]

VIII. SQLite Database for Object Detection

SQLite is a lightweight database system that supports the ACID principle. This makes SQLie ideal for embedded systems which would serve our project well if we are able to process the object detection on the camera sub system and still mee processing time requirements for our system. The liteweight, configuration free, self-contained , and transactional nature of SQLite enables such implementation. SQLite also works with python thus reducing the complexity of our overall system. [11]

The ubiquitous nature of an ACID based database schema is crucial for an inventory management system that is focussed on localized aisle items. Therefore desirability of partitioning at the cost of consistency is not an issue. In addition, we will not require high traffic requests for the object given the specifics of application for our object detection. [11]

IX. System Implementation

Our project can be seen as four submodules integrated together. These four submodules are:

- Wireless Transfer
- Computer Vision
- Web Application
- Cloud Services

Note: See *Figure 1* and *Figure 2* above for reference throughout this section.

Before we delve into each submodule, we will discuss how the aforementioned parts will be integrated together. To start, we will have a continuous video feed monitoring the aisle for human activity. Once a person is detected in an aisle (using openCV, specifically background subtraction), the RPI/Laptop that was used with people detection will ping the other RPI/Laptop to run object detection (i.e. through SSH). Now, we will use our RPI/Laptop and camera for object detection to take images of the items on the shelf. Once an image is taken, we will use our object detection algorithm to examine the images and send an output detailing what items were on the shelf. Processing and running our people/object detection algorithms will be tested in two ways: on the RPI and on our laptops. The former is because if our user requirements are met (i.e. latency), then we do not have to integrate our data with the cloud (of course, our web application will still be hosted on the cloud). This will remove a layer of difficulty during implementation stages. The latter will be ultimately chosen if the latency from processing on the RPI is too slow. Thus, our output will be tested in potentially two ways: continuously running AJAX from our web application to the RPI or sending the output to the cloud and retrieving the data from AWS through our web application. Once the data has reached our web application, we will add the data to our database, which will trigger our app to send the updated data to the frontend for the user to view. Now, let’s examine each submodule in detail.

A. Wireless Transfer

The wireless system will consist of the following components: RPI/Laptop, camera, lens, and camera ribbon. There will be two sets of the aforementioned components. One for people detection and one for object detection. To set up the system, we will have the RPI and camera connected by a camera ribbon. Then, the camera will have a lens attached to it. For the people detection, a live continuous video feed will be running and once a person is detected using openCV, the other system will be triggered (likely through SSH). Then, the people detection system will take a picture of the shelf for processing. The way that the video feed and pictures will be conducted will be through programming the RPI to control the camera. If we choose to do our computer vision algorithms on the RPI, we will have the data ready to be sent from the RPIs to our web application. If we do not, we will be sending the data to a laptop for processing, through some wireless protocol (i.e SSH, TCP, etc).

B. *Computer Vision*

For our computer vision step, we will have two parts: people detection and object detection. For people detection, we will be using MOG2 background subtraction. For object detection, we will be testing SIFT, SURF, BRIEF, etc. At this stage, we are unsure about which algorithm is ideal for our system (i.e. meeting our requirements), but we will be testing and evaluating multiple algorithms. For object detection, we will extract key points and descriptors from the image that we are analyzing and compare those points with our training data set. We will then take note of the items that exceeded a certain threshold, which means that we have found a match. The image will then be discarded and the list of items that were found will then be sent to the cloud or straight to the web application.

C. *Web Application*

For the web application, we will be using the Flask web framework with SQLite3 as our database. The motivation behind choosing these two were mainly because of functionality and speed. For our use cases, Flask provides more than enough (i.e. user authentication, forms/form validation, database compatibility, HTML templates, etc). At its core, Flask is very bare bones and we add modules to it as we need. This lightweight service compared to other services (i.e. Django) is much faster. Flask may not be as comprehensive as other web frameworks, but for our use case, it's perfect. As for SQLite3, just like Flask, it's very lightweight. To meet our user requirements, we are going to need fast access and management with our database. SQLite can handle up to 281 Terabytes, which far exceeds our project scope [12]. Once the data from the computer vision algorithms are sent over to the web application, we will process that data and store it in our database in a table where the name of the product, the existence of the product, and a unique identifier will be located. After this, we will trigger a call to the frontend where we will upload the new data to the webpage. The way the fronted will be designed will be through three main pages: login/registration page, homepage, item content page. For login/registration, that is self-explanatory, but for the homepage, it'll have various tiles on the page (similar to a grid view). The tiles represent an aisle in the store and when clicked, the user will be redirected to a page that has all of the items in that aisle shown as present or not (item content page). The frontend will be styled with Bootstrap, HTML, CSS, and JavaScript.

