SortBot

Authors: Teddy Lin, Ethan Lu, and Alejandro Miranda
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable of sorting trash within trash processing facilities. Our system is a robotic gantry that moves along the x, y, and z-axis to pick up trash, using a vacuum-based end effector to pick up the trash, identify it using computer vision, and place it into its corresponding bin. Additionally, our system will provide a sorting accuracy of 90% via autonomous means.**

***Index Terms*—Gantry, YOLO, Vacuum, Web App, OBB, Computer Vision, Trash, Recycling**

1 INTRODUCTION

Current trash sorting solutions are ineffective in sorting all of different types of trash without the involvement of manual human sorting. In providing a more cost effective and efficient solution, we have designed SortBot, a trash sorting robot that aims to eliminate the need to manual human sorting. Trash processing facilities usually have many people working at a conveyor belt to sort the trash.

It is important that our project exist, since it provides a solution to the amount of waste that gets passed through waste sorting facilities every day. The need for properly sorting trash types is essential to the maintenance of our planet. There are 2.12 billion tons of waste produced each year across the world [1].

Trash processing facilities are the main users of our technology as they need this system to enhance their trash sorting capabilities. Currently, trash processing facilities employ human workers that have to stand by the conveyor belts and manually sort the trash using their hands. Our robot will operate above the conveyor belt and do the same task as the human workers do, but now autonomously.

Some competing technologies utilize robotic arms to sort the trash on conveyor belts into the appropriate bins. Despite the dexterity of a robotic arm with a gripper type end effector, our solution provides a simpler approach, which reduces the cost of the product significantly. Another competing technology is that of ZenRobotics, a company that employs the use of a gripper type end effector on a gantry type system like that of ours. It also suffers from the complexity in that our suction/vacuum based approach to picking up trash with our end effector limits the number of moving parts and makes our robot more versatiles in being able to manipulate a wide array of trash. The gripper is limited in the types of objects it can manipulate as the objects must be of a certain size.

Our goals in implementing this system are to provide an autonomous and seamless trash sorting experience over conveyor belts for trash processing facilities. We also intend on achieving a sorting accuracy of 90%, making our project comparable to that of the human manual laborers and potentially superior to them. With SortBot, we aim to create a cleaner and more eco friendly society.

2 USE-CASE REQUIREMENTS

The use-case requirements for our project can be divided into three major components, each crucial to our project's overall success.

2.1 Reliable Sorting

Overall, after the object is identified, the entire system should take no less than 8 seconds to sort the object into the correct bin. The end effector should be able to both pick up and release objects at the correct times with 95% accuracy. Our system will use a suction cup to pick up items that will be lowered and raised so that objects that are adjacent to each other (but not overlapping) can be sorted without issue. The end-effector should be able to deal with common recyclable items. This means it should exert a force strong enough to lift objects weighing up to 1 lbs, and should be able to handle objects that range from 1 in in their shortest dimension to objects that are 8 in in their longest dimension. Additionally, the end-effector must be able to deal with objects that have a wide variety of surfaces, such as smooth, porous, or rough surfaces.

2.2 Real-time Monitoring

As for the monitoring aspect of the system, we want to provide a live stream through a web application. This web application will provide some form of real-time monitoring through bounding box drawing on detected objects. The bounding box will appear in four different colors depend on their trash type: red for plastic, green for waste, blue for metal, and yellow for glass. These visual updates will happen within 50 ms the bounding box metadata. Moreover, this live stream will have a resolution of 720 x 720 pixels and have a frame rate of 30 fps to ensure a good viewing experience on most laptops.

