Arushi Patel
Pallavi Bannai
Shreyas Gatuku

Cart-i B

Final Report
18-500:ECE Design Experience
Fall 2018
12/12/2018
V 2.1

Table of Contents:

1. Project Description: 1
2. Design Requirements: 1-3
3. Architecture: 4-5
4. Design Trade Studies: 5-8
5. System Description & Depiction: 9-16
6. Evaluation/Testing & Results: 16-21
7. Project Management: 21-23
8. Related Work: 23-24
9. Conclusions & Future Work: 24
10. References: 25

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

1. Project Description

For our capstone project, we designed an autonomous shopping cart that is able to identify and follow a customer as they shop. To do so, we used image processing, a robotics kit, sensors, and color coded targets to ensure that the cart follows the correct human and does not run into obstacles. As intended, the goals of our project were iterative in nature such that the first milestone was to have the cart follow the human in a straight path and stop when the human stops. The next milestone was to have the cart follow the human through turns. And, our last set of milestones consisted of having the cart avoid obstacles and do a basic search for the target human, if he/she disappeared from sight.

The application of the technology developed for this project maps to a variety of uses. It would be helpful in not just in adding an aspect of convenience to shopping but also in aiding those with medical conditions who cannot push a cart by themselves. Even more so, the system implementation can be extracted and applied to any application where there is need to have an object follow a human.

2. Design Requirements

The overall goal of our design consists of three major components: identify the target human/shopper, have the robot move accordingly, and avoid obstacles in its path. Specifically we will go more in depth into each of the three aspects.

Overall Interfacing:

There are design requirements for the interfacing between the three components, to make the three parts come together into one functional whole. We need to make sure that both the independent calculations and the transfer of the data between the three aspects and the centralized processor happen at a rate that is comparable to the rate of movement and walking of a human, which is 1.4 meters/second as mentioned earlier.

In order to achieve this, we would like the entire software computation block to take at most **120 ms** on average from the time the target is processed to the robot moving in response. Looking at the larger picture, there are several design requirements as to how the entire system should function as whole. Cart-i B will have to follow a human through a mock grocery aisle and stop when the human stops. When the human moves, the robot should recognize the movement within **50 ms** and act on it within **100 ms**. Additionally, it should be able to follow the human through turns and implement basic obstacle avoidance. If there is an obstacle in between where Cart-i B is, and where the target is, the cart should re-route its plan such that it does not hits the obstacle and still reaches the person, assuming there is a viable path within

the area. Cart-i B should also be able to make sharp turns from where the human did to avoid the chances of it getting lost. The cart should also stop when the human goes to make pivot turns. Moreover, if Cart-i B is lost it should search for the human.

A major part of our project is to have Carti-B be user-friendly. Our requirements for user-friendliness includes having users not have to do anything extra in order to have the cart follow it. For example, Eli is a similar product to Carti-B that follows the user by having the user constantly talk to it, something a user usually would not do. A stretch goal of ours was to have Carti-B follow the user without any extraneous beacons or markers. With this, the user should just be able to enter the grocery store, select a cart, and start shopping without having to attach anything to them. In order to achieve these overall we have outlined our specific subsystem goals below:

Image Processing Module:

For the image processing side, the goal is to not just identify the target human but also estimate the distance of the human's main target from the robot to within **0.5 m,** such that the pid is properly calibrated for it stop within a safe distance from the human. Another requirement is that the image processing algorithm will have to be able to detect the human at a speed that is reasonable and consistent with human movement, for which the average walking rate is 1.4m/sec. The algorithm should be resilient to different lighting conditions, angles of the human, and various close range distances.
Specific quantitative requirements include:

- Under **100ms** to go through entire iteration of a loop → detect circle & set pid values
- Under **50ms** when either target found (checks 20 times in a second)
- Must always stop on viewing red
- Must predict human's location of sharp turn within **25cm**
    - Average grocery store aisle = 8 feet = 244 cm of which 10% = 25cm
    - We went with 10% because most aisles can fit three carts, thus each takes up approximately 33% of the aisle. Being off by 10% would allow for the cart to be in the general correct area as the human such that it can re-lock onto the human, and this margin of error/hitting another cart or human is small enough that it can be prevented with obstacle detection

Robot Control Module:

To have the system physically follow the human, we need a robotic platform that interfaces well with our microcontroller to have refine control of the motors, as well as something small and adaptable to act as a shopping cart. The main requirement for the robot control module is that given a distance (in meters) or angle (in degrees) or a speed to move/turn by from the image processing, the robot can reliably perform those movements. The movement should have minimal latency but also be smooth as possible.The main logic of this module is

calculating the correct motor values to send using PID based on the targets identified from the image processing module. Specific requirements include:

- Must be able to follow human without running into human
- Travel at least as fast as the average human walking speed (1.4 m/s) Perform pivot turns to within 15 degrees of accuracy
- Perform hard-stops when necessary
- Perform distance commands to within 0.2 meters of accuracy

Object Detection Module:

We required that our sensors detect two main scenarios for obstacles - obstacles that block the path temporarily and those that remain in the path. For both these scenarios, we would like Carti-B to stop only if the obstacle *directly* blocks the path between Carti-B and the target. This means that out of the four sensors at the front of the cart, **two adjacent** sensors must be detecting in order to have Carti-B stop and detect and obstacle. Specific quantitative requirements include:

- Detect obstacle within **30 cm** range when it directly obstructs path to target, both when moving forward and backwards
    - Continues to move if an obstacle is detected in periphery
- Can move around an obstacle that is stagnant, continues to move in intended path if obstacle is introduced and then leaves
- No more than **30ms** delay from when obstacle is introduced to when iRoomba gets command to stop. This is calculated by taking the average speed of Carti-B, 0.28 m/s and the obstacle detection range of one foot.

The individual requirements are summarized in the table below:

Image Processing	Robot Control	Object Detection & Path Planning
● Under **100ms** to go through entire iteration of a loop → detect circle & set pid values ● Under **50ms** when either target found (checks 20 times in a second) ● Must always stop on viewing red ● Must predict human's location of sharp turn within **25cm (**Average grocery store aisle = 8 feet = 244 cm of which 10% = 25cm	● Must be able to follow human without running into human ● Travel at least as fast as the average human walking speed (1.4 m/s) ● Perform pivot turns to within 15 degrees of accuracy ● Perform hard-stops when necessary ● Perform distance commands to within 0.2 meters of accuracy	● Detect obstacle within **30 cm** range when it directly obstructs path to target, both when moving forward and backwards ○ Continues to move if an obstacle is detected in periphery ● Can move around an obstacle that is stagnant, continues to move in intended path if obstacle is introduced and then leaves ● No more than **30ms** delay from when obstacle is introduced to when iRoomba gets command to stop. This is calculated by taking the average speed of Carti-B, 0.28 m/s and the obstacle detection range of one foot.