D. *Cloud Services*

For the cloud services, we will be testing AWS, specifically S3 buckets. These buckets will contain the data from a given aisle (i.e. if an item is present or not in an aisle). Our web application will be hosted on AWS. If we choose to use AWS to store our data, that's where we will be retrieving our data for our web application.

X. Test, Verification and Validation

Seeing as our project has 3 primary sections we shall divide our testing of project requirements into the 3 parts as we have discussed in our block diagram, where we look at the computer vision portion, wireless transmission, and the web application as separate entities and individually test those. After successful testing of these separate components, we shall test the integration of our components with full system tests after our full implementation.

A. *Presence Test - Tests for item detection on shelf*

For our Presence Test, we will test the shelf item detection accuracy. From research paper [5], SIFT, ORB, and SURF reached accuracies up to about 50-70% after undergoing several comprehensive tests. From research paper [6], SIFT reached accuracies up to 100% in some unit tests. Therefore, our goal is to be able to detect the presence of items on shelves above 75% accuracy by using either SIFT, SURF, or BRIEF feature matching algorithms. Our data set will be multiple images containing one or more of the intended shelf items. We will split the data set into 25% testing and 75% training groups (not training as in that of a machine learning algorithm but just for explanation sake). We will compare the training set to the test set to benchmark accuracy ratings for our project, and to help us select which algorithm to proceed with. We will also use the metrics to tweak parameters such as the filtering thresholds, or the bounding on the algorithms (BRIEF can perform unbound or bounded detection given a threshold to decrease processing time).

B. *Motion Test - Tests for motion detection in aisle*

For our Motion Test, we will test detection accuracy of people within the aisle. From research paper [7], we found that background subtraction methods can reach accuracies up to about 90%with averages around 80%. Therefore, our goal is to be able to detect the presence of items on shelves above 80% accuracy. Since our motion detection will be a video feed as opposed to the image capture used for item presence detection, our testing process will be different. We will run multiple tests of different motion states: (1) steady state - a person standing in an aisle simulating browsing items, (2) a person moving through the aisle to simulate traffic / motion through the aisle, (3) no people in the aisle.

C. *Transmission Test - Tests for Wireless Data Transfer*

For our transmission test, we're going to test out our implementation at first with dummy data. Initially, we're going to test our Raspberry Pi access point and check that we can connect our computers to it. Next we'd have to provide our raspberry pi with a dummy file e.g., a CSV, and we'd test to check if we could access that from our raspberry pi by sharing that to a local machine via FTP or wifi.

Once those work, we shall have our capture device store image data in our raspberry pi which we shall share through the same FTP or wifi procedure. In this step we shall have to test that the camera can store usable data in our pi, by unit testing the apparatus. We shall put different objects in front of our camera and check in our pi for the images that should be present.

D. Experience Test - Tests for web-app user experience

For our experience test, we want to test the quality of the user experience which is ultimately the customer facing portion of our project. There is no benefit to information that is unintelligible or difficult to obtain no matter the breadths and depths undergone during the data acquisition stage. We will qualitatively test the web - application experience with voluntary surveys of various aspects of the application (1) Navigation, (2) Data Visualization, (3) Usefulness of information, (4) Overall Ease of Use of the Application, (5) User recommendations. The user recommendations will be used to make any possible changes to the application if we see a trend in the requests.*Experience Test - Tests for web-app user experience*

E. Overall Test - Test the overall system performance

For our overall test, we want to primarily test the latency and integration of the entire system. Firstly, we want to ensure that from start to finish of the motion activity, to object detection, to data presented to the user works qualitatively as expected i.e. the transitions between the various states of the system work as desired.

Then, we will test to ensure that we meet the 15 second user requirement by walking in and out of the aisles and making adjustments to the items while also retesting the Presence and Motion Tests with the complete system.

XI. Project Management

A. Schedule

Please refer to *figure 2* for a detailed description of our schedule, which includes milestones and team responsibilities.

B. Team Member Responsibilities

While everyone's primary responsibility is listed below, we will all be assisting each other in their respective roles if they need help. Additionally, we will all participate in the wireless transfer process and cloud management (AWS). Allen will be responsible for setting up the wireless system (i.e. connecting the RPI, camera, and lens together and taking pictures and videos). Then, everyone will assist in having the data sent to a local machine for analysis (i.e. Computer Vision Algorithms). As for the cloud, everyone will partake in creating, implementing, and managing our cloud usage and integrating that with our components.