2.3 Real-time Object Detection

The system is designed to provide continuous real-time object detection for objects moving slowly down a conveyor belt. Since waste management facilities are chaotic, our system must support multi-object detection with up to 10 items per image. In addition, the system must be able to detect these objects within 150 ms of them appear in the camera feed, that way the system can actually determine the object's location regardless of how fast the conveyor belt is moving. The detection will be ensure the corresponding bounding box per each object successful lands on the object with a tight fit.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

As you can see in Fig.[1], we use the eYs 3D stereo camera to capture both RGB images of the objects on the conveyor belt as well as their heights. The RGB image will be sent to a bounding box classification model to identify the objects and provide their location. This information, along with the heights of the objects, will be sent serially as xyz coordinates to an Arduino. This Arduino will then control a robotic gantry to move over to the objects, pick them up, and place them into the correct bins. The end-effector of this gantry will use a suction-cup shaped end effector that will pull a vaccum to pick up objects via a vacuum pump. The video, as well as the objects and their bounding boxes, will be sent to a web app for users to monitor the robot's operation.

4 DESIGN REQUIREMENTS

Each of our three primary use-case requirements can be translated into multiple design requirements to support them. In this section we will break them down by use-case requirement.

4.1 Reliable Sorting

For our requirement The system must be able to pick up and move the objects to the correct location reliably and within an acceptable time frame. After the model has identified the centroid, the time that the gantry takes to sort an item should be no more than 8 seconds. Ideally, the gantry should take 2 seconds maximum to move to the centroid, 4 seconds to pick it up, and 2 seconds to move the object above the correct bin and drop it. In order to achieve this, the x-y movement of the system should be able to move at around a rate of 1.5 ft/s, and the z-axis movement should be able to reach speeds of around 1 ft/s. Additionally, in order to ensure that the robot can move to the correct location to pick up the object, the gantry should be able to move in 0.2 mm increments on the x-y axes and in 1 mm increments in the z axis. In order to manipulate items as small as 1in, the diameter of the suction cup should be no larger than 1in. Additionally, the vacuum pump should be able to hold a vacuum of at least 1 lb, and evacuate air quickly enough in order to lift objects that have a porous or rough surface.

4.2 Real-time Monitoring

For our requirement of having real-time monitoring, we are able to derive some more design requirements. In order to achieve a 50 ms second each component: Jetson Orin Nano data processing, network transmission, and broswer rendering must add up to less than 50 ms. The Jetson Orin Nano should take no more than 20 ms, he network transmission should take no more than 10 ms, and finally the browser rendering should take no more than 15 ms. Together this all adds up to 45 ms. When testing the network speed on the CMU-SECURE network, we achieved a 83.2 Mbps upload speed, with a raw data rate of:

$$720 \times 720 \times 3 \text{ bytes} \times 30 \text{ fps} = 46.66 \text{ MB/s}$$

If we were to use H.264 compressiion, we can achieve a sufficient bit rate of:

$$720 \times 720 \times 0.1 \text{ bytes} \times 30 \text{ fps} = 4.70 \text{ MB/s}$$

This leaves us well below the 83.2 Mbps available upload bandwidth: the bandwidth utilization is 4.7/83.2 = 5.65

4.3 Real-time Object Detection

For our requirement of having real-time object detection, we are able to derive some more design requirements. We want the system to consistently identify objects of the same type in the same way, to minimize false positives, thereby reducing waste contamination. We are requiring a precision of 0.70 for this. Moreover, the identified bounding boxes must be lie up properly with their ground truths, in order for the gantry to pick the object. A mIoU of 0.95 would allow the gantry to be within ± 5 pixels of the true bounding box centroid.

5 DESIGN TRADE STUDIES

5.1 Mechanical Movement

In order to physically manipulate the objects in 3D space, there needs to be a mechanical system that can move the objects from the conveyor belt to the bins.

5.1.1 Articulated Robotic Arm

Initially, we considered using an articulated robotic arm to pick and place objects. These arms are often used in production lines as they are able to move with 6+ DoF, meaning that they can also control the orientation of their end-effector as well as its position. However, these systems are often costly, since they require motors and parts that are able to withstand high torques and jerks. Additionally, these arms are restricted by their joint limits, and require the use of inverse kinematics in order to move from one place to another. This means that moving between certain configurations will require additional time as it moves to more suitable configurations, or else the movement becomes impossible.