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

3. Architecture

Our autonomous shopping cart has the sub-goals of identifying the location of the human, moving towards the human, and avoiding any impeding obstacles. These sub-goals are handled by different subsystems, Image Processing Module, Robot Control Module, and Obstacle Detection Module, as introduced above and shown in Figure A below.

Image Processing Module

- Hardware platform: Raspberry Pi
- Input: Camera frames from Camera via CSI interface
- Logic: Blurring, Dilation, Thresholding, Clustering, and Segmentation algorithms
- Output: Center coordinates and size of target (Human)
- Basic Description: Gets video frames from Camera, converts to hsv, establishes a red and green threshold and sees if a valid circular shape is found and acts accordingly (shown in Figure C & G)**.**

Robot Control Module

- Hardware platform: Raspberry Pi
- Input: Center coordinates and radius of target from Image Processing Module
- Input: Distance sensors state via UART from Arduino
- Logic: PID calculations, object location identification, and path planning
- Output: Motor values to iRoomba
- Basic Description: Based on the coordinates of the center of the circle & its radius, uses pid to establish motor commands (shown in Figure G)**.**

Object Detection & Path Planning Module

- Hardware Platform: Arduino Uno
- Input: Distance to closest obstacle from 5 ultrasonic sensors
- Logic: Threshold data, Encode data into packet
- Output: Packet representing activated sensors to Robot Control Module via UART serial, interrupt fired to Robot Control Module
- Basic Description: Checks if any ultrasonic sensor has violated the obstacle range threshold of 30 cm. If so, fire an interrupt to the Raspberry Pi, send packet representing which ultrasonic sensors have been fired to Raspberry Pi and reroute the robot.

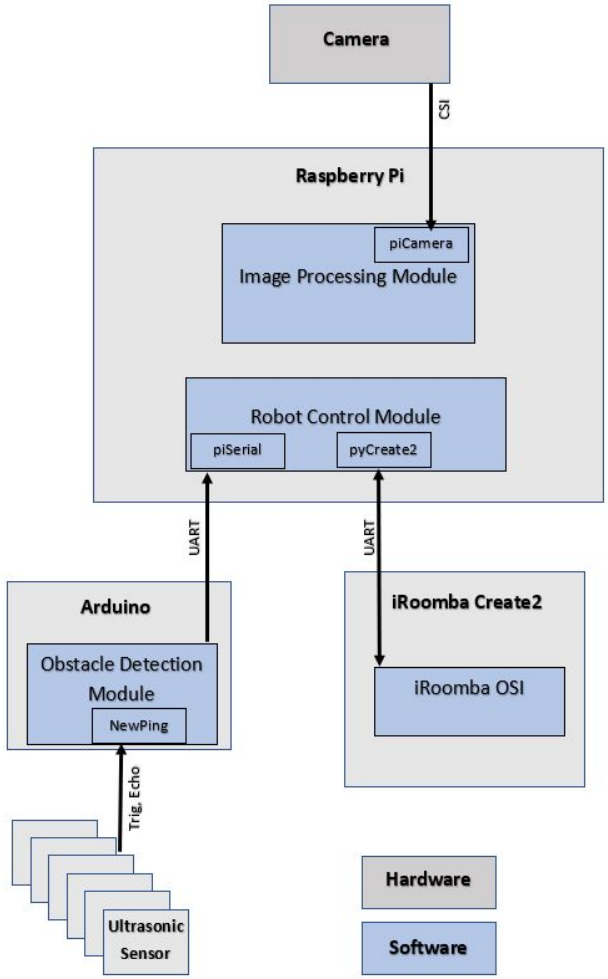


Figure A: Overall System Architecture (Hardware + Software)
Algorithmic specifics shown in Figure G

4. Design Trade Studies

Image Processing

Throughout the design process, we learned that it is just as important to decide on what not to implement and choose, as it is to decide which decisions to move forward with. In terms of the image processing part of the project there were three general approaches that we were considering: training and using a neural network, using a histogram based algorithm, using pixel-independent operations. The benefit to neural networks is that there are already frameworks out for algorithms to implement to find humans. But we decided not to go down this approach because Shreyas had previously implemented a neural network to detect a person based on color and the algorithm ran very slowly. Furthermore, for a neural network to detect

humans, the algorithm must look through each area of a frame and identify all humans, which is not essential to our application so we decided to not pursue this means of image processing. The next option we considered was using histogram based algorithms such as OpenCV HOG, which is an abstracted version of implementing our own neural network. However, this is also slow for the operations look at a small set of pixels at a time as the operations are heavily pixel-interdependent.

We decided to go with our last idea which is to use thresholding to identify a target. In this implementation, we will have to make the compromise of adding a target to the human, but we feel that the benefits, which include the speedup in the computation as well as a means of distinguishing which specific human the cart should follow, outweigh the costs. For this method, we decided to define what our target is and on each frame perform erosion and dilation to reduce followed by thresholding based on the color of our decided target. Then based on the threshold, we use the OpenCV minEnclosedCircle function. There are still aspects, such as the the minimum enclosed circle functions that are not pixel independent, but the computation for those are not as taxing as the HOG and neural net based computations. Moreover, before we completely decided to move forward with this approach, we made sure to implement it to ensure the circle was consistently being drawn on the edge of the target on Arushi's back as she walked. We also tested in a variety of locations: an empty hallway, a room with a lot of backlighting and a busy area - in front of La Prima in Wean. An image of our target we tested is included below.

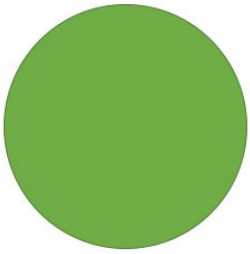


Figure B: Main Target Used for Human Detection

Another aspect of discussion throughout the design of our project was how to implement sharp turns. Originally, our idea was to have a circle on the right and left sides of the back such that as the human turned, the areas could be compared to identify in which direction the human was turning, for example: if the right circle was getting smaller faster, the human is turning to the left and vice versa. We realized though that if one of the circle gets covered though, there would be no way to detect the turns, so we considered having an array of 5 circles, such that as long as any two circles were seen, there dimensions could be compared. However, we realized that doing so would not be feasible, because often the human takes right turns quicker, and the small change in distance between the circles, and their relative areas, as the human turns would not be large enough to accurately identify the direction and where exactly the human turned. We next looked into image processing algorithms that have been used for this kind of purpose, and learned about homography. Specifically, we were going to make the target a checkerboard, and use a OpenCV function to identify the checkerboard, from which we could find the angles of the

lines of the board to establish in which direction the human was turning. Similar to our overall image processing design studies, the algorithm for finding the checkerboard is pixel dependent and thus takes a very long time, which made this idea infeasible. Lastly, we settled on using targets on the front and sides of the jackets to be able to both stop the cart and make it take right turns from the relatively same spot as where you made that turn. Also, because red is a common color, to prevent turning false positives, we incorporated a concentric blue circle such that if the centers of the two circles are within 25 pixels, only then will the code actually recognize it as the *correct* red target.

