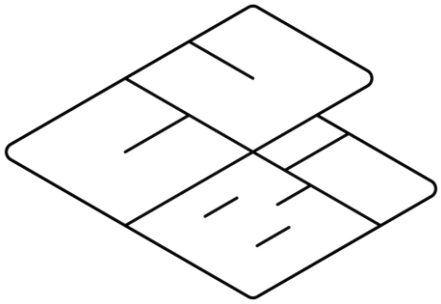

CuBi: Room Decluttering Robot

MRSD Team D

The Team:

Laavanye Bahl

Paulo Camasmie

Jorge Anton Garcia

Changsheng Shen (Bobby)

Nithin Subbiah Meganathan

Mentor:

Dr. David Held

Customer:

Cyert Center for Early Education

Carnegie Mellon University

May 2019

Abstract

Despite robotics making advancements in various fields, except for iRobot's Roomba, no other robot has survived the home setting. It is because Roomba is the only commercial robot that addresses a utility need. Our goal is to create a utility robot that declutters the floor. Families with both parents working find it laborious to maintain a clean home with their kids cluttering the floor with toys. For our customer, at Cyert daycare, teachers find it hard to clean the clutter while taking care of the babies. To tackle this problem, we are developing CuBi - an autonomous robot that seamlessly declutters the floor off toys.

At this phase of the project CuBi can perceive toys on the floor based on size, move towards it, pick it up, and drop it in the start position. It can perform this operation continuously until all the objects within a certain area are cleared of toys.

Table of Contents

1. Project Description

There is a need for a task-specific domain robot, that will help people with daily chores, such as the "Roomba" [1], a robotic vacuum cleaner. However, to our knowledge, there is no consumer-level robot that will declutter the room, a task which is needed prior to a vacuum operation. A mobile base with a robotic arm on it is not a novel concept. There are examples from toys to research-grade robots. However, by optimizing the design of a robot, its mechanisms, perception and algorithms to a specific task, it should be possible to develop an efficient, affordable, and commercial version of it [2][3]:

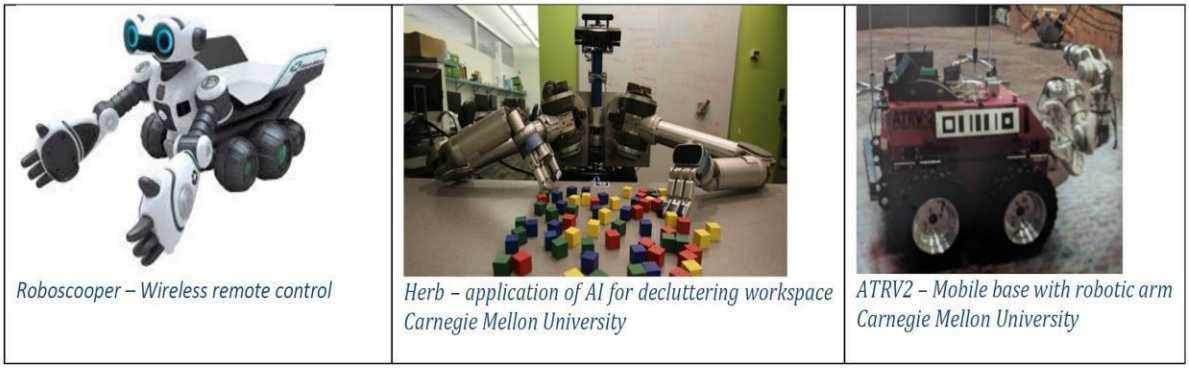

Fig 1. Different applications of robotic manipulator

The main goal of this project is to automate the task of picking up clutter to improve the daily lives of parents, pet owners, and daycare workers. By the end of the project, our robot should be able to encounter a room in an initial state, such as the one on the left, and work autonomously, avoiding people, pets and obstacles along the way, to achieve the state on the right, in an optimized matter, with all objects picked up from the floor and placed at a desired destination.

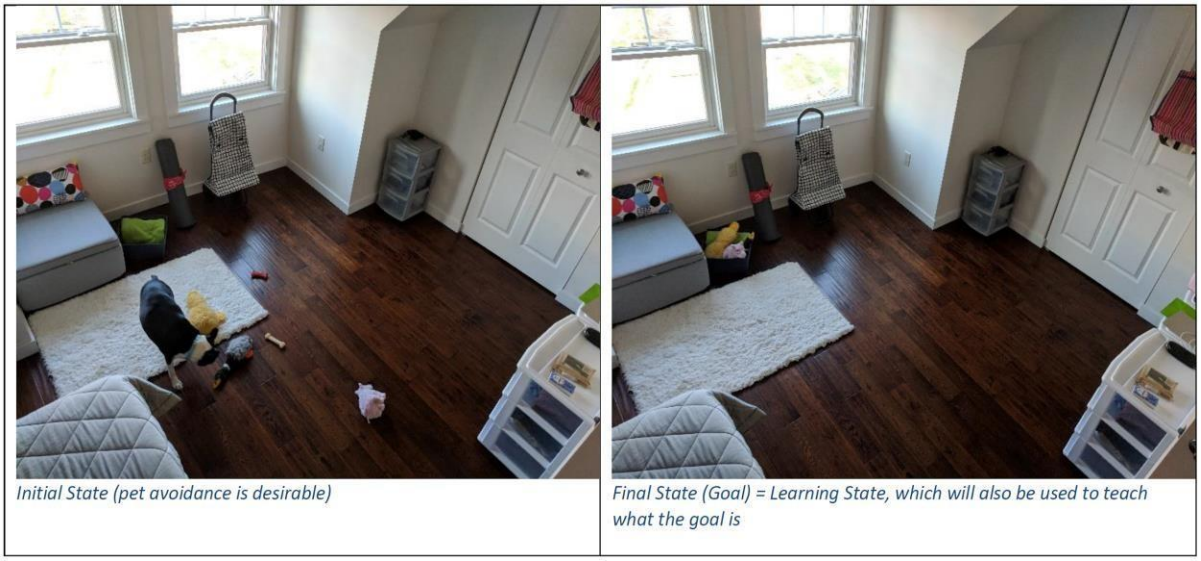

Fig 2. State of the room before and after CuBi's operation

2. Use Case

Zachary is an early childhood educator in an infant toddler classroom. Most of the children are under one year of age. Ellen, who is under his care, suddenly started crying after dropping a rattle from her hand. As Zachary comforted her, he looked around the room. Over the last hour, the room had become cluttered with a dozen toys. As the children explored the space, the educators were busy with changing diapers, giving bottles, and offering children snacks. Now that all of that is finished, it was time to get the room cleaned up and organized so the morning activities can begin.

Fig 3. CuBi picking clutter off the floor

Fig 4. CuBi dropping the clutter at designated location

Once Ellen stopped crying and was playing peek-a-boo with an educator in the other playroom, Zachary turned on CuBi to perform its work while he gathered the materials needed for the light and shadow exploration. CuBi went around the room, when no babies were around, picking up tennis ball-sized toys from the floor and placed them in the bin at the corner of the room. For 30 minutes, CuBi performed its work without crashing into anything and placed almost all the toys in the bin. Then CuBi went back to its dock to self-recharge for a few hours. Now that the floor was decluttered, Zachary was ready to set up for the light and shadow activity. He was happy to see the clean floor and quickly set up the next experience.

Fig 5. The clean room after CuBi's operation

3. System-level Requirements

The system-level requirements are divided into two categories: mandatory requirements and desirable requirements. Under each category, requirements are further classified as performance requirements that are functional requirements with qualitative measures, and non-functional requirements, based on their essence. The requirements originate from the project goal, derived from the use case, and validated through preliminary calculations and stakeholders' feedback.

There have been no changes in the mandatory requirements since the PDR since we believe we are on track to achieve all the requirements. We might modify M.P.9. depending upon future design iterations. In the desirable performance, the auto-charge requirement has been removed since our base does not have that capability unlike iRobot's Create 2, the robot base we started with.