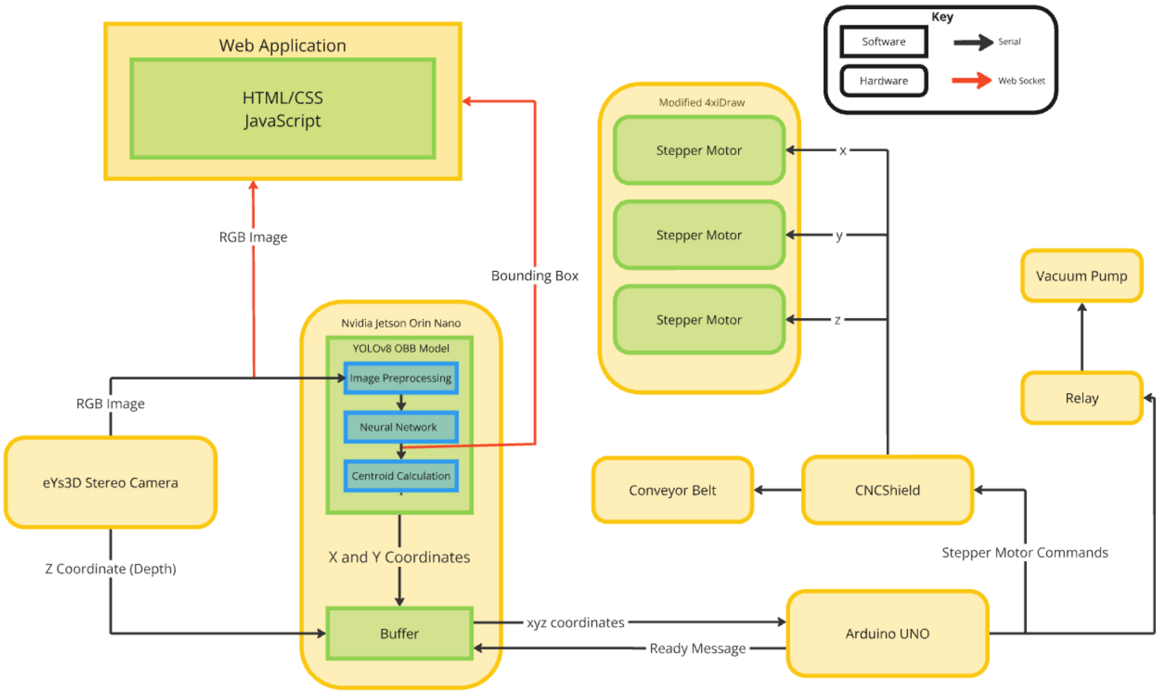


Figure 1: System Architecture

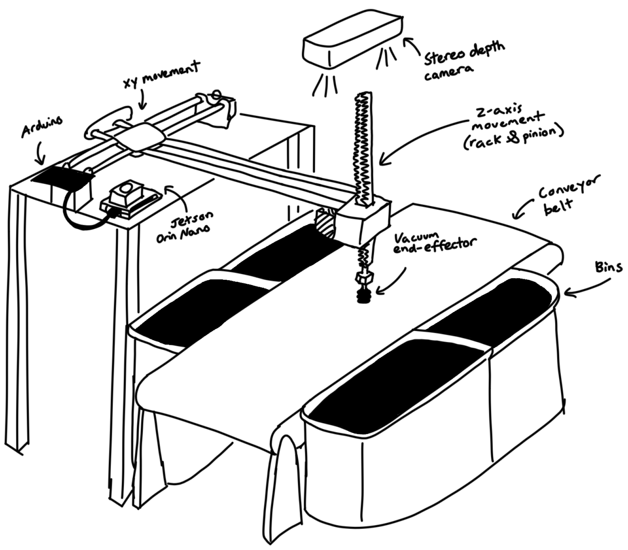


Figure 2: Rough Sketch of System

5.1.2 End-of-Conveyor Ramp

One potential way of sorting the objects into the correct bins is to put a moving ramp which is able to adjust itself such that it guides objects that fall onto it into the correct bin. This implementation is able to deal with heavy objects, since it doesn't need to lift anything off of the conveyor belt, and instead uses the force of gravity to move objects into their respective bins. However, one downside to this implementation is that it doesn't work with objects that are next to each other, as they will fall onto the ramp at the same time, so the ramp is forced to sort two potentially different types of trash into the same bin.