Robot Control

During the initial stages of our project, a lot of thought was put into what kind of robotic platform should we use to hold the cart. We were first debating if we should make the drivetrain from scratch. This would mean buying DC motors, a motor driver board, wheels, power distribution board, and securing materials. Although it is definitely possible to construct this, we realized it would not be worth our time since our project should focus on how we're using a robot for our application and not how we're building the robot. We then researched what are viable off the shelf products that would meet our requirements and serve our applications. It came down to either a Arduino controlled robot kit off of RobotShop.com or an iRoomba developer kit. Both fit our budget and could perform the basic requirements. The robot from RobotShop.com was unfortunately too small to support a decently sized shopping basket and did not have much online support. The iRoomba create was the biggest off shelf kit we could get that had good product support and reputation. We were convinced this would be our best option considering there were examples online of applications that used a raspberry pi to control it as well as clear specifications on how to mount other systems on top of it.

Once we acquired the iRoomba, we focused on how to secure the remaining components like the raspberry pi, arduino, camera, sensors, and basket. Since none of us were mechanical engineers, we were initially unsure how to tie everything together. The main debate was whether to use wood or light metal alloys as the infrastructure of the robot. Since our TA knew of where we could easily find wood and the wood was both light and durable enough, we went with wood for our infrastructure. The metal frames would also have been hard to drill screws into. The last design trade study for the general robot was how to make our system actually replicate a shopping cart. Since the iRoomba could only support 20 lbs (according to its specifications), we had to keep our method light. One idea was to put hooks on the main center pole that would hold shopping bags with few items in them. The second idea was to have a small basket that sits on the mid-level platform. Since the latter is more aesthetically pleasing and closer to our application, we decided to go with that method.

Object Detection

Another major design trade study was the hardware we planned to use. In terms of the object detection sensors, we looked at two main pieces - the Elegoo Ultrasonic HC-SR04 and OSOYOO IR Obstacle detector. The Elegoo Ultrasonic sensor is a polled sensor that have several library functions useful for object detection. Because of its polled nature, it requires the

microprocessor to continuously be probing the sensor, potentially eating into the processor's computational power. The OSOYOO IR sensor, on the other hand, is interrupt based. There is a potentiometer attached that can be adjusted in order to adjust the threshold obstacle distance. Because of this, the sensor only returns a 1 or 0, indicating whether the threshold has been passed. Although the OSOYOO sensor would be better in terms of latency, it greatly reduces the granularity of the obstacle detection. Because our design specification for obstacle detection is to not only detect obstacles but also avoid them, the OSOYOO sensor is not useful for our design.

Although we decided to go with the Elegoo Ultrasonic sensor, the polling nature is something that still needs to be considered. Another design trade off is how exactly we will connect the Elegoo Sensors to our design. In order to initially test the sensors, we have the sensors connected to an Arduino Uno board. One configuration to integrate the sensors into our overall design is to have them connected to the Arduino and have this Arduino communicate with our Raspberry Pi. In this setup, the Raspberry Pi would not have to waste any computation polling the sensors. However, there is an extra layer of communication necessary to go from the Arduino to the Pi. Obstacle detection needs to be real time, so this added communication may delay the iRoomba's response. Connecting to the sensors directly to the Pi would allow for more direct communication but may slow down our overall design, especially since we also want to run image processing on the Raspberry Pi.

General Hardware

Because we had some concerns regarding the processing power of the Raspberry Pi, we looked at several alternatives and weighed the pros and cons. The Raspberry Pi itself has a lot of documentation and libraries, and all of our team members have worked with them before. However, it is fairly difficult to fully utilize the quad core processor. Another board we looked into was the Parallela. This board has a very high fast processor and is often used for real-time image processing. It also has some interesting features, like an FPGA built into the board. The downsides are that there is very little documentation on the board and not much community support. Libraries available on the Raspberry Pi would also have to be ported to the Parallela, adding to the amount of work we would need to do in just setup. The final board we considered was the NanoPC-T3. The NanoPC has higher processing power than the Raspberry Pi, though not as high as the Parallela. The NanoPC also has a good amount of community support and all libraries available on the Raspberry Pi are available on the NanoPC.
Taking all of this into consideration, we will begin development on the Raspberry Pi. If there are any significant issues with latency, we will look into moving our development onto the NanoPC-T3.

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

5. System Description/Depiction

As previously mentioned, our project consists of three major subsystems: image processing, object detection and iRobot motion control.

Image Processing

For image processing we decided to move ahead with using OpenCV to identify the target in the image, without using neural networks. Specifically, our image processing algorithm consists of taking each frame and converting the data from an red-green-blue frame(rgb) to hue-saturation-value(hsv). Then image is threshold such that if a pixel lies in the range of two hsv arrays that we have defined, the pixel in the hsv range is a 255 and all others are 0. Then once this black and white frame is computed, we perform and gather information about the minimum enclosed circle. We realized while doing initial testing that with this method, it is difficult to identify that a human took a turn. We categorized three types of turn-like movements: 1. Moving side to side in an aisle, 2. Pivoting in place to pick up something in an aisle, 3. Turning out of an aisle. The information about the center of the circle and its radius, explained above, is enough to address the first type of turning movements. For turn types 2 & 3, we introduced a second type of target, a red circle that will be on each shoulder of the target. Moreover, because red is a common color, to prevent turning false positives, we incorporated a concentric blue circle such that if the centers of the two circles are within 25 pixels, only then will the code actually recognize it as the *correct* red target.

As soon as the image processing module detects a red circle, by the same means explained above, it stops, which addresses the second type of turning movement. If this red circle leaves the frame, once the circle has left the frame, the cart will move forward to where it last saw the human, stop and then turn in place starting in the direction the human left the frame, to find the human. This is summarized and further depicted in the flowchart below:

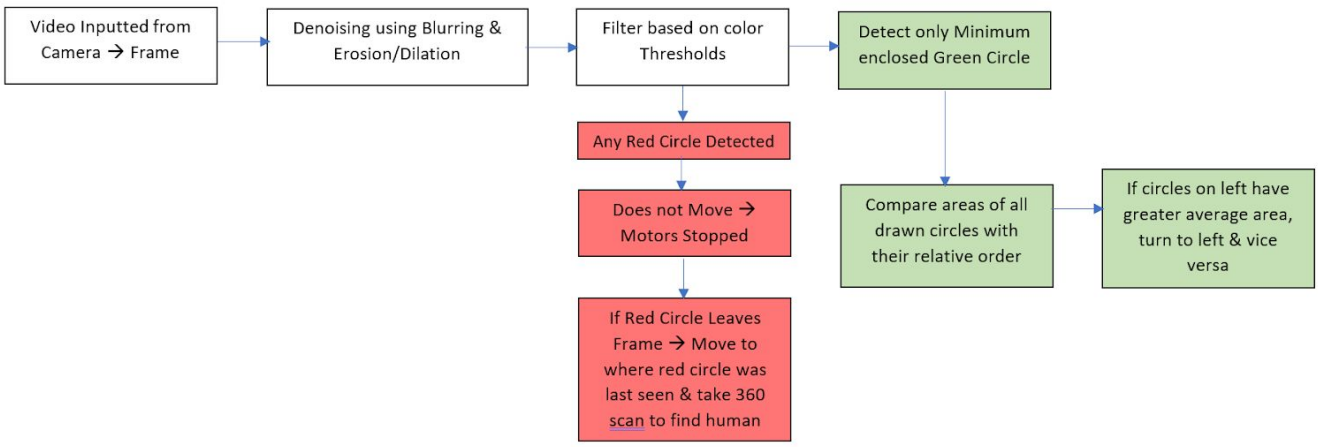


Figure C: Image Processing Flow Chart

Sensors

In the Obstacle Detection Module, we used the NewPing Library. For each sensor, we call a function that sends out a ping from the ultrasonic sensor and converts the amount of time it takes for that ping to return to the distance from the obstruction in centimeters. We want to detect an obstacle within 30 cm in order to give the iRoomba enough time to change its movement. In order to not constantly spam the Raspberry Pi with information from the sensors, we only want to send an interrupt to the Pi when any of the sensors detect something within 30 cm. When this occurs, the path planning algorithm requires information about which sensors detect an obstacle. In order to do this, we use a serial protocol of 5-characters where each character represents whether that sensor has had the threshold violated. If sensor 1 is violated, we send the string “10000,” indicating sensor 1 has been violated. The conversion of this packet is taken care of in our Raspberry Pi. A visual representation of the module internals is shown below.

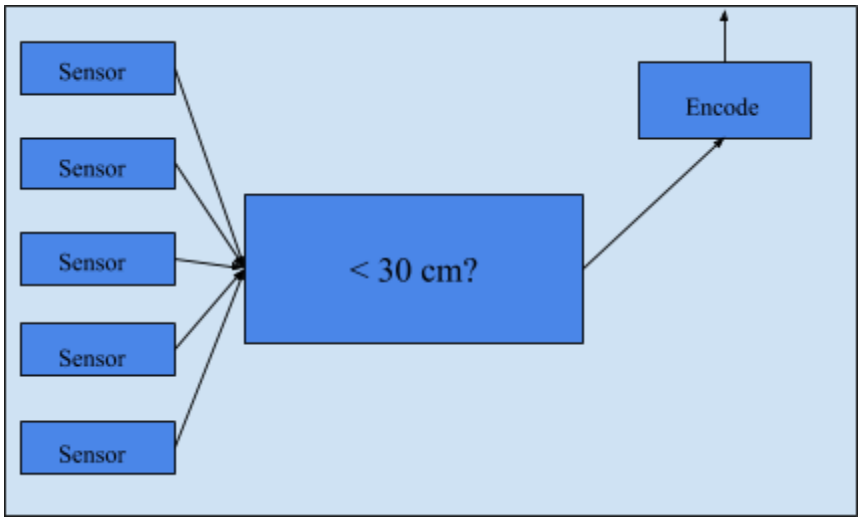


Figure D: Obstacle Detection Module Internals

Path Planning

The Robot Control Module is also responsible for handling the input from the Obstacle Detection Module. The Obstacle Detection Module on the Arduino will provide which sensors on the platform detected an object. The priority of the Robot Control Module is to direct motor commands to follow the human but before it sends motor commands, it will check all sensor data to see if there is a possibility of collision. For example, if the human is towards the right of the picture frame and the robot needs to move towards the right, the logic will check if the front and right sensor values from the Arduino are triggered. If these sensors stay triggered for more than 3 seconds, a new route must be devised. This new route is based on the rest of the sensor values as well. If there is no obstacle detection on the left, then the robot will proceed towards the left by a distance of 0.5 m to get around the obstacle, and then try to proceed to the right again. However if all the sensors indicate an obstacle then the robot will not move until a viable path is found. This algorithm is also visually represented in Figure E and G below. To avoid having the Raspberry Pi constantly check if the sensors are activated (which would waste CPU

time), the Arduino sends an interrupt whenever there's a change in the sensor values. The interrupt will read the incoming serial data and update a global variable in the Pi that keeps track of the current sensor state. Once an obstacle leaves, this variable gets reset to it's default. This saves CPU time by not having to poll the sensors as well as not having to go through all the path planning options when there are no obstacles. While the cart is path planning to avoid an obstacle and another obstacle gets in the way, the cart does try to avoid it as well. This leads to the chance that due to the placement of multiple obstacles and the path planning algorithm, the cart may lose the human, which is another reason why we implemented that if the human is lost, the cart will rotate to search.

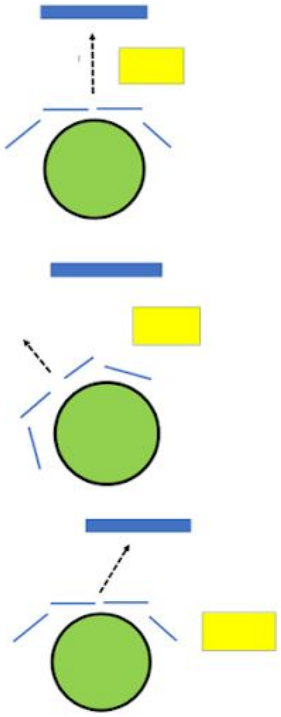


Figure E: Diagram of Path Planning

Robotics

In the Robot Control Module the coordinates and radius are used to find the relative angle and distance change needed to maintain the human in the center of the camera frame. For example, if the given x coordinate is far greater than the frame's center x coordinate then that indicates the human is going to the robot's right. The error in x is sent to the PID algorithm to determine the magnitude of the motor values needed to turn the proper amount. The turn should bring the human x coordinate closer to the center x coordinate. A "golden" radius is found using manual calibration as a reference to how close the robot should be to the human. Currently we are using a 0.5 meter required distance between the robot and the human. If the radius given by the image processing module is smaller than the "golden" radius that maps to 1

meter, then the robot must move forward. The error in radius is also sent to the PID algorithm to determine the magnitude of the motor values needed to move forward or backward. Since the iRoomba can either only move forward/backward or spin at a given time, we added logic for when to prioritize turning over distances and vice versa. By default, turn commands are prioritized over distance commands so that the robot doesn't lose the human as they make sharp or wide turns. However if the distance magnitude is greater than a large threshold, distance commands are prioritized so that the robot can catch up to the human. The motor values are sent to the iRoomba through the pyCreate2 library that provides functions that encode and decode the data for us. The iRoomba uses two motors with each motor power value ranging from -500 to 500. The PID output is scaled to fit within this range.