<u>Lucky:</u>

- Object-Detection Algorithm
    - This process entails testing and comparing various algorithms (i.e. SIFT, SURF, BRISK, etc.) to determine which one is most optimal for meeting our requirements. Additionally, Lucky will be in charge of deciding how to pre-process the images, what specific features to extract, and how we will determine a match for an item to our training data.

<u>Allen:</u>

- Web Application Development
    - The web application is built on Flask, and this will be supported by a SQLite3 database. Allen will be in charge of the entire stack of the web application (i.e. user authentication, database management, data transfer from AWS to the application, UI, etc).

<u>Takshsheel:</u>

- People-Detection Algorithm
    - Takshsheel will be in charge of implementing and testing various people detection algorithms to ensure that we reach our requirements on this end. Once this is completed, as a team, we will work together to determine how to ping the other RPI/laptop to start running object detection.

C. Bill of Materials and Budget

Please refer to *Table I* for our bill of materials and budget.

D. Risk Mitigation Plans

Below are the critical risk factors in our design followed by our plans for how we will manage these risks:

- Potentially having to work with two different camera systems considering the widespread scarcity of Raspberry Pi's right now.
    - We already have a set of a Raspberry Pi, a camera module, and a wide-angle lens. Thus, for our other set, we may have to pivot to using a laptop/webcam as our second set.
- We are unsure how well the ARDUCAM lens will work with object/people detection (i.e. we believe that the lens is a fisheye lens and the media feed might be distorted).
    - We will test out the fisheye lens and do more research on the ideal lens for object detection/order another type of lens if need be.

- Integration
    - Our team has little/no experience with computer vision and RPIs/ARDUCAMs prior to this project, thus we need to set aside a few weeks just for testing and integration when our components are completed.
- Object-Detection - Detecting objects of similar size and color with different labels (i.e. A cereal box vs. a pancake box)
    - Utilize feature extraction to distinguish between color and the certain text (i.e. the largest and boldfaced text is most likely the title).
- Cloud Integration - we have limited experience with cloud management and downloading/uploading data from the cloud.
    - After doing research, we concluded that using AWS buckets would be ideal to store and retrieve our information. Also, a plus is that we have used AWS before. However, integrating with the cloud will still be challenging as our cloud experience is little.

XII. Related Work

Throughout the web, there are many *standalone* projects or products that implement human detection or object detection.

Currently, there are no projects or products that are using computer vision, wireless transfer, and a web application for the purpose of keeping track of store inventory.

From our research, there was a product that was similar to ours in the sense that it uses openCV to examine objects in an inventory system before it leaves the factory. This product is called the *Smart Factory Defect Detection System*. [8]

In summary, this product continuously monitors the products in the manufacturing process and searches for defects. For the detection algorithm, this product mainly uses YOLO with an CNN to find flaws (i.e. defects on metal surfaces, such as scratches, dents, etc). The size and location of the defect(s) are then reported. [8]

We observe here that there are underlying similarities not just in our software and hardware, but in the use case as well for inventory. While we are not looking for defects in a product, we are both examining inventory and using openCV to do so.

XIII. Summary

In summation, our design consists of four main components (Note: the “etc.” are there because we are listing out the main hardware/software components, not all of them).:

- Wireless Transfer
    - (i.e. RPI, ARDUCAM, local machine(s), etc.)
- Computer Vision
    - (i.e. Object/people detection using SIFT/SURF, BRISK, etc.)
- Web Application
    - Flask, SQLite3, etc.
- Cloud services
    - AWS (i.e. S3 bucket)

Essentially, our system will have a continuous video feed (using the RPI and ARDUCAM) that once a person is detected in a given aisle (through background subtraction), our other RPI/laptop will be triggered to use our object detection algorithm. Once the objects in the aisle are scanned and evaluated (i.e. checking for the existence of an item), information about the item (i.e. name of the item, if it exists, etc.), will be uploaded to an AWS S3 bucket (or our web application will directly use AJAX to retrieve the data). Next, comes the web application. The web application will be constantly retrieving information from the S3 bucket (or RPI), parsing the information, storing the information in our database, and uploading that data to the web page for the end user (i.e. manager of the store) to view. Updates to the web application will be asynchronous because every time the AWS bucket is updated, the database will be updated, which means the webpage will be updated. For the RPI, once the RPI is ready to send its output, our web application will be ready with AJAX to grab the data. The cloud process eliminates unnecessary calls to refresh the data of the page when nothing new is updated. In short, this would be a more efficient use of AJAX compared with continuously calling on the RPI.