5.1.3 3-Axis Gantry

A robotic gantry (otherwise known as a Cartesian Coordinate or Linear robot) which controls its motion through linear motion in each respective axis. These types of robots are used in a wide variety of applications, such as pick-and-place, welding, plotting, and 3D printing. They are highly precise, and since the movement in each direction is linear, they are able to move to any location that falls within the extremes of each linear motion (which ends up being a rectangular prism). This type of motion is also much simpler, and does not require the use of inverse kinematics. Although these robots only have 3 DoF (x,y,z), it is enough for our application, since the robot only has to move the end-effector down on top of the objects without any rotation.

5.2 Web Application Architecture

In order to stream video with sufficient frames per second, delay, and resolution, the web application must provide a software architecture framework suitable for video streaming.

5.2.1 Django

Django is a Python web application framework. Initially, we considered using the Django web framework for creating the web app and streaming the video. Despite it having various tools useful for account management purposes such as in the case of a social media application, it fails to serve for a mainly streaming application. It has higher than average memory usage per connection. Nonetheless, it is less optimized for real-time streaming workloads which is the main purpose of our application on the web.

5.2.2 Node.js

Node.js is a JavaScript web app framework. It has some oustanding advantages in that it has non-blocking I/O, which allows for multiple concurrent connections with a minizmied overhead. This makes it ideal for streaming applications. It is memory efficient, since it uses less RAM than average per connection in comparison to thread-based frameworks. It has low latency as it minimizes the delay between server processing and client delivery. Finally, it is optimized for streaming as it has native stream API built into the core of the platform. Despite it not having as good account management properties, it provides more than enough in terms of the streaming capabilities that we will be utilizing.

5.3 Deep Learning Model Architecture Selection

Object detection models are categorized into two types: one-stage and two-stage, based on their approach to the detection pipeline. One-stage models treat object detection as a single unified task, meaning that they predict bounding boxes and class probabilities as a single unified task. On the other hand, two-stage models first generate a set of region proposals during the first stage and then during the second stage the model will refine these proposals by classifying the objects and fine-tuning the predicted bounding boxes. While two-stage models tend to be more precise due to their more thorough approach, they are generally much slower and require significantly more computational resources than their counterpart. One-stage models fit our use-case much more as we are willing to sacrifice a bit of precision for a faster inference.

5.3.1 Single Shot Multibox Detector

The Single Shot Multibox Detector (SSD) is an approach proposed by Wei et al. [2] in 2016 during his time as a Ph.D. student at the University of North Carolina at Chapel Hill. SSD discretizes the output space of bounding boxes into a set of default bounding boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. The refinement process is contingent upon the predefined default bounding boxes at each feature map location. While this approach may be effective for many scenarios, it can become a problem when objects appear in unusual sizes, orientations, and locations. Since the refinement process uses a naive approach of using predicted offsets in xy position and width and height values to adjust the default bounding boxes, it is possible that SSD can still fail to properly detect objects in the previously mentioned scenarios. Without guarantees on object size, orientation, and location, SSD is not the right approach for this problem.