Integration

The overall flow is if the image processing module identifies the arm targets, the robot does not move unless it sees the arm target exit the frame. If this occurs, the robot moves to where it last saw the human and takes a 360 scan, starting in the direction in which the human left the frame to find the human again. If only the back green target is seen, the robot uses pid to follow the human at a safe distance. For any movement, the robot first checks if moving in that direction will be safe, based on the sensors. Depending on which sensors are high, the robot adjusts its path, moves forward and reevaluates using the same process. If no target is seen, the robot rotates in the direction it last saw the human and continues to rotate until a target is seen again. A flowchart representation of this algorithm in conjunction with the image processing and robotics logic is on the next page as well.

Overall Changes Made to the Design through the Implementation Process:

Throughout we made some important implementation changes which include the placement and number of sensors, establishing the pid motor values based on the radius of the target rather than the area, minimizing jitter and identifying a sound way to implement turns.

Specifically, in terms of sensor placement, we originally began with having three in the front and three in the back but instead resorted to having four in the front so they can cover obstacles directly in front of the robot, as well as near the side and only one sensor in the back for moving backward without hitting an obstacle or wall.

Another overall goal was to minimize the jitter of cart-iB. To do so, we made two changes the first being adding a threshold such that the robot doesn't move unless the change in the target's area is greater than that threshold and the second being change now we calculate our motor controls and distance to based off of the radius rather than area. The change from area to radius really smoothened out the robot's movement, which makes sense because our calculations were based on a linear relationship, and unlike area, the radius does in fact grow linearly/proportionally in terms of distance.

Since our early stages focused more on the image processing side, we wanted to first secure the camera and raspberry pi. We made a wooden platform that screwed into the iRoomba to support the main pole that would house the camera. After initial testing we realized we wanted to make the main pole as short as possible to reduce the tipping force which was

severely affecting the movement but still long enough to see a human from chest level. Later on when we were ready to connect the Arduino and sensors, we added more wooden infrastructure to secure a platform mid-level to hold everything else.

Additionally, upon doing user testing with people outside our capstone group, we realized that there is a period of time that it takes for a person to get used to using Cart-i B. As a result, initially the person moves out of the sight of Cart-i B, so we implemented a feature that if the human is not found, the cart will continuously rotate until a target is found.

The last major set of changes we went through was how to identify that a human is turning to pick something up form the aisle or turning out of the aisle. We originally thought about using a checkerboard and the angle it creates during a turn to do turn prediction, but decided against it as the image processing to identifying checkerboards on openCV is very time intensive (more information given in design trade studies). We ended up implementing a similar image detection algorithm to what we did for the green target but instead looking for the red target. As mentioned in *Design Trade Studies*, upon implementing red circle detection, we added concentric blue circles such that the algorithm only recognizes a red circle with a concentric blue circle as the red target. We saw that in terms of timing, finding the green targets didn't take too long, and thus settled on adding side targets too.

Another minor change made was to remove the cross on the green target.

FIgure F: Final targets used for image processing

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

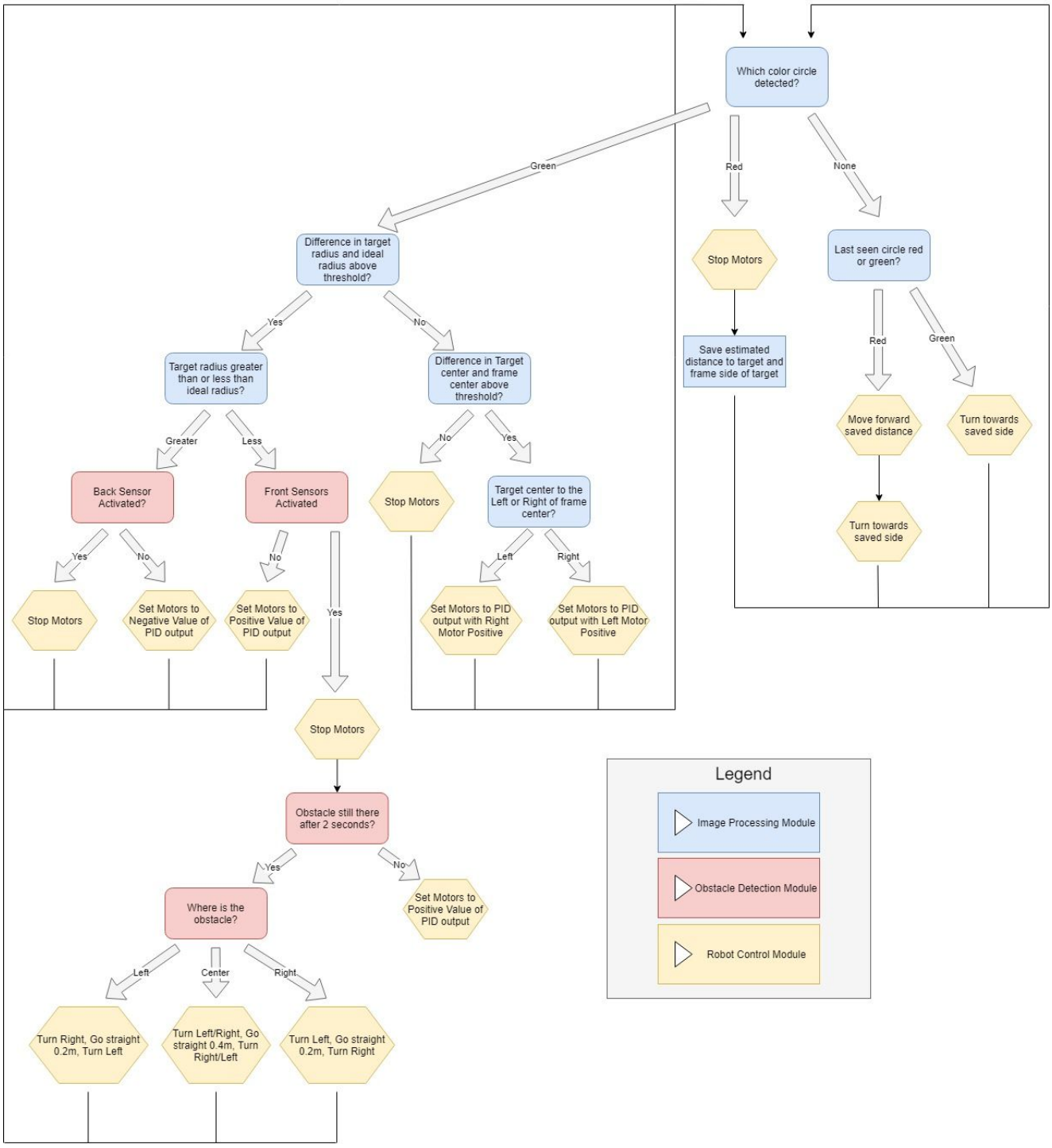


Figure G: Overall Integrated Flowchart

In terms of the overall visual for our system, the following images depict our prototype.