The main impact of our project to our stakeholders is that we are presenting them with an inventory management system with real time analysis while keeping costs to a minimum. This is compared with other major inventory management systems on the market today that have much higher costs and lack of real time analysis (See Section IX).

As with any system, there will be challenges. The main challenges that stem from our project are the following:

- Our team is inexperienced with object/people detection, wireless transfer, and cloud management. This will not hinder our progress as learning is part of the project.
- Integration:
    - Our anticipated latency may not be reached

given all of the components we have to integrate (we originally had the cloud as a reach goal).
  - Being able to successfully connect each component together.
- Given that many objects on the shelf can have similar sizes, colors, names, etc., being able to distinguish objects at our desired accuracy rate will be a challenge.

Glossary of Acronyms

AJAX - Asynchronous JavaScript and XML
RPI – Raspberry Pi
BRISK - Binary Robust Invariant Scalable Keypoints
FLANN - Fast Library for Approximate Nearest Neighbors
SIFT - Scale Invariant Feature Transform
SURF - Speeded Up Robust Features
YOLO - You Only Look Once
CNN - Convolutional Neural Network
AWS - Amazon Web Services
SSH - Secure Shell

References

[1] IEEE, *IEEE Author Center: Author tools*, Accessed on Jan 17, 2022, [Online]. Available: Source Link

[2] Wahba, Phil. "Another 80,000 Retail Stores Will Close by 2026, Says UBS." Fortune, Fortune, 5 Apr. 2021, Accessed on February, 7, 2022, [Online]. Available: Source Link

[3] Campbell, Jeff. "How Do Grocery Stores Keep Track of Inventory?: The Grocery Store Guy %." The Grocery Store Guy, 29 Aug. 2021, Accessed on February, 7, 2022, [Online]. Available: Source Link

[4] Gallucci, Nicole. "Marty the Robot: Non-Essential Worker." Mashable, Mashable, 23 Nov. 2020, Accessed on February, 7, 2022, [Online]. Available: Source Link

[5] Karami, Ebrahim et al. "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images, " *Faculty of Engineering and Applied Sciences, Memorial University, Canada*, Accessed on February, 7, 2022, [Online]. Available: Source Link

[6] Sun, Yan Li et al."Performance Analysis of SIFT Feature Extraction Algorithm in Application to Registration of SAR Image," *Department of basic experiment, Naval Aeronautical and Astronautical University, Yantai264000,China*, [Online]. Available: Source Link

[7] Marcomini, L. A. et al."A Comparison between Background Modeling Methods for Vehicle Segmentation in Highway Traffic Videos," [Online]. Available: Source Link