3.1. Mandatory Performance Requirements

The system will:

M.P.1. Explore, scan and create a 2D map for 90% of the reachable area in a room

M.P.2. Clean up a 20m² room with a dozen tennis-ball-sized objects within 30 minutes.

M.P.3. Navigate to a designated reachable location in a room with pose error < 10%.

M.P.4. Go over carpets and rugs with thickness less than 12mm.

M.P.5. Detect and avoid 95% of the obstacles with a clearing distance of 20cm.

M.P.6. Classify all tennis ball-sized objects with classification error < 20%.

M.P.7. Pick up and collect each classified object within 5 attempts.

M.P.8. Pick up at least 80% of the classified objects in the room

M.P.9. Carry at least 2 tennis ball-sized object to the drop-off location.

M.P.10. Drop the clutter in a designated container marked with AprilTag with success rate > 90%

3.2. Mandatory Non-Functional Requirements

The system shall:

M.N.1. Operates autonomously.

M.N.2. Be mechanically safe (i.e. no sharp edges).

3.3. Desirable Performance Requirements

The system will:

D.P.1. Continuously operate for at least 2 hours once fully charged.

D.P.2. Have a sensing range of 15 cm to 4 m.

D.P.3. Have a physical dimension limit of 0.5 x 0.5 x 0.5 m.

D.P.4. Be affordable with a maximum cost of $5000 USD.

3.4. Desirable Non-Functional Requirements

The system shall:

D.N.1. Be easy to use by pressing buttons or through a GUI.

D.N.2. Have an inconspicuous, seamless appearance.

D.N.3. Be reliable and not get stuck or malfunction frequently.

4. Functional Architecture

The functional architecture aligns heavily with our use case. After CuBi is turned on, it is constantly looping through four major functions: exploring, categorizing objects, picking up objects, and dropping them off at a predefined location. It will continue repeating these major tasks until the room has successfully been decluttered. At this point, it will return to its initial location and start re-charging.

One thing to note about the architecture is that the boxes marked with a red arrow require localizing, trajectory planning and moving. The boxes shaped like diamonds show functions which output decisions that determine what the robot will do next.

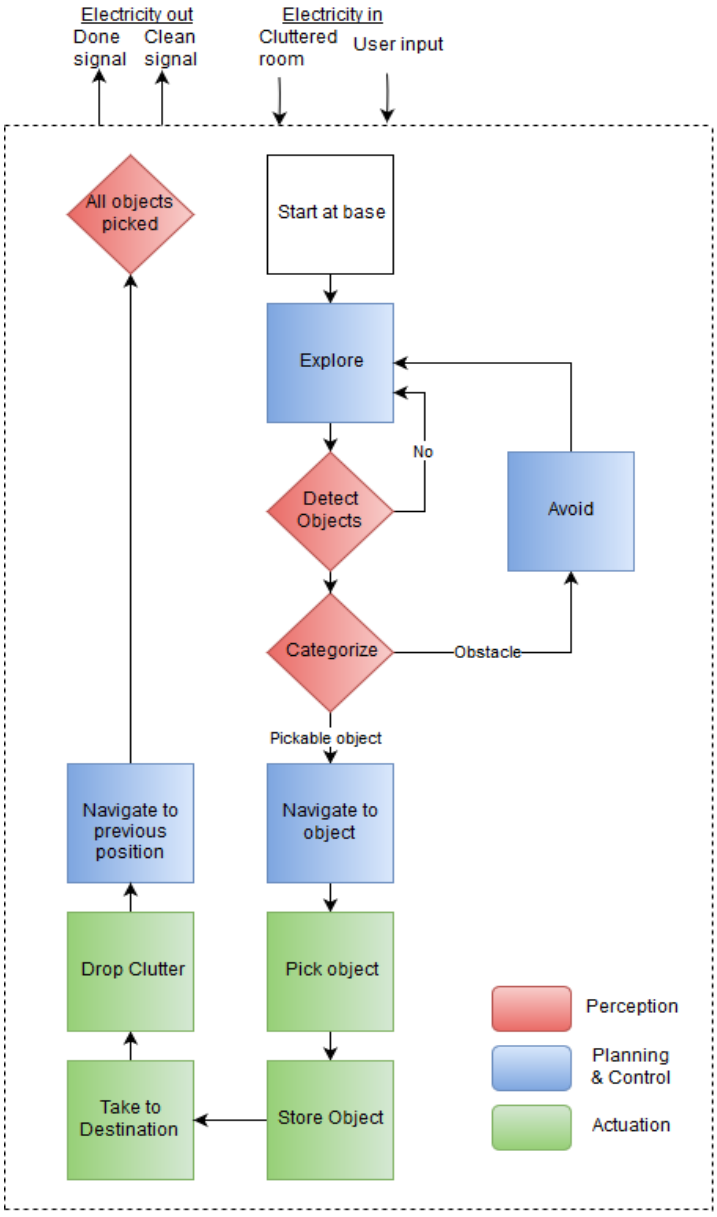

Fig 6. Functional Architecture

5. Cyberphysical Architecture

The Cyber physical architecture shows the interactions between the hardware and software components of the system. It is divided according to the basic functionalities of a robot: Sensing, Perception, Planning, and Actuation.

Cyber physical connects each function in our architecture to a physical and information aspect of our system. For example, the 'Avoid Obstacle' function requires CuBi to sense, identify, and plan in order to accomplish it. Thus, our architecture has been designed in such a way that the subfunctions of each function relate to individual components of our system.