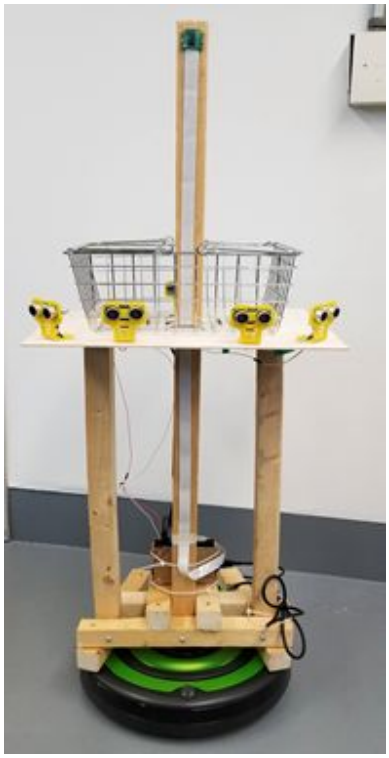
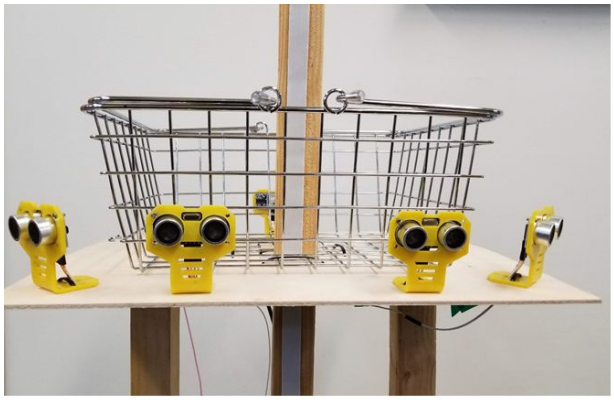


Figure H: Side View of Design Prototype

This is a view of the sensors and hardware attached to our cart:



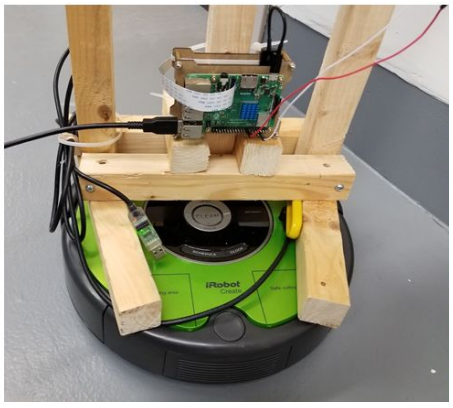


Figure I: Sensors & Hardware (Raspberry Pi)

Figure J: Jacket with green (main) and red (turn) targets

6. Testing and Results

Overall Testing Approach:

Our general approach to testing was similar to our iterative designing and implementation approach. For each component, we had an initial proof-of-concept test, and implementation test and then for all, we had an integration test.

Image Processing
Initial Proof-of-Concept: Wrote out bare bones of "*finding a green circle*" algorithm and tested with a tennis ball video and eventually a printed target example, on Arushi's personal laptop
Implementation Test: Moved the code onto the pi, such that the input was the pi camera and catered the values for better identifying the proper color picked up by Cart-i B's camera. We made sure to display a screen to show what the threshold frame looks like, with a circle drawn based on what the image process model was identifying as the target.
Limits: It is difficult to set the range of color hsv values such that there are no false negatives as such the range must be large enough to take into account the camera looking at the target in a variety of lighting and shadows.

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

Robotics

Initial Proof-of-Concept: Sending basic motor commands to move forward, turn and stop.
Implementation Test: Combined module with image processing and pid algorithm to move based on distance of the target from the robot.
Limits: If the proportional ("P") value in the PID controller is too large, the robot gets very jerky, but if the "P" value is too small, there is a large gap between the human and the robot, making the robot more prone to losing the human. Also there is an inherent jerk in the motion based on the iRoomba's drivetrain design.

Object Detection

Initial Proof-of-Concept: Setting up sensors and testing on Arduino to make sure that the sensors are working correctly/being pinged for the correct distances.
Implementation Test: Setup connection between Arduino and Pi, such that when anything comes within the threshold distance of any of the sensors, an interrupt is sent on the Pi. Next we incorporated the path planning algorithm and integrated this into the rest of the system.
Limits: Cannot have the threshold be too large or the target human will be falsely detected.But if the threshold is too low, there is not enough time for the sensors to do the following: identify all sensors triggered, set the interrupt pin and have the Arduino send the sensor values to the Pi via UART and actually stop the robot before it hits the obstacle.

Integration

We first focused on integrating the initial image processing module with the robotics module. We tweaked the pid values in order to be more appropriate while doing manual testing. Also, while doing manual testing we realized that even while the human wasn't moving significantly, the robot would jitter in place and when the human took a step back towards the robot, the robot would oscillate greatly back and forth. Arushi soon realized that this was because that math for distance we were doing, was in relation to the area of the enclosed circle. However, the distance is not proportional to the area of target, but rather the radius. Thus, once we modified the constants and math to be dependent on the radius of the enclosed circle, the robot's movements smoothened out greatly.

We next added the red circle/turning logic and made sure to test not just the new features but also that the previous functionality was not disturbed. Lastly, we did the same when incorporating the sensors and the path planning. When the sensors were being incorporated, we made sure to test all of the different general cases of what Cart-i B would have to do to respond to certain sensors being set off.

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

Test Plans

Test	Input	Target
Straight Line	Human moving in straight line	Carti-B follows human at safe distance, record success or failure
Red Circle Detection	Human starts showing green target and then shows red	Carti-B moves while green target is visible and immediately stops when sees red target, record success or failure
Distance Prediction when Human Lost	Red circle shown to Carti-B then taken out of frame	Carti-B estimates the distance it needs to move forward, which is compared against a mark on the floor, record displacement from mark
Overall Sharp Turn	Human walks forward, turns and continues forward	Carti-B is able to follow human and relocate them after turn, record success or failure
Smooth Turns	Human moves in general straight direction but not in straight line	Carti-B is able to follow general path, record success or failure
Temporary Obstacle	Obstacle appears in between path of human and Carti-b and is removed	Carti-B is able to stop, not hit obstacle, and then continue towards target, record success or failure
Stagnant Obstacle	Obstacle appears in between path of human and Carti-B and remains	Carti-B stops, not hit the obstacle, and then moves around obstacle

Each test was attempted 10 times.

For the temporary obstacles, we made 2 attempts blocking the front right, 2 attempts blocking the front left, 2 attempts blocking the center, 2 attempts blocking the back, 1 attempt blocking right periphery, and 1 attempts blocking left periphery.