5.3.2 EfficientDet

The EfficientDet architecture is a family of object detectors proposed by Tan et al. [3] in 2019 during his time on the Google Deep Brain team. As opposed to previous state of the art approaches, EfficientDet introduces a new

Table 1: YOLO vs EfficientDet

Model	mAP^{val}	CPU ONNX (ms)	T4 TensorRT10 (ms)	Parameters (M)	FLOPS (B)
EfficientDet-d0	34.6	10.2	3.92	3.9	2.54
EfficientDet-d1	40.5	13.5	7.31	6.6	6.1
EfficientDet-d2	43.0	17.7	10.92	8.1	11.0
EfficientDet-d3	47.5	28.0	19.59	12.0	24.9
EfficientDet-d4	49.7	42.8	33.55	20.7	55.2
EfficientDet-d5	51.5	72.5	67.86	33.7	130.0
EfficientDet-d6	52.6	92.8	89.29	51.9	226.0
EfficientDet-d7	53.7	122.0	128.07	51.9	325.0
YOLOv8n	37.3	80.4	1.47	3.2	8.7
YOLOv8s	44.9	128.4	2.66	11.2	28.6
YOLOv8m	50.2	234.7	5.86	25.9	78.9
YOLOv8l	52.9	375.2	9.06	43.7	165.2
YOLOv8x	53.9	479.1	14.37	68.2	257.8

Note: These numbers are from running inference on a 640 x 640 pixel image.

type of feature extractor, the a weighted bi-directional feature pyramid network, for fast multiscale feature fusion, and a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature networks, and box/class prediction networks all at the same time. While EfficientDet takes a more thorough approach to bounding box localization, by using repeated top-down and bottom-up feature fusion, this level of refinement is excessive for our purposes. Since we are allowing a margin of error of ±5 pixels for computed bounding box centroids, dedicating computational resources for finer localization would introduce unnecessary overhead, and would ultimately degrade real-time performance. The additional computational resources needed to take full advantage of the EfficientDet architecture is not justified in our use case, where it is acceptable to sacrifice some precision in bounding box localization if it can lead to significant improvements to real-time performance.

5.3.3 You Only Look Once

The You Only Look Once (YOLO) architecture is a family of object detectors proposed by Redmon et al. [4] in 2016 during his time as a Ph.D. student at the University of Washington. As opposed to previous state of the art approaches, YOLO frames object detection as a regression problem to spatially separate bounding boxes and associated class probabilities. As opposed to other state of the art approaches, YOLO uses a single network to predict bounding boxes and class probabilities. Since the whole detection pipeline fits inside of a single network, YOLO has been optimized end-to-end directly for detection performance. YOLO's appeal comes from its comparable performance to other state of the art approaches despite using a fewer number of parameters. The following Performance Comparison Table 1 between the YOLO architecture and the EfficientDet architecture clearly illustrates this fact. A YOLOv8n model with 3.2M parameter has a greater mAP-VAL on COCO [5], a common object detection benchmark, and faster inference speed on a T4 TensorRT10 (which we can extrapolate to our GPU-based hardware) than its 3.9M parameter EfficientDet counterpart, EfficientDet-d0. This trend continues for YOLO sizes and their EfficientDet counterparts. The strong real-time performance without loss of precision is what makes YOLO very attractive to us.

5.4 Deep Learning Model Classification Head Selection

Object detection is a task in deep computer vision that involves identifying the location and the class of objects in an image or video stream. Objects that are found are given a bounding box, typically represented with an xy coordinate that represents the center of the detected object and a width and height value that is used to indicate the size of the object. These values are inferred from the last layer of object detection models, which is referred to as the classification head. The effectiveness of an object detection model heavily depends on its classification head. In order to select a classification head that we can trust to meet our performance requirements we will need to inspect how each type of bounding box will be visualized on static images.

5.4.1 Axis-Aligned Bounding Box Prediction

Object detection models trained for axis-aligned bounding box prediction infer a detected object's location using the *(x, y, w, h)* format where x and y are the predicted coordinates for the center of the bounding box and w and h are predicted width and height respectively. As illustrated in Fig. 3, we can see that axis-aligned bounding boxes are not very space-efficient, resulting in a less accurate representation of an object's true shape and location.