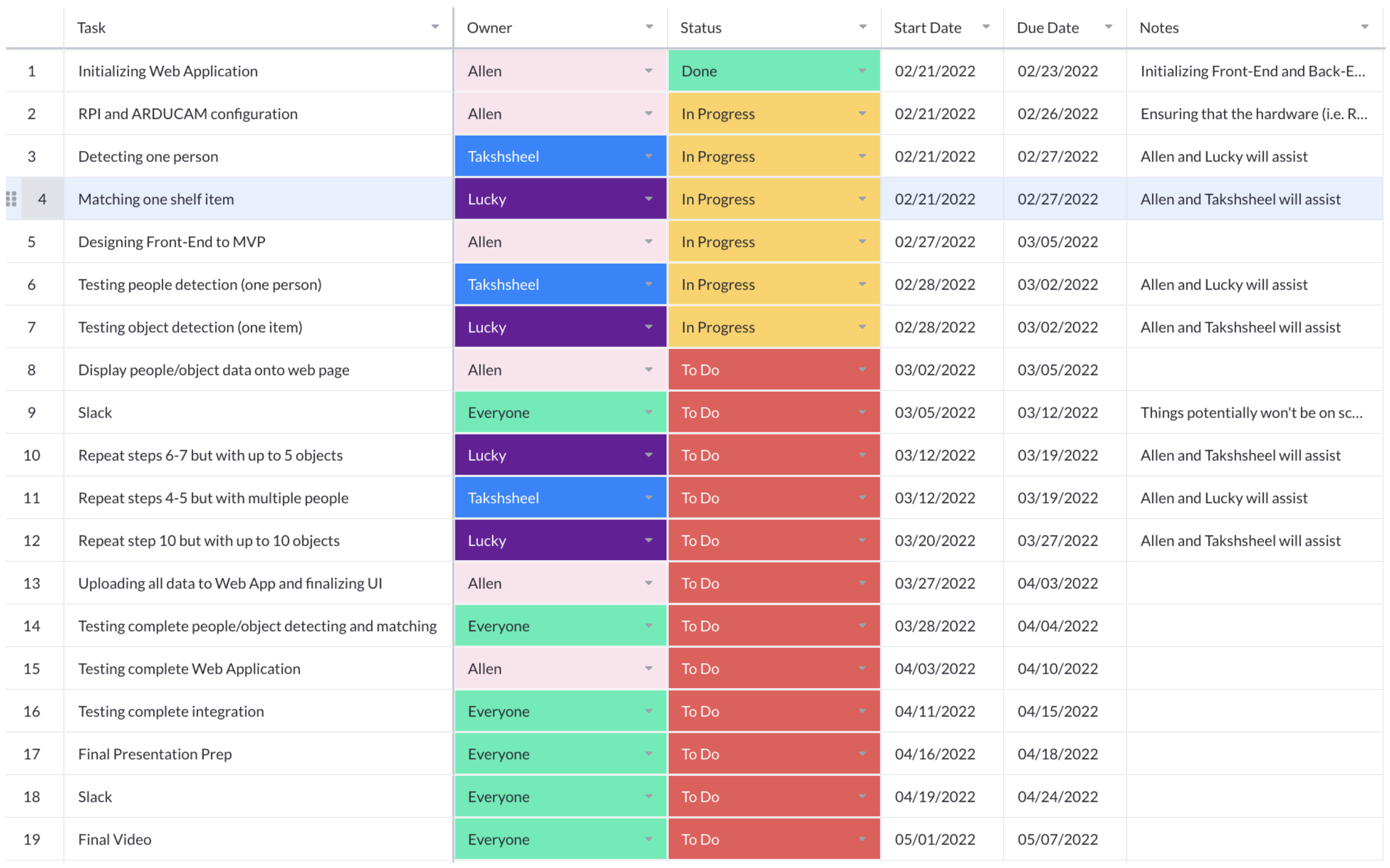
[8] Park, Sang-Hyun, et al. "Deep Learning-Based Defect Detection for Sustainable Smart Manufacturing." *MDPI*, Multidisciplinary Digital Publishing Institute, Accessed on February 25, 2022, [Online]. Available: Source Link.

[9] Tareen, Shaharyar, et al. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," *IEEE*, Accessed on March 1, 2022, [Online]. Available: Source Link

[10] "Flask vs Django in 2022: Which Framework to Choose?" Hackr.io, Accessed on March 2, 2022, [Online]. Available: Source Link.

[11] "Sqlite Reviews &amp; Ratings 2022." *TrustRadius*, Accessed on March 2, 2022, [Online]. Available: Source Link

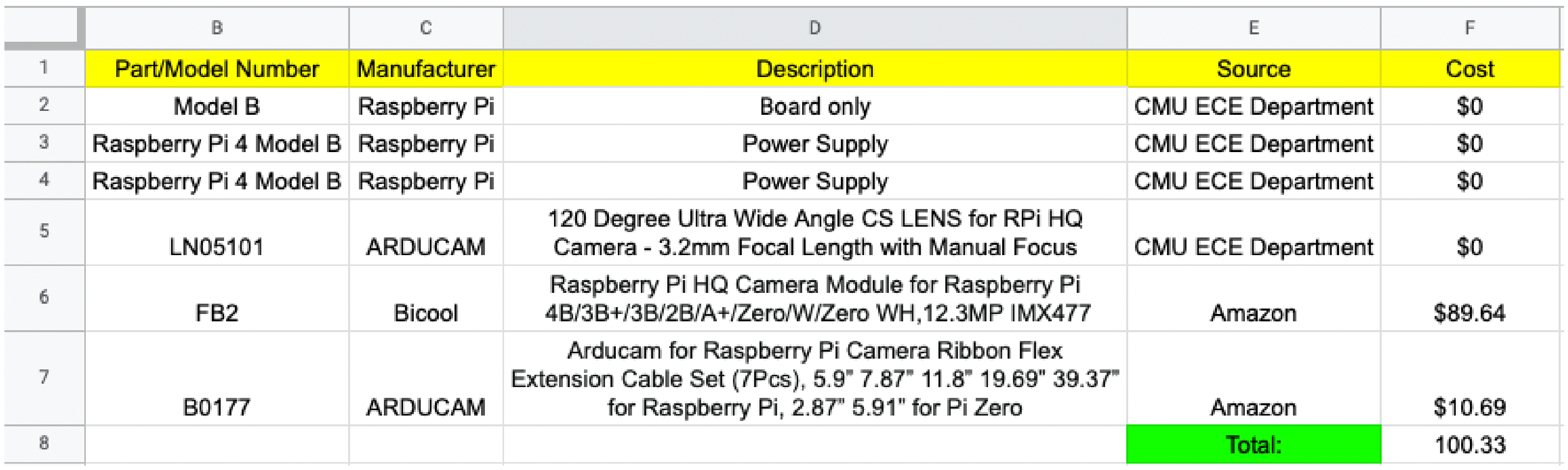
[12] "Appropriate Uses For SQLite." *Appropriate Uses for SQLite*, https://www.sqlite.org/whentouse.html#:~:text=SQLite%20supports%20databases%20up%20to,will%20support%20281%2Dterabyte%20files. Available: Source Link