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

Results

The overall results of Cart-i B in action were seen at the Demo on 12/7 and can be seen in this video. The video, along with our demo path, makes sure to highlight each of the main features we implemented: https://www.youtube.com/watch?v=NZGGJV0h-k0

Test	**Result**
Straight Line	10 successes
Red Circle Detection	10 successes
Distance Prediction when Human Lost	+15.5 cm displacement average, 690.25 variance
Overall Sharp Turn	10 successes
Smooth Turns	10 successes
Temporary Obstacles	10 successes
Stagnant Obstacles	10 successes

From these results, we can see that we were able to succeed in our metrics. Our goal for distance prediction from the Image Processing module was +30 cm from the human's location, so we succeeded in that area as well.

Metrics

The methods we used to evaluate our system measurements are summarized below. All times were recorded using the time.time() function in python.

System Measurement	**Method**
Movement	Measured the time the cart took, unloaded to move one meter - starting with the cart standing still 0,5m from the target
Image Processing	Record time before all image processing and at end of image processing, taking the difference

Sensors	Ping time - record time before pinging all sensors and at end of pinging all sensors, taking the difference Response time - measure amount of time it takes for an obstacle to cause Carti-B to stop, take average over 10 tests
Overall Loop	Record time at beginning of loop and at end of loop, taking the difference
Weight Limit	Periodically add objects to basket and run Carti-B, looking for significant changes in the motion to indicate a weight overload

System Measurement	**Value**
Movement	0.28 m/s average (unloaded)
Image Processing	Neither green nor red target found - 50ms Green target - 0.15ms Red target - 0.13ms
Sensors	550ms to ping, 1.2 seconds to respond on average
Overall Loop	0.15ms
Weight Limit	2 kg max

In terms of metrics, our average speed was of 0.28 m/s, significantly lower than our goal of 1.4 m/s. This was due to the limitations on the robot itself in terms of kick-back caused by the wooden mount on top of the iRoomba. In order to mitigate these affects, we lowered the speed at which Carti-B moves.

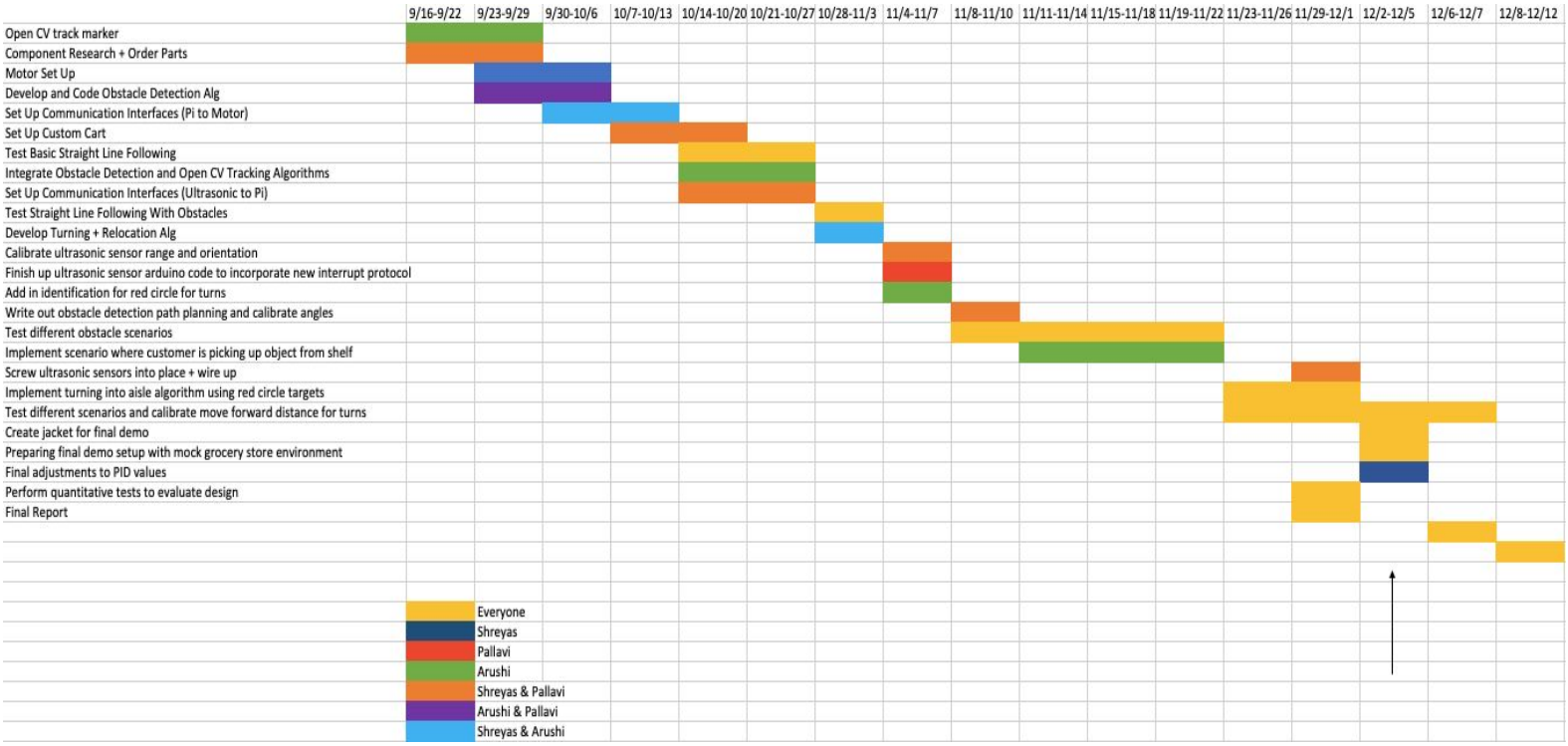
For image processing, we were able to meet our latency goal in all three scenarios of target detection and overall loop.

For the sensors, one ping of all sensors took 550 ms. The 10 attempts to block with an obstacle resulted in an average response time of 1.2 seconds. This average is quite a bit higher than our design requirement 30 ms. Although we do see an effect in Carti-B's ability to stop in time of an obstacle in very close range, we are able to stop in time of an obstacle placed not in extreme close range.

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

Our weight limit was tested by gradually adding items to the cart. When Carti-B had 2 kg or more in the cart, the weight caused the cart to swing back and forth too much, which inhibited proper identification by the camera/image processing module. This was not caused because of the robot's specifications, but rather because the cart is higher in the air and causes more imbalance.

7. Project Management



As shown above, in general the distribution of work is that Arushi worked on the image processing aspect, Pallavi worked on the sensors and the integration of them with the raspberry Pi and Shreyas was responsible for the components related to the robot. The portions of the project that we are working on together are the interfacing aspects between our separate parts.