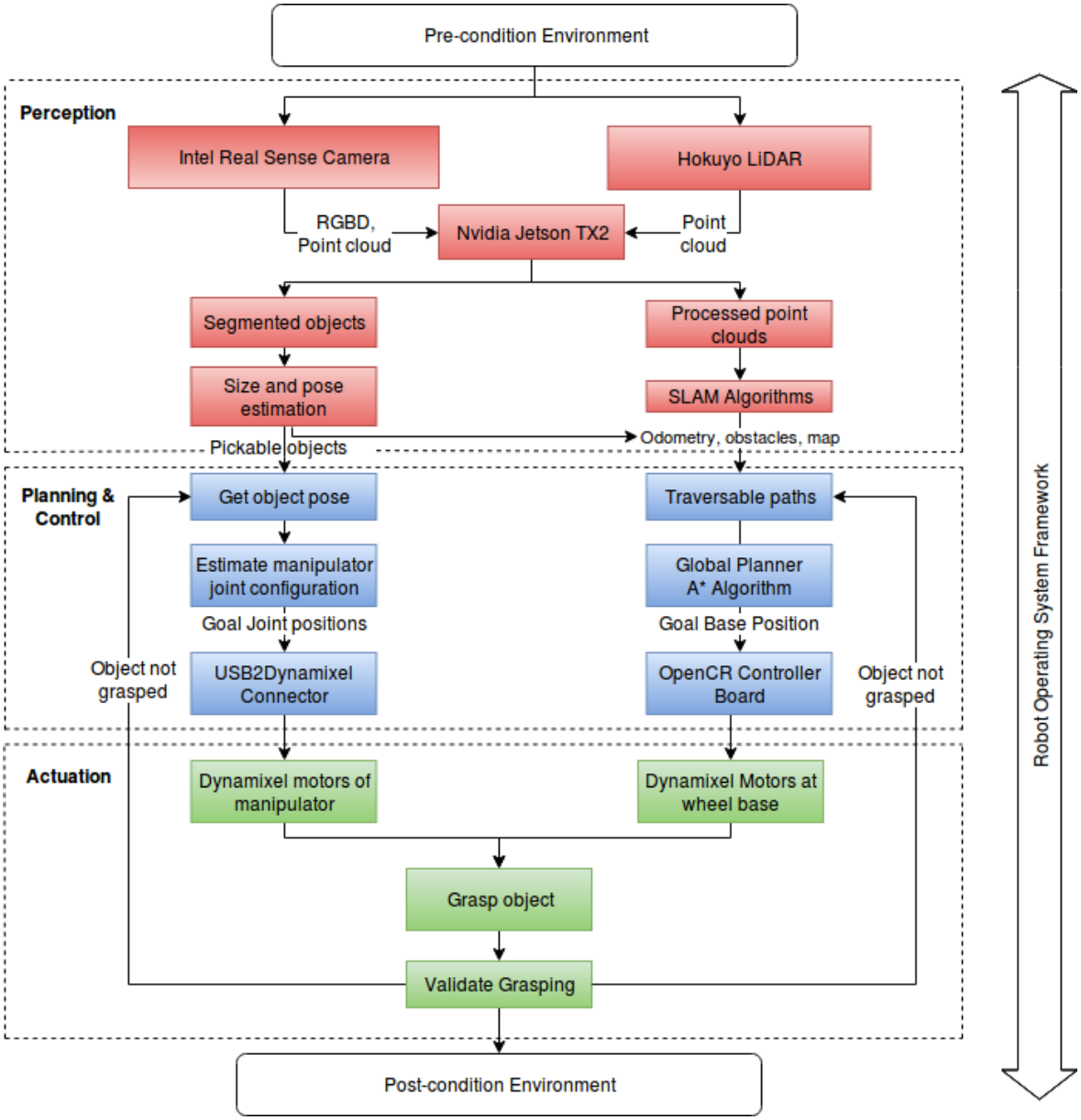

Fig 7. Cyberphysical architecture

6. Current System Status

6.1 Targeted Requirements

The following set of requirements were targeted and achieved during our Spring Validation Demonstration.

M.P.2. Clean up a ~4m² area with 5 tennis-ball-sized objects within 10 minutes
M.P.3. Navigate to a designated reachable location in a room with pose error < 10%
M.P.6. Classify all tennis ball-sized objects with classification error < 20%
M.P.7. Pick up and collect each classified object within 5 attempts
M.P.8. Pick up at least 80% of the classified objects in the area
M.P.10. Drop the clutter in a designated container at a predetermined location with success rate > 90%

For M.P.2. the fall requirement is to clean up a 20m² room with a dozen ball-sized objects within 30 minutes and we achieved a toned-down version of it. Also, for M.P.10 we were able to go to the container using only odometry with reasonable position accuracy. We will utilize AprilTags in the future to reset the odometry.

Half of the requirements were achieved using the integrated system which validates the robustness of each subsystem. Individual subsystems that helped achieve the requirements are perception, manipulation, planning and control.

6.2 System Description

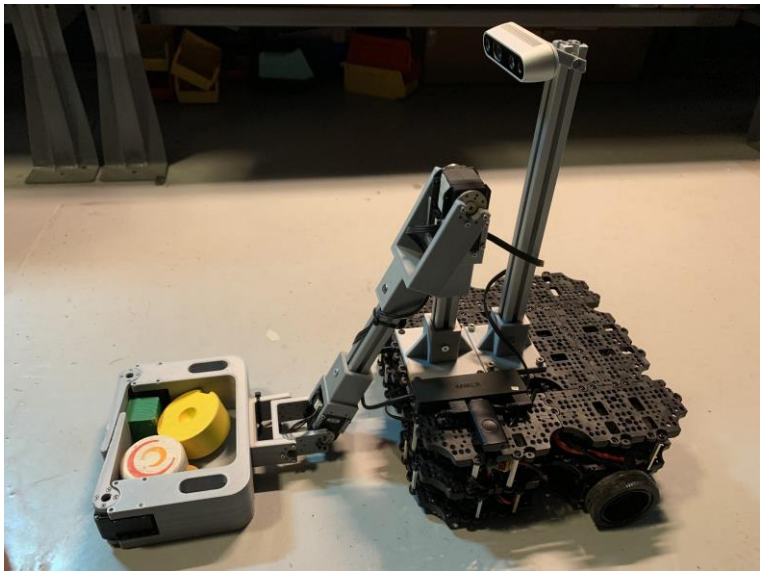

Fig 8. Fully integrated CuBi

Figure 8 shows the completed assembly of CuBi that we used for our demonstration. The robot has the Turtlebot 3 Waffle Pi as its mobile base. The base has manipulator fitted on the top layer and has Intel RealSense and Hokuyo lidar mounted on it. It has a PCB for power distribution and an OpenCR controller that is used for the mobility of robot. The system is complete and functional, and we will improve the design for robustness and performance in the coming semester.

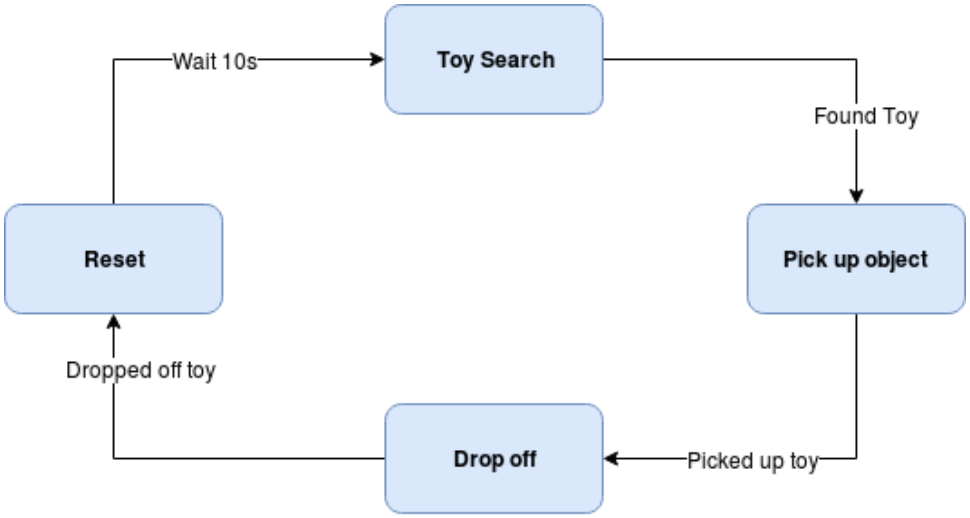

Fig 9. State machine

The state machine that is depicted above is used by CuBi to perform its entire operation. When CuBi is turned on it goes on to ‘search mode’ when it performs a 90 degree sweep in either

direction till it detects an object. Once an object is detected, CuBi switches to ‘pick-up object mode’ when it moves to the goal pose, picks up the object. Then in the ‘drop-off mode’ it goes to the home position to drop the picked object on the tray. Then it resets to the home position and starts searching for an object.

6.2.1 Mobile Base

We have performed trade studies on various available off-the-shelf mobile robot platforms, as well as the option to build our own mobile base from scratch. Then we chose the TurtleBot 3 Waffle Pi as our mobile base platform. The main reason to select this platform is the high-modularity design and capability to be easily modified and integrated with other components.

During the past semester, we have finished the assembly of the entire robot. The base is powered by two Dynamixel XM430 motors with a differential drive steering mechanism with two caster wheels on the back. Dual large capacity Li-Po batteries are mounted on the base to power the system, ensuring at least 1 hour running time once fully charged. In the middle layer, an OpenCR microcontroller board is installed to control the chassis motors. It has an embedded IMU onboard, and continuously sends fused IMU and wheel odometry information to the onboard computer through a serial port. On the top layer, we have mounted the manipulator, the NVIDIA Jetson TX2 onboard computer, and the power distribution PCB that we designed.

Currently, both software and hardware components of all other subsystems have been fully integrated together with the mobile base. The base can be controlled either by using a remote joystick, or by receiving velocity commands sent from other nodes through ROS.