5.4.2 Oriented Bounding Box Prediction

Object detection models trained for oriented bounding box prediction infer a detected object's location using the *(x, y, w, h, θ)* format where *x, y, w, and h* represent the same values as above, and θ is the angle of rotation. As illustrated in Fig. 3, we can see that oriented bounding boxes provide a much tighter fit. Moreover, we can see that the calculated bounding box centroid lies closer to the true centroid than the axis-aligned bounding box equivalent.

5.5 Central Computing Platform

Being able to achieve on-device deep learning is a key component in being able to achieve our design requirements as local compute not only minimizes the number of external data transfers which will reduce overall latency, but can also provide the necessary control to perform further on-device optimizations.

5.5.1 Nvidia Jetson Nano

The Jetson Nano is a mid-range mobile graphics chip that was released by Nvidia in 2019. Built on the 20 nm process and based on the TM660M-A2 variant of the GM20B graphics processor, the Jetson Nano offers a strong computational power relative to its cost of $99. The Jetson Nano offers an AI performance of 472 GFLOPS and a GPU and CPU max frequency of 921 MHz and 1.42 MHz respectively. These specifications will not be sufficient enough to run modern deep computer vision models which typically require a GPU frequency of 1 GHz.

5.5.2 Nvidia TX2 Jetson

The Jetson TX2 is one of the direct successors to the Jetson Nano. Jetson TX2 can deliver up to 2.5x the performance of the Jetson Nano offering the AI performance 1.33 TFLOPS and a GPU and CPU max frequency of 1.3 GHz and 2.2 GHz respectively. Despite having attractive properties in terms of raw throughput, the Jetson TX2 suffers when it comes to its memory constraints. The Jetson TX2 only has 8GB of LPDDR4 RAM which is unideal in our use cae. Large models and high-resolution inputs will create memory bottlenecks which will lead to slower performance than advertised.

5.5.3 Nvidia Orin Jetson Nano

The Orin Jetson Nano is one of the more modern graphics chips that has come out from Nvidia. Released in 2023, the Orin Jetson Nano offers a promising 67 TOPS and a GPU and CPU max frequency of 1.2 MHz and 1.7 GHz. Unlike its older counterparts the Orin Jetson Nano uses 8 GB 128-bits LPDDR5. Compared to the LPDDR4, LPDDR5 has a high data transfer rate boosting a maximum bandwidth of 6400 Mbps which is nearly 1.5 times faster than the LPDDR4's 4266 Mbps. These specifications combined with the fast data transfer rates makes the Orin Jetson Nano very attractive for our use case.

6 SYSTEM IMPLEMENTATION

6.1 3-Axis Gantry

6.1.1 XY movement

For the xy movement of the gantry (parallel to the plane of the surface of the conveyor belt), we based our design on the 4xiDraw, a robot made to plot svg images onto physical paper. This design involves two perpendicular pairs of rods joined by a sliding clamp which slides across the rods in both directions to allow for movement. The movement is driven by a singular belt, powered by two stepper motors (one for each direction).

6.1.2 Z movement

The z-axis movement (perpendicular to the conveyor belt's surface) will be controlled by another stepper motor which drives a rack-and-pinion gear system. This will be mounted on the ends of the free moving rods in the 4xiDraw, in place of the drawing tool in 4xiDraw's original design.

6.1.3 End Effector

The end-effector, which allows the robot to hold objects while they are being moved, will be a small suction cup attached to a 12 V vacuum pump. The idea is that the suction cup will be pressed onto the surface of a trash, and the vacuum pump will pull a vacuum inside the cup to maintain the grip. This end-effector will be mounted onto the end of the rack gear, so that it can be lowered and raised.