	Task	Owner	Status	Start Date	Due Date	Notes
1	Initializing Web Application	Allen	Done	02/21/2022	02/23/2022	Initializing Front-End and Back-E...
2	RPI and ARDUCAM configuration	Allen	In Progress	02/21/2022	02/26/2022	Ensuring that the hardware (i.e. R...
3	Detecting one person	Takshsheel	In Progress	02/21/2022	02/27/2022	Allen and Lucky will assist
4	Matching one shelf item	Lucky	In Progress	02/21/2022	02/27/2022	Allen and Takshsheel will assist
5	Designing Front-End to MVP	Allen	In Progress	02/27/2022	03/05/2022	
6	Testing people detection (one person)	Takshsheel	In Progress	02/28/2022	03/02/2022	Allen and Lucky will assist
7	Testing object detection (one item)	Lucky	In Progress	02/28/2022	03/02/2022	Allen and Takshsheel will assist
8	Display people/object data onto web page	Allen	To Do	03/02/2022	03/05/2022	
9	Slack	Everyone	To Do	03/05/2022	03/12/2022	Things potentially won't be on sc...
10	Repeat steps 6-7 but with up to 5 objects	Lucky	To Do	03/12/2022	03/19/2022	Allen and Takshsheel will assist
11	Repeat steps 4-5 but with multiple people	Takshsheel	To Do	03/12/2022	03/19/2022	Allen and Lucky will assist
12	Repeat step 10 but with up to 10 objects	Lucky	To Do	03/20/2022	03/27/2022	Allen and Takshsheel will assist
13	Uploading all data to Web App and finalizing UI	Allen	To Do	03/27/2022	04/03/2022	
14	Testing complete people/object detecting and matching	Everyone	To Do	03/28/2022	04/04/2022	
15	Testing complete Web Application	Allen	To Do	04/03/2022	04/10/2022	
16	Testing complete integration	Everyone	To Do	04/11/2022	04/15/2022	
17	Final Presentation Prep	Everyone	To Do	04/16/2022	04/18/2022	
18	Slack	Everyone	To Do	04/19/2022	04/24/2022	
19	Final Video	Everyone	To Do	05/01/2022	05/07/2022	

Fig. 3. Schedule: Milestones and Team Responsibilities

TABLE I. BILL OF MATERIALS



	B	C	D	E	F
1	Part/Model Number	Manufacturer	Description	Source	Cost
2	Model B	Raspberry Pi	Board only	CMU ECE Department	$0
3	Raspberry Pi 4 Model B	Raspberry Pi	Power Supply	CMU ECE Department	$0
4	Raspberry Pi 4 Model B	Raspberry Pi	Power Supply	CMU ECE Department	$0
5	LN05101	ARDUCAM	120 Degree Ultra Wide Angle CS LENS for RPi HQ Camera - 3.2mm Focal Length with Manual Focus	CMU ECE Department	$0
6	FB2	Bicool	Raspberry Pi HQ Camera Module for Raspberry Pi 4B/3B+/3B/2B/A+/Zero/W/Zero WH,12.3MP IMX477	Amazon	$89.64
7	B0177	ARDUCAM	Arducam for Raspberry Pi Camera Ribbon Flex Extension Cable Set (7Pcs), 5.9" 7.87" 11.8" 19.69" 39.37" for Raspberry Pi, 2.87" 5.91" for Pi Zero	Amazon	$10.69
8				Total:	100.33