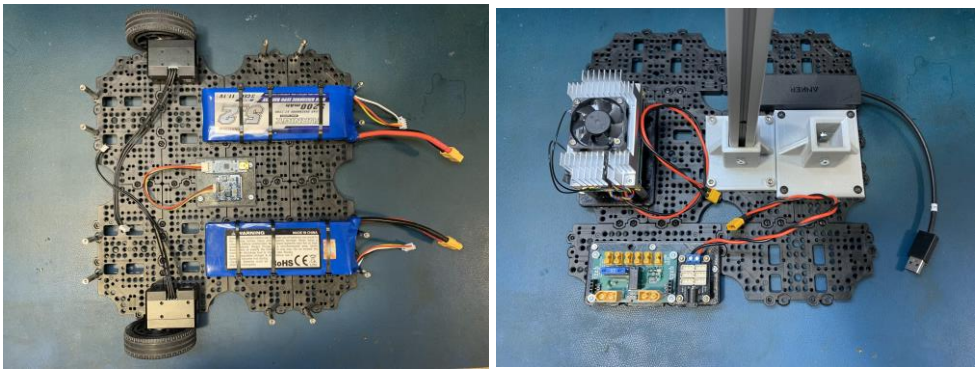

Fig 10. Top and bottom layer of the base

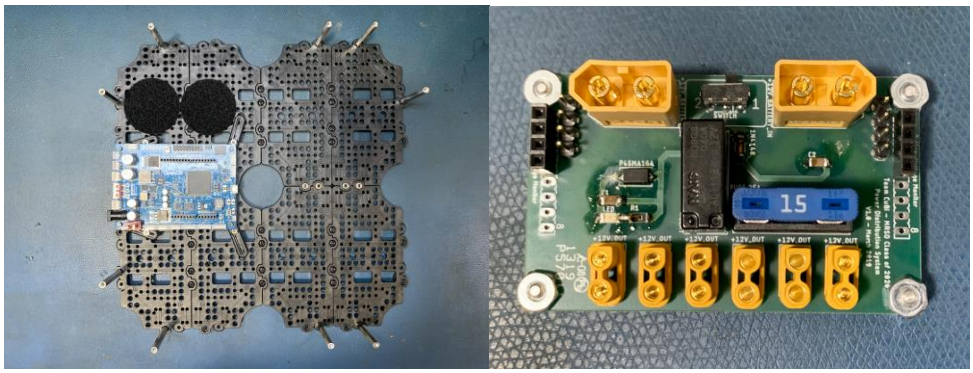

Fig 21. Middle base layer and PCB used in CuBi

6.2.2 Manipulation

Manipulation subsystem is a crucial aspect of our project since our goal is to create a task-specific robot. This led us to design and fabricate a manipulator arm from scratch rather than using an off-the-shelf manipulator. We decided to use caging mechanism for gripping as it generalizes grasping for a wide range of objects. Caging is a grasping strategy where an object's mobility is restricted using the end effector. The below figure depicts the design of the manipulator.

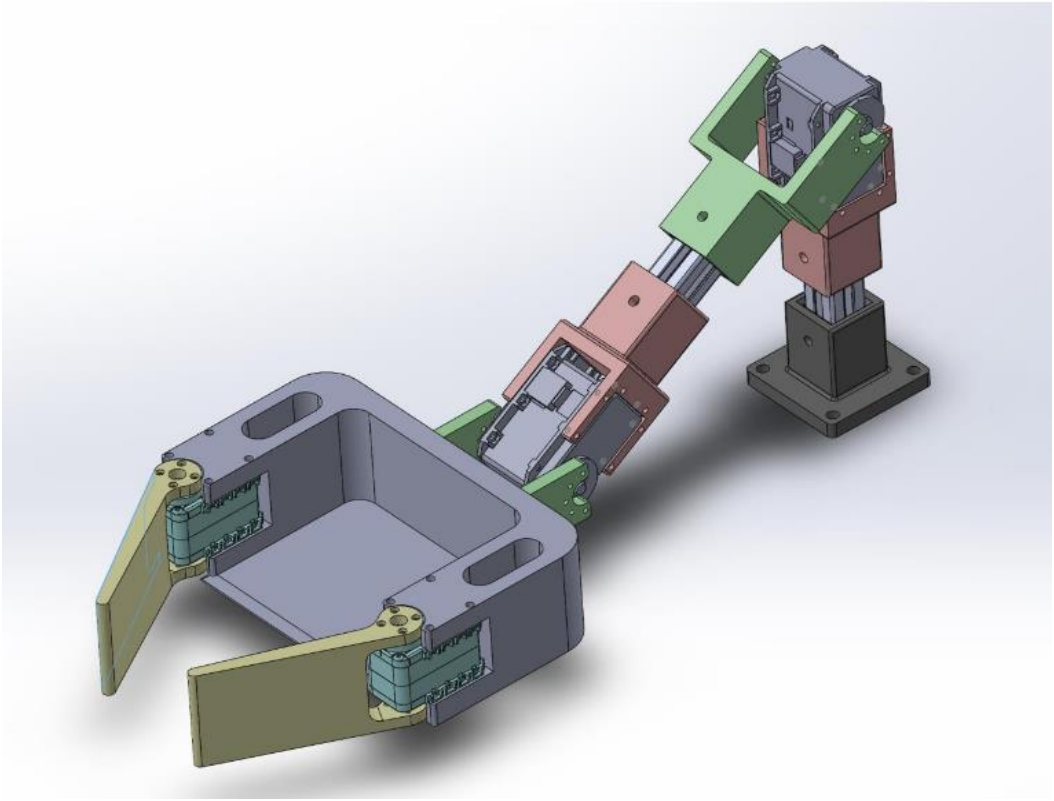

Fig 11. Manipulator design

The manipulator is actuated by a pair of Dynamixel AX-12A motors for the gripper and a pair of Dynamixel MX-106 motors for the arm, making it a 4-DOF manipulation system. Each paddle is half the size of the tray which minimizes the possibility of the object getting stuck. The links are made up of 80/20 extrusion aluminum bars, and the brackets and tray are 3D-printed. We calculated a maximum payload of 500g for this manipulator.