6.1.4 Arduino Control

The stepper motors vacuum pump for the gantry and conveyor belt will be controlled by an Arduino Mega 2560. The Arduino will be connected to a stepper motor controller board (CNCShield) to control the stepper motors. Since the linear motion is controlled by belts, we can translate translational movement into the number of steps that the motors have to turn:

$$\text{num steps} = \frac{\text{linear distance}}{\pi * \text{pitch diameter}} * \left(\frac{\text{steps}}{\text{revolution}}\right)$$

Since the conveyor belt is moving at a constant speed, and we have the timestamp of when the object is detected, the Arduino will be able to calculate where the gantry should move to pick up the moving object. When the gantry has moved to the correct position above the object, the end-effector will be lowered until the suction cup is firmly planted onto the object's surface. The Arduino will then send a signal through a relay in order to activate the 12V vacuum pump, which will pull a constant vacuum to keep the object secured. The Arduino will then move the gantry over to the correct bin (which has a predetermined location), and send the signal to the relay to stop pulling the vacuum, which will then allow the object to release.

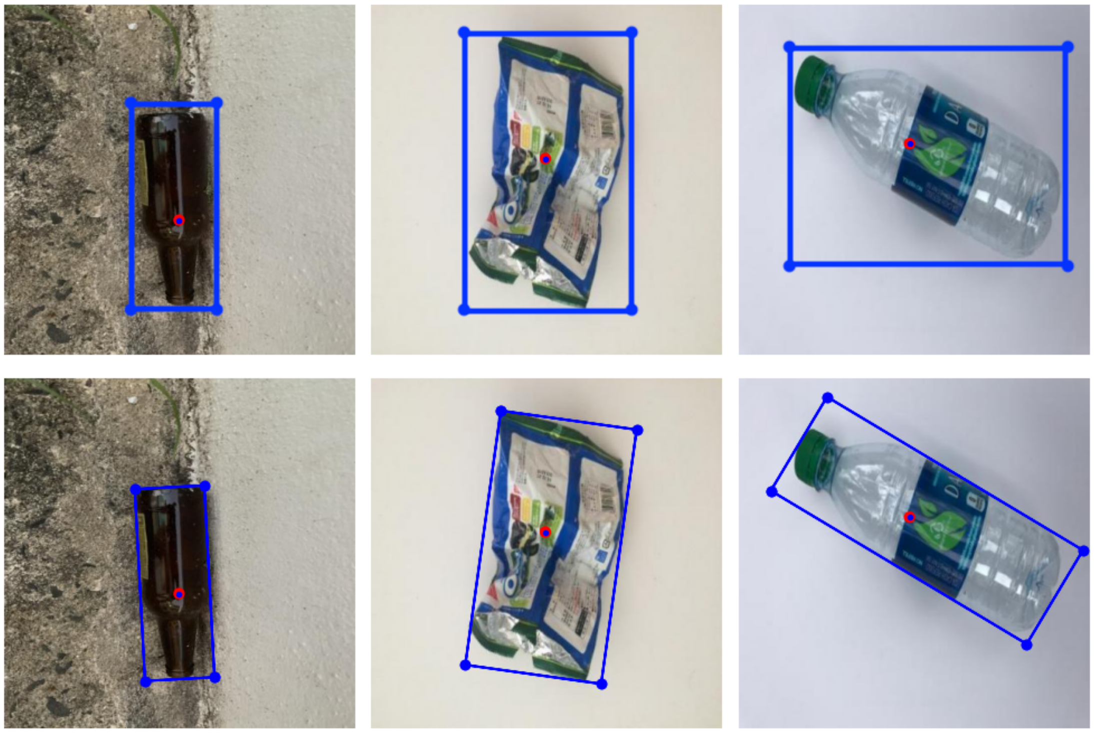


Figure 3: Axis-aligned Bounding Box vs Oriented Bounding Box

Table 2: Central Computing Platform Comparison

	Jetson Orin Nano	TX2 Jetson	Jetson Nano
AI Performance	67 TOPS	1.33 TFLOPS	472 GFLOPS
GPU Max Frequency	1.02 MHz	1.3 GHz	921 MHz
CPU Max Frequency	1.7 GHz	2 GHz	1.43 GHz

6.1.5 Jetson-Arduino Communication

The Jetson Orin Nano will be connected to the Arduino via a USB-A to USB-B cable, and will send information serially. When the centroid of an object is determined, it will be placed on a buffer. When the Arduino indicates via serial communication that it is ready to pick up an object, the Jetson Orin Nano will send a pair of x, y, z coordinates to the Arduino, along with a timestamp of when the object was detected.