Arushi Patel
Pallavi Bannai
Shreyas Gatuku

A more descriptive breakdown of the workload is given below:

i. Image Processing Module + initial research/design: Arushi
ii. Robotics Module + PID + initial research/design: Shreyas
iii. Wiring Sensors + Arduino + Sending Sensor Info Module + initial research/design: Pallavi
iv. Path Planning Design: Pallavi, Shreyas & Arushi
v. Path Planning Module: Shreyas
vi. Path Planning Module Refining Modification & Testing: Shreyas & Arushi
vii. Building Frame: Ben, Arushi, & Shreyas
viii. Ordering Parts: Pallavi
ix. Finalizing Wiring & Placement of Sensors: Pallavi

Budget:

Item	**Cost**
iRoomba Create 2	$228
Raspberry Pi Kit	$55
Raspberry Pi Camera	$25
MicroSD	$7.79
Camera Extension Cable	$6.29
Raspberry Pi Extension Cord	$8
Raspberry Pi Battery Pack	$19
Ultrasonic Sensors	$20
Arduino	$0 ($30)
Basket	$12
Perfboard	$6
Jumper Cables	$7
Wire Sheath	$10
Ultrasonic Sensor Mounts	$11

Plastic Standoffs	$10
Jacket	$37
Extra iRoomba Battery	$32

Total = $490

Risk Management:

A potential risk factor we faced was problems with integrating the separate components. Although we could do our best to test the individual components, it was hard to predict what problems we would face when putting the components together. To minimize this risk, we allocated more time for this part of the project. Moreover, instead fully developing each aspect, we integrated early once we have a basic image processing algorithm and robot human-following algorithm. This way we ensured that we have a basic model implemented before further developing aspects such as turns and obstacles.

A risk we had around the time of midpoint demos was coming up with our final plan for addressing turns and path planning, but had no guarantee of it working. Originally, the plan was to implement the path planning and then turns. However, we made sure that both features were being worked on in parallel, such that when we were having problems with the sensors, Arushi was able to get the turns working and integrated into the cart. In the meantime, Pallavi fixed the sensors, and the path planning was soon integrated. Working in parallel, and integrating frequently really helped us in minimizing our risks.

Additionally, in terms of the demo, we ran the risk of someone coming in with a pant color that is similar to the target color. To minimize this, Arushi spent a lot of time trying to further narrow the target color ranges, but because the hsv range must cover seeing the target in a variety of lightings it is hard to further narrow the range down. We also had the risk of not testing much with other people, so we made sure to do a few trial runs with friends before deciding as to if for our demo we will be wearing the jacket or let the people who come to see our project wear the jacket. A challenge we had not initially faced was the cart losing the human as much. When we tested it, we got used to how to cater our walk to prevent the cart from losing us, but when others came into test it, there was a turn that everyone would keep losing the cart, so we implemented a feature that if the cart loses the human, it rotates in place in efforts to find the human. This addition proved to be quite helpful in demos, where we had a lot of people using the system for the first time.

8. Related Work

Some similar products to Cart-i B include the Dash Robot, Eli Robot and Walmart has filed a patent for a similar robot. The Dash Robot has been displayed at technical conferences

and it uses many expensive components such as a LIDAR, and 3D camera. It has the added benefit of having more features for shopping such as it scans an item as you put it in. However, our idea is different in that we are focusing on developing technology that can have an object follow you whether it be a cart in a grocery store in this case, or just a cart that can follow you when you're walking home. Walmart has filed a patent for a similar design of a robot on the bottom of a shopping cart but there have been no prototypes seen. A company called Emart in South Korea recently started testing a robot called Eli, but Eli finds the human to follow based on voice recognition which means now that a shopper would have to constantly be talking to have the cart follow, which does not seem like it would be natural behavior. For our prototype, you do need to have the identifying targets but its a one time cost of putting it on/taking it off, which we feel is better than constantly having to talk to your cart.

As you can see there are other projects that are to solve similar problems as Cart-i B is but there are some major differences. At a production level, we would expect that price of a product like Cart-i B would sell around $350 and from what we can see, because the other products are focusing on multiple aspects of the shopping experience and have expensive components such as LIDAR and a 3D camera, we expect our solution to be significantly cheaper. (At this time we can only estimate the costs of our competitors as we cannot find any on the market currently.) Moreover, an added edge to our product is that we are implementing a user friendly solution such that during the process of shopping, a shopper has to minimally change actions.

9. Conclusion & Future Work

Conclusion

Overall, we are very pleased with what we were able to accomplish this semester, but there would still need to be a lot of work for Carti-B to be ready for production. Although all of us came in with overall relevant skills to this project, we really learned a lot not only in terms of the technological aspects such as image processing or obstacle detection, but also in learning how to scope out and go about designing an open ended project. We started from trying to think of an application of using an FPGA for image processing to wiring sensors and implementing path planning, which we had initially not thought about. Another learning lesson for us was to always have a small goal for each week and make sure it is completed. We worked in parallel as much as possible, such that when the obstacle detection implementation was delayed due to difficulties with the sensors, we were ready to pivot and start testing turns in the meantime. Some of the work we would do to further this project is outlined in the *Future Work* section. The result of our work is a robot with a shopping cart that can follow a human based on targets, turn when the human turns out of the aisle and prevent itself from hitting obstacles. These functionalities are displayed in this link: https://www.youtube.com/watch?v=NZGGJV0h-k0

Future Work

In terms of improvements, for the image processing aspect, we would look more into investing in a more powerful processor and more robust algorithm such as one that involves neural networks to be able to more accurately identify the target human and obstacles. For robotics, to scale this idea up, a faster robot, with four wheels and a greater load limit would be used such that we would be able to mount an actual shopping cart on it. Additionally the robot would ideally, be able to turn and more forward at the same time, rather than just pivot turn. Having this kind of turning, along with four wheels, would greatly minimize the jitter. For the path planning portion, we would improve our design by being able to better map our environment by having sensors on multiple heights of the cart and adding image processing to aid in the detection and path planning aspects. Overall, we would want to move towards not having to have as many colored targets on the human to prevent false positives, and further explore our algorithm and implementation as to what to do when the human is lost. Ideally, once these aspects are addressed, we would like to have a larger and stronger robot platform to support a larger size shopping cart. This would also help with the robot stability, and prevent the swinging of the camera when there is a high load, as well as increase the load limit itself.

10. References

Ackerman, Evan. “Walmart and Five Elements Robotics Working on Robotic Shopping Cart.” IEEE Spectrum: Technology, Engineering, and Science News, IEEE Spectrum, 28 June 2016, spectrum.ieee.org/automaton/robotics/industrial-robots/walmart-and-five-elements-robotics-working-on-robotic-shopping-cart.

Anna. “E-Mart Introduces Eli Autonomous Shopping Cart.” Robotics & Automation News, 4 May 2018, roboticsandautomationnews.com/2018/05/04/e-mart-introduces-eli-autonomous-shopping-cart/17113/.