6.2.3 Perception

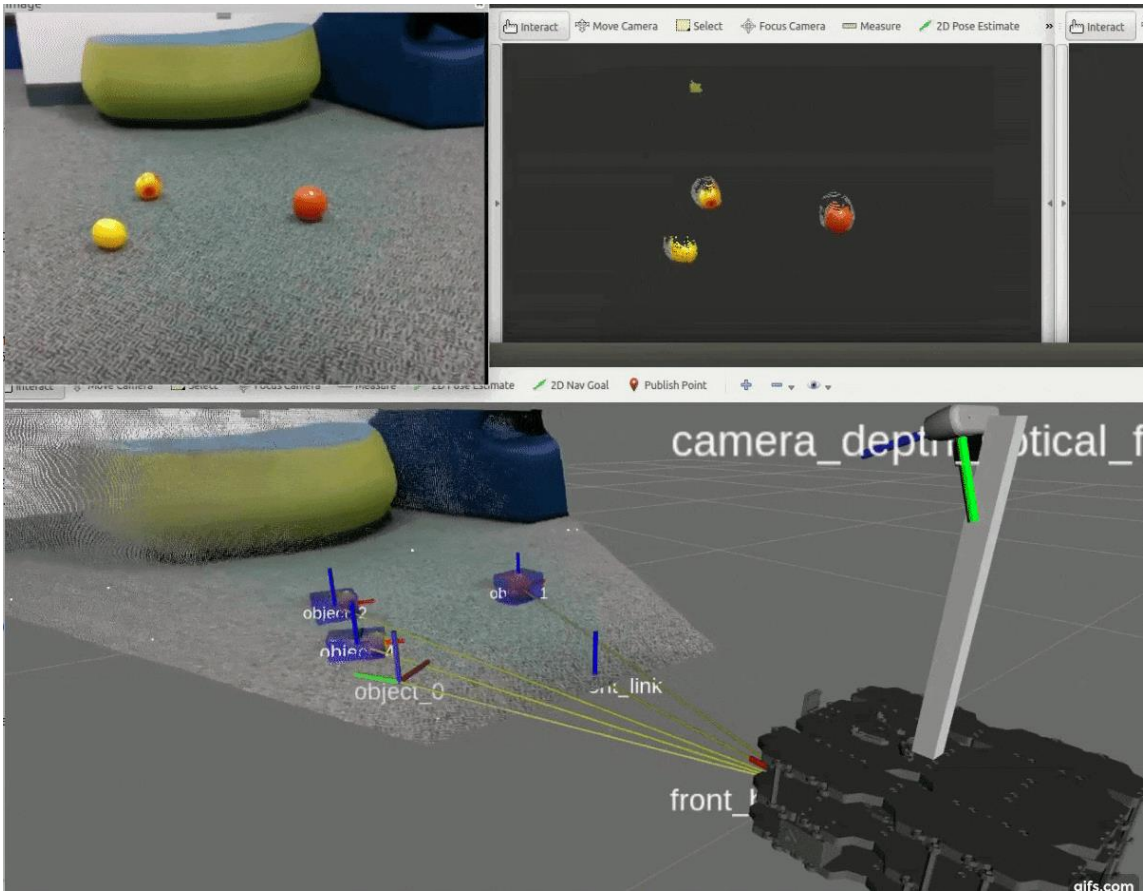

Fig 12. Perception visualization

Perception is used to sense the environment and to identify different objects. A stereo RGBD camera is used for obtaining rich visual RGB data for classification, as well as depth and point cloud for size and distance estimation. Combined information is used to estimate the size, pose and type of object. A combination of techniques such as geometric vision, learning-based and probabilistic methods are used to classify objects into two categories: objects to pick and obstacles to avoid. Figure 12 shows how are perception pipeline works in real-time. Currently, classical vision techniques and PCL library is used to process the point cloud. A region of interest is cropped, noisy points are removed, RANSAC plane segmentation is performed, followed by clustering of remaining objects on ground. A 3D bounding box is fitted around each detected object

and the pose is estimated to classify them according to threshold criteria of position and size. Finally, object tracking is applied to uniquely identify these objects over time. To complement geometric features, learning based methods with additional labeled or synthesized data will be used. This will also aid in obstacle detection. A list of detected objects with their absolute positions in the map (calculated with the help of relative positions with respect to the robot) will later be maintained and utilized by the planner for navigation.

6.2.4 Planning & Control

Given a goal position, CuBi's motion is defined as follows:

- Orient such that CuBi faces the target point
- Move in the x-direction to reach the target point
- Rotate till CuBi reaches the target orientation

Planning and control subsystem roughly consists of three layers. The outermost layer interacts with the vision subsystem and receives the object pose. The middle layer consists of the state machine as depicted in figure 9. Once the middle layer receives the object position, using the odometry data to get CuBi's position, it computes the relative error between the current position and the goal. This error is sent to the last layer which uses a PID controller to publish velocities to the base.

In addition to the control of the base, the state machine also interacts with a node that controls the Dynamixel motors of the manipulator. The manipulator has a fixed set of configurations - moving, pick-up, and drop-off. The entire pipeline is integrated using Robot Operating System (ROS). The availability of ROS packages for Dynamixel motors and the Turtlebot base proved very beneficial to our cause.

It is to note that the maximum range till which CuBi operates is limited by the range of the vision system as CuBi does not perform any sort of exploration or global planning as of now.

6.3 Modeling, Analysis, Testing

As a proper system engineering design process, we have been doing a lot of modeling, analysis and testing while designing and implementing our system.

Specifically, for the perception subsystem, we modeled the objects to pick up based on size, which is specified in our system functional requirements. We then implemented and fine-tuned our computer vision algorithm to classify the objects accordingly, with outlier rejection.

For the mobile base subsystem, first we validated our maximum traction that the chassis motors and wheels can provide by testing the robot with payload on different ground surfaces, such as smooth indoor floor surface and carpets. We then adjusted the weight distribution of the system, in order to avoid slipping on smoother surfaces.

We also evaluate the accuracy of wheel odometry, by comparing the reported value to the actual distance measured by hand that the robot travelled, to make sure that the amount of odometry drift was within the range specified in our performance requirements.

For the grasping subsystem, we designed, tested and iterated multiple generations of the paddle design of various length, roughness and shape. Half size of the tray as the paddle length was proven to be the most robust for grasping. Moreover, we estimated the maximum payload required for each joint of the manipulator, and chose motors based on the calculated torque required.

For the electronics system, we calculated the power consumption of all the onboard components. Given the 1 hour running time as one of the performance requirements of our system, we then calculated the battery capacity needed and installed dual batteries based on that.

6.4 SVD Performance Evaluation

The high-level goal for our Spring Validation Demonstration (SVD) was that CuBi had to find toys randomly placed within a radius of 1.2m and drop them off in a box that was right behind it. This required CuBi to be a fully integrated system.

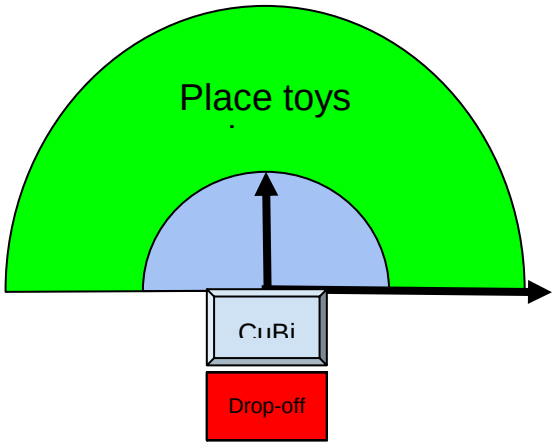

Fig 13. Set-up of the SVD.

Our test set included several plastic toys in the shape of fruits. The fruits varied in size ranging from a strawberry fruit to a pear. They were plastic and very light weight. They would also roll if they were pushed.

Fig 14. Examples of toys in our test set.

Our metric for success was counting the number of toys in the box at the end of 25 minutes. If there were three or more toys, then the SVD would be considered a success. We were able to pick up all five toys in a bit over 10 minutes. In doing this, we also achieved our secondary goals listed below:

a. Detect a toy and calculate its pose relative to CuBi
b. Approach a toy within 2 cm
c. Pick up one toy on the tray
d. Lift the tray (with a toy inside) up from the ground.
e. Return to the start position
f. Drop a toy on the ground

During our Spring Validation Experiment, we were able to perform the test several times as we were able to pick up 5 objects in about 10 minutes. During the first test, our team would place the toys and during the second test, we asked the evaluators to try to break our system.

Our system would always detect the toys. However, 5% of the times we would miss them when approaching them. This was because we currently only use the one frame to determine the object's pose and there may be some jitter. In the future, we will use the information about the object's detected pose over several frames. In terms of picking up objects, we were able to pick up our test set objects over 95% of the time. Our system would only fail if the object got stuck on the lip of the tray. This would cause on overload in the Dynamixel motors. Nevertheless, most of the times CuBi would still be able to carry the object even though it was never actually lifted on the tray. Overall our system was robust and since the encore, every time we tested it, we passed our targeted requirements.

We also had time to test objects outside of our scope like phones, computer mouse, white board markers among other items. We successfully picked up two white board markers which were placed right by each other. We were able to pick up the mouse when approaching it from specific directions, otherwise it would fail. We also had difficulty detecting phones potentially due to the reflection on the screen.

6.5 Strong/Weak Points

Strong points:

One of the aspects which makes the system robust is our accurate object detection and pose estimation algorithm. It is very good at filtering out what is not clutter and the number of false positives is close to zero. We tested at Cyert at the end of the semester and it was able to detect even toys which had holes within them like the one in figure 15.

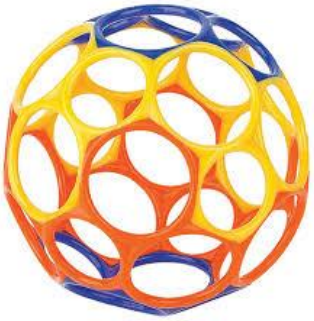

Fig 15. Unseen toy tested at Cyert center

In addition, CuBi's caging strategy is very good at generalizing the objects we can pick up from the ground. Even though we had never tested with picking up small objects like markers, it was able to do so.

Finally, CuBi has very clean assembly and wiring. This makes it very easy for us to charge the batteries or swap any necessary components. We have spent very little time debugging this aspect thanks to having all the components clearly labeled and visible.

Need to be improved / implemented:

One of the biggest aspects that needs to be improved is CuBi's odometry drifts. After picking up 5 toys, CuBi is off by around 30º and 15cm. With this odometry drift, we will be unable to drop off 12 toys in the box and hence fail our FVD. Currently we are just using wheel encoders and the IMU, but in the future we are looking to fuse it with visual odometry. We also want to reset odometry whenever we reach the drop off location.

Furthermore, CuBi is unable to pick up toys which are flat on the side that is touching the ground. The reason for this is they get stuck when trying lift the onto the tray. CuBi's able to pick them up even though the toy is not on the tray, but the motors get overloaded. We can either use this as a grasping strategy by preventing the motors from overloading. Otherwise, we have started testing different finger designs to try and life the toys onto the tray and we are considering decreasing the thickness of the bottom part of the tray.

7. Project Management

7.1 Work Breakdown Structure

[✔] Completed [□] Currently working [⊙] Next semester

1. **Visual sensing and perception**
   - 1.1. [✔] Sensor identification, calibration, setup and installation
   - 1.2. [✔] Object detection algorithm
     - 1.2.1. [✔] ROI identification and segmentation
     - 1.2.2. [✔] Size and pose estimation
     - 1.2.3. [✔] Object tracking
     - 1.2.4. [✔] False positive removal
     - 1.2.5. [✔] Geometry based classification
   - 1.3. [✔] Destination/ base identification algorithm
   - 1.4. [✔] Validation on toys
   - 1.5. [⊙] Obstacle detection algorithm
   - 1.6. [⊙] Data collection, labelling and fusion for learning based techniques

2. **SLAM**
   - 2.1. [✔] Sensor(s) selection
   - 2.2. [✔] Initial 2D LiDAR SLAM
   - 2.3. [⊙] Multi-sensor SLAM
   - 2.4. [⊙] Implementation and integration

3. **Planning and navigation**
   - 3.1. [⊙] Planning and navigation algorithm
     - 3.1.1. [⊙] Global path planning
     - 3.1.2. [⊙] Trajectory generation
     - 3.1.3. [✔] Local planner
     - 3.1.4. [⊙] Dynamic obstacle avoidance
   - 3.2. [⊙] Path optimization algorithm (research)
   - 3.3. [⊙] Learning based planning algorithms (research)

4. **Grasping**
   - 4.1. [✔] Manipulator
     - 4.1.1. [✔] Grippers R&D: Trade studies, prototype drawing, and fabrication
     - 4.1.2. [✔] Actuation mechanism (arms and fingers)

4.1.2.1. [✔] Selection and set up of motors and controller

4.1.2.2. [✔] Validation of assembly on actual toys

4.2. [□] Control algorithms for manipulator

4.2.1. [✔] Control program for motors

4.2.2. [✔] Target position identification

4.2.3. [✔] Grasping strategy and trajectory planning

4.2.4. [⊙] Validation with feedback

4.2.5. [✔] Dropping

4.3. [⊙] Storage and body

4.3.1. [⊙] R&D: Trade studies, prototype and fabrication

4.3.2. [⊙] Control program for motors

4.4. [⊙] Learning for manipulation (research)

5. Mobility

5.1. [✔] Mobile base trade studies (E.g. payload, torque, turning radius)

5.2. [✔] Modification of motors and actuators

5.3. [✔] Microcontroller selection and programming

5.4. [□] Odometry

5.4.1. [✔] Wheel odometry

5.4.2. [⊙] Multi sensor odometry and fusion

5.5. [✔] Controller for mobility: algorithm and feedback mechanism

5.6. [⊙] Improve mobility mechanism (E.g. Mecanum wheels)

6. Robot integration

6.1. [✔] Electric and mechanical components integration

6.1.1. [✔] Attachment and assembly of sensors, motors and actuators, battery, manipulator, jetson and microcontrollers

6.1.2. [✔] Power management, electric circuit design and assembly

6.1.2.1. [✔] Trade studies for suitable power source

6.1.2.2. [✔] Calculate power consumptions for each component

6.1.2.3. [✔] PCB design

6.1.2.4. [✔] Wiring of all components

6.2. [□] Software Integration

6.2.1. [✔] Sub-Systems ROS implementation

6.2.2. [✔] Robot urdf files

6.2.3. [✔] System ROS Integration

6.2.4. [✔] Communication protocols

6.2.5. [⊙] Parallel processing / CUDA coding

7. Validation, testing and benchmarking

7.1. [✔] Subsystem level

7.2. [✔] System-level

7.3. [✔] Battery consumption and duration

7.4. [ ] Non-functional / desired (e.g. mechanical and electrical safety, speed, noise)

7.5. [ ] Failure recovery

7.6. [⊙] Improvements (E.g. pick challenging objects, grasping strategies)

8. Project management

8.1. [ ] Team and work management

8.2. [ ] Schedule management

8.3. [ ] Cost management

8.4. [ ] Risk management

8.5. [ ] Resource management

7.2 Schedule

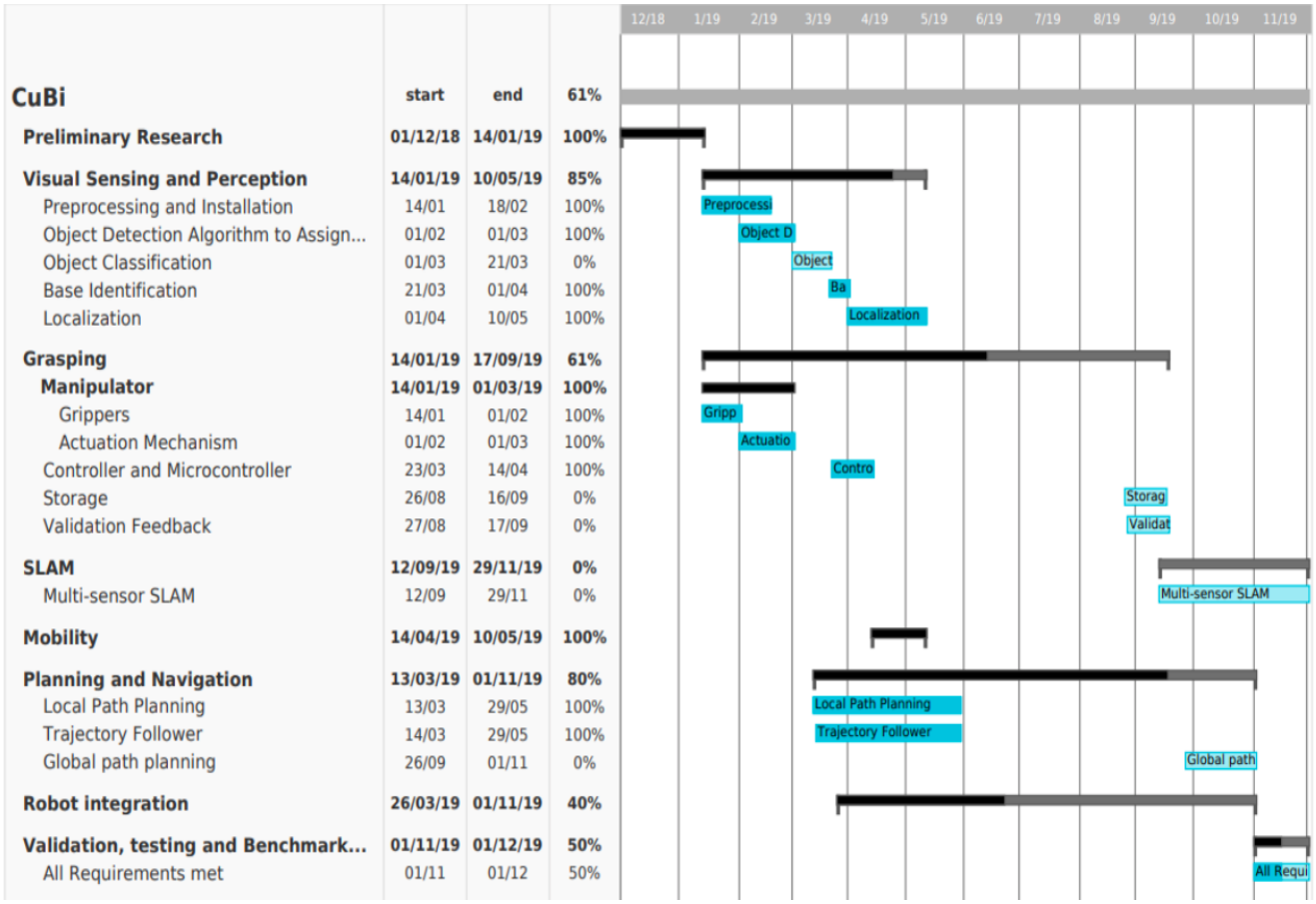

Fig 16. Gantt chart

Biggest Milestones:

Oct. 1: Can carry several toys at once.
Nov. 1: Explore, scan and create a 2D map for 90% of the reachable area in a room
Nov. 15: Can successfully perform FVD with 90% accuracy

After Nov.15, our focus will be to just make the system more robust. We will not create any new features in this time.

Summary:

We are ahead of schedule as we did not expect to have a fully integrated system which could autonomously perform a scoped down version of our FVD. This was achieved thanks to making the SVD more like the FVD. This increased our requirements for this semester, but this gives us more flexibility for next semester.

7.3 Test Plan

Table 1. Progress review goals

PR No.	Test Description	Metric
7	Place five toys on the ground half a meter apart and half a meter in front of CuBi. Turn CuBi on. Have CuBi pickup all toys and reset odometry after every pickup using the position of the AR tag on the box. Subsystems: Perception (add visual odometry), Planning (reset odometry)	Odometry drift less than 10 cm.
8	Place two toys on the ground half a meter apart and half a meter in front of CuBi. Turn CuBi on. Have CuBi pickup both toys bring them to a desired location. Repeat three times. Subsystems: Storage Mechanism (where to place toys), Gripper Mechanics (how to place toy in storage)	CuBi can carry several toys at once.
9	Place two flat toys on the ground half a meter apart and half a meter in front of CuBi. Turn CuBi on. Have CuBi pickup both toys bring them to a desired location. Repeat three times. Subsystems: Gripper Actuation (not allow for overload of motors), Gripper Mechanics (change design)	CuBi can pick up flat toys.
10	Have CuBi follow 10 waypoints and place obstacles (chairs, large toys, feet, etc…) between all of them. Subsystems: Local Planner (obstacle avoidance)	Does not crash into any obstacle
11	Manually create a map of the reachable space of a room. Allow CuBi to create a map of the room for 15 minutes and compare both manual and automatically created maps. Subsystems: Global Planner, Perception (SLAM)	Intersection area between CuBi's reachable map and ground truth is greater than 90%.

12	Independent party will place 12 toys at predefined reachable areas and will turn CuBi on. All the toys which end up in the designated location will be considered picked up Subsystems: All	Number of toys at the desired location at the end of 30 minutes.

Fall Validation Demonstration

Objective:

Validate all subsystems of CuBi and their integration and to meet all the performance requirements.

Location:

Indoor open area near the RI commons on the 4th floor, Newell Simon Hall. It will have obstacles like chairs placed in the area. Area will be closed with walls created by furniture**.**

Equipment:

15 tennis ball-sized toys, CuBi, any necessary replacements for any major subsystems which are at high risk of breaking, a box, and an AprilTag.

Setup:

1. Third party places 15 toys in the designated area and can place obstacles as desired. There are no requirements as to where these objects are placed.
2. Team CuBi will place an AprilTag at the starting location and this will designate where the toys will be placed by the end of the 30 minutes.
3. CuBi is placed in the designated starting position.

Procedure:

Table 2. FVD Procedure

Step	Description	Performance Measures
1	CuBi must be able to traverse over carpets with thickness of 12mm or lesser.	
2	CuBi starts to explore the room and perform SLAM to build a 2D map of the room.	90% of the reachable area should be mapped by the robot; validated using ground truth
3	As SLAM is performed, CuBi also uses its perception pipeline to detect objects and classify them as pickable or obstacles and records their rough positions in current map.	Classification error is less than 20%
4	After building the 2D map, CuBi applies path planning to generate a traversable path through positions of objects, obstacles and drop off location, and achieve the desired poses.	The error should be less than 10% of the desired configuration
5	CuBi avoids slow-moving and stationary obstacles.	Avoid 95% of the obstacles
7	CuBi then uses its manipulator to pick up at least two objects off the ground.	Manipulator should pick-up within 5 attempts
8	CuBi should be able to pick most of the toys off the ground.	At least 80% of the toys should be picked up by CuBi
9	CuBi should be able to drop the clutter at the designated position marked with AprilTag accurately.	The success rate of dropping should be more than 90%. Error is the difference in Euclidean distance between the AprilTag and the dropped toy's location. The threshold for counting as a failed drop is 0.5m.
10	CuBi should be able to clean up 20m² area in a reasonable time.	The time should be within 30 minutes

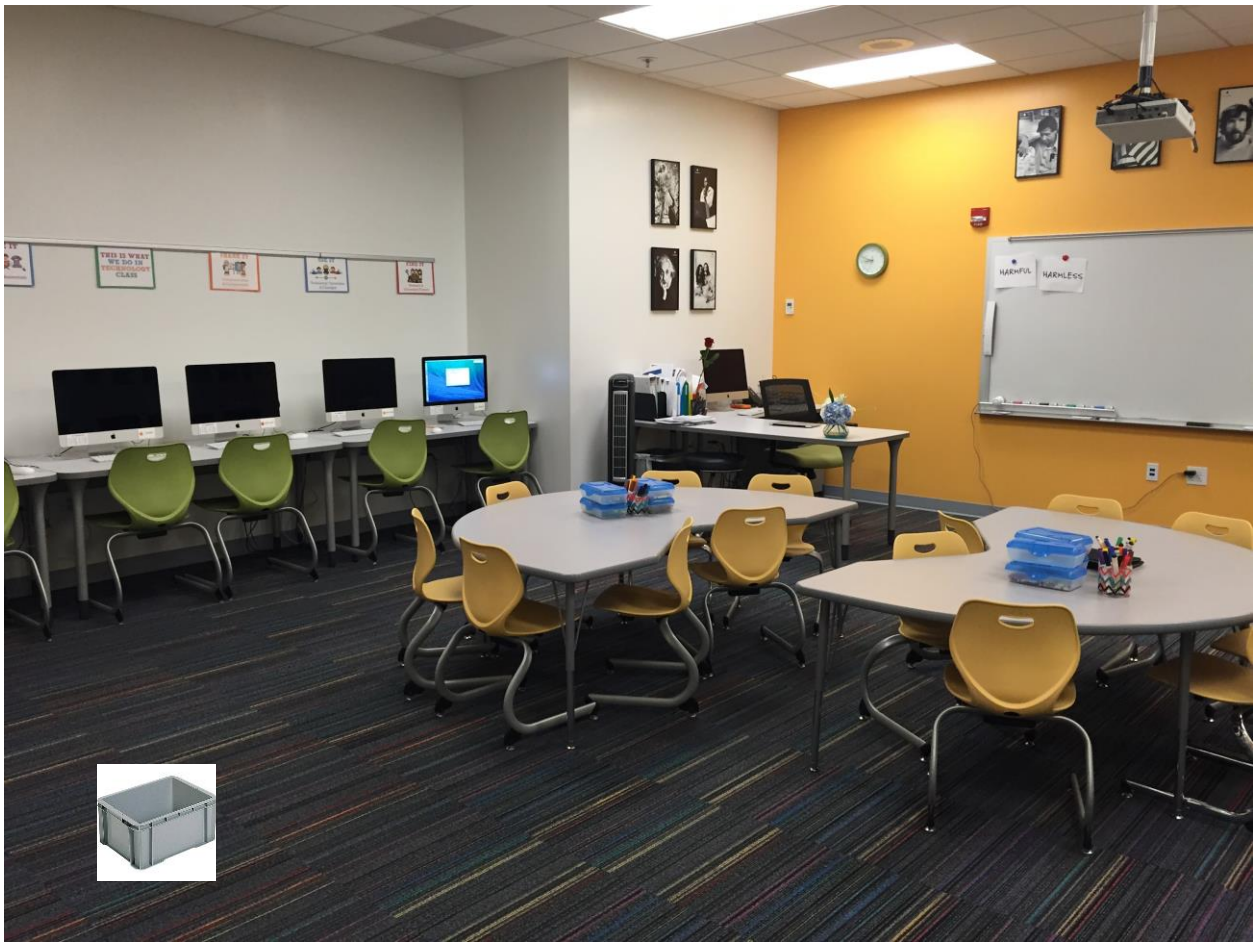

Fig 17. Example test area for FVD

7.4 Budget

Table 3. Budget

Component	Manufacturer	Part #	Unit Price (USD)	Quantity	Total Cost
Mobile Base	TurtleBot	TurtleBot 3 Waffle Pi	1399	1	1399
RGBD Camera	Intel	RealSense D435i	199	1	199
Computing Platform	Nvidia	Jetson TX2	599	1	599
Laser Scanner	Hokuyo	URG-04LX-UG01	1115	1(Inventory)	0
Servo Motor	Dynamixel	MX-106T	499	1(Inventory)	0
Servo Motor	Dynamixel	MX-64T	299	1	299
Servo Motor	Dynamixel	MX-28T	219	2	438
USB-Serial Converter	Dynamixel	U2D2	50	2	100
Connector	Dynamixel	ROBOTIS FR12- H101K Set	33.9	2	67.8
Connector	Dynamixel	ROBOTIS FR12- S101K Set	23.9	2	47.8
Battery	Turnigy	5200mAh 3S 12C LiPo	40	4	160
Misc Electronics	Multiple	Misc	200	1	200
T Slot Aluminum Extrusion	Zyltech	EXT-2020-REG-1000- 10X	7.99	10	79.9
Aluminum Profile Connector	PZRT	2020 Series	25.99	1	25.99
					3615.49

The total budget of the project is $5,000. Currently we have spent $3,615.49, which is 72.3% of the total budget. However, we have already made most of the purchases for big ticket items, including the TurtleBot 3 Waffle mobile platform, Dynamixel servo motors and the onboard computer, which comprised most of our budget. Overall, the budget management is in a good status.

7.5 Risk Management

Table 4. Risk table

Risk ID	Risk	Type	Description	L	C	Mitigation
1	Manipulator	Technical	Design of the manipulation system should be robust to pick objects from ground	5	2	Iteration and validation of manipulator design
2	Privacy issues	Technical	People might be skeptical about putting a robot with camera at homes	2	3	Blur people's faces and perform edge computation
3	Plane segmentation	Technical	Different surface textures for plane removal poses problem	2	4	Combine RGB images with stereo point clouds
4	Low-light conditions	Technical	Low-light or differing light conditions can affect vision algorithms	4	3	Sensor fusion can help mitigate
5	Dynamixel motors	Technical	Control of inventory Dynamixel motors is proving hard	3	1	Troubleshoot with help or replace with other motors
6	Auto-reset	Technical	CuBi might get stuck during operation requiring hard reset	2	3	One-push button to perform hard reset
7	Pickable objects together	Technical	Multiple objects might be together causing unintended actions	2	4	Perform interactive perception of the environment

The above table lists the important risks that we anticipate encountering in our project. Out of all the risks, we have majorly dealt with risk ID: 1,2,5. For risk ID 1 which is manipulator might not be able to scoop objects, we were able to get good results for our test set objects. However, it cannot generalize for many objects. But this is expected since grasping is still an unsolved problem. Risk ID 5 has also reduced in number since now we have the experience of troubleshooting Dynamixel motors. We also purchase additional motors as backup in case if we can't fix it. Risk ID 3 has not been updated since we did not test on many different floors. Risk ID 2 has been reduced because achieving blur of people's faces is a comfortable task for the team. Risk ID 6 and 7 have been added during after PDR.

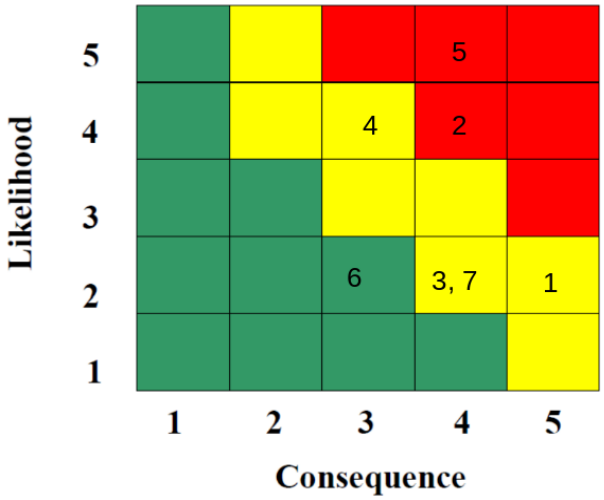

Fig 18. Risk Matrix at PDR

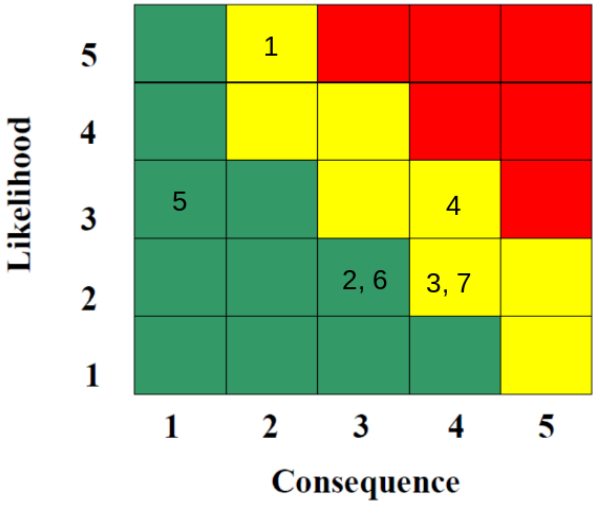

Fig 19. Risk Matrix after PDR

8. Conclusions

Key Lessons Learned:

- Working together more effectively saves tremendous amount of time for integration. Even if one person oversees a subsystem, they should make major decisions discussing with the rest of the team first. We have had several times were people have spent long hours designing a component to later realize would interfere with another subsystem. Furthermore, when integrating, it's important to have everyone present, so that questions can be solved immediately.
- Utilizing GIT branches and discuss before pushing code onto the robot. No matter how small the change, never push changes to the robot without testing. It is also important for us all to develop on separate branches, so that when debugging, we know exactly what changes were made.
- Do not over commit for each progress review. We usually send ideal case scenarios as our progress review goals, but we should be a bit more conservative.
- Reach a point where you can integrate everything as soon as possible to allow early and constant validation

Goals for Fall Semester:

- Environment exploration and mapping: currently all the toys were placed within the field of view of CuBi, so all it needed to do is sweep to find toys. Next semester we want CuBi to autonomously explore and find toys.
- Robust odometry based on multi-sensor fusion: We currently are using wheel odometry and IMU for odometry. We will incorporate visual odometry too.
- Advanced global path planning: We will create an algorithm to choose which toys to pick up and in which order.
- Faster smooth trajectory following: To get a smooth movement of CuBi, we have penalized speed. It would be interesting to see how quick we can make CuBi without having it feel intimidating.
- Dynamic collision avoidance: Perform collision avoidance in real-time.
- Manipulator design improvement: Modify the design of the manipulator to better generalize to objects. The main aspect we are trying to fix is the ability to pick-up flat objects.
- Failure detection & recovery mechanism: If CuBi fails, currently we need to reset it to start again.

9. References

[1] https://www.irobot.com/

[2] http://www.theoldrobots.com/Roboscooper.html

[3] https://www.cmu.edu/news/stories/archives/2016/may/robots-clutter.html