6.2 Web Application

6.2.1 Web Socket Communication

For the web application's architecture, data will be communicated through a web socket connection between the Jetson Orin Nano and a designated PC. The PC will instantiate a web socket connection by running a server script. Then, the Jetson Orin Nano will connect to the PC with a client script that has the IP address of designated PC. With both the PC and the Jetson Orin Nano connected to the same Wi-Fi network, a successful web socket connection can be established. The PC will utilize a Node.js interface for handling the server setup and process management.

6.2.2 Information Displayed

As for the frontend visual component of the web application, the server code that instantiates the web socket, will also instantiate the HTML file that is written for generating the HTML. The code for handling the receiving and processing of the data from the web socket connection will be written in JavaScript. This information includes the video frames, as well as classified objects, and their bounding box coordinates/precision metrics. The interface will then draw on top of the video feed to show the bounding boxes and the objects' classification. The app will also keep track of the history of the quantity of objects sorted for each category. Finally, a short summary of the recycling rules of California being followed will be included at the bottom of the monitoring page or on a separate page. These rules will provide the user with an idea of the recycling protocols being followed by our robotic system.

6.3 Deep Learning Model

6.3.1 Pipeline

RGB images from the eYs 3D stereo camera will undergo a bit of feature engineering right before being feed to the model. In particular, we are going to normalize the image on along in each RGB channel respectively using the mean and standard deviation of each RGB channel from the train dataset. This hopefully helps the model generalize better on the real-world images. The specific model we plan to implement is a YOLOv8 OBB model, so we can take advantage of the efficient inference speed of the YOLO architecture and the tight bounding box fit from the OBB classification head.

6.3.2 Training

Once the deep learning pipeline has been implemented, we can then transition into training the model for a sufficient amount epochs. Before, we can train the model on our trash dataset [6], we first to need to do a bit of image preprocessing. Out the box, the images in the dataset come from a variety of background, we plan to standardize the background such that all images "laying" on the same color background as the conveyor belt. This can easily be done since the dataset also comes with the polygon bounding box as well. We plan to start training with an initial learning rate of $\eta = 0.001$ toensure stable training and to avoid overly large updates that may force us out of a global minimum. Moreover, we plan to use the AdamW optimizer and the ReduceLROnPlateau learning rate scheduler to help easy our convergence. The combined decoupled weight decay in addition to reducing the learning rate only when validation precision decreases will support the model's ability to generalize especially on real-world objects. We plan to train on Google Colab T4's using the \$300 free credits provided to new users ($3 \times \$300$ hours = \$900 hours to train).

7 TEST & VALIDATION

7.1 Test for Reliable Sorting

In order to ensure that the gantry system can reliable pick up and drop off from their location on the conveyor belt to their correct bin, we plan to measure things: (i) the performance of the z-axis end-effector and (ii) the real-world precision of the programmable movement.

7.1.1 Z-Axis End-Effector

To analyze the performance of the z-axis end-effector, we will create our own assortment of trash objects and recyclables that the z-axis end-effector will pick up and move. For this custom assortment, we plan to use items of various shapes, sizes, weights, and textures to ensure the end-effector generalizability for real-world trash. We also plan to run multiple trials per object, making sure to place them at different orientations to ensure that the system is adequately robust.

7.1.2 Precision of Real-World Movement

Since, we are translating code to real-world movement, we need to ensure the movement of the gantry system meets our previously requirements. We plan to test this by marking static locations on the conveyor belt and programming the gantry to move to those location. Once the gantry stops we will then measure the distance from where the gantry stopped to the desired location. We plan to run multiple trials of this, each where the gantry system starts at a different origin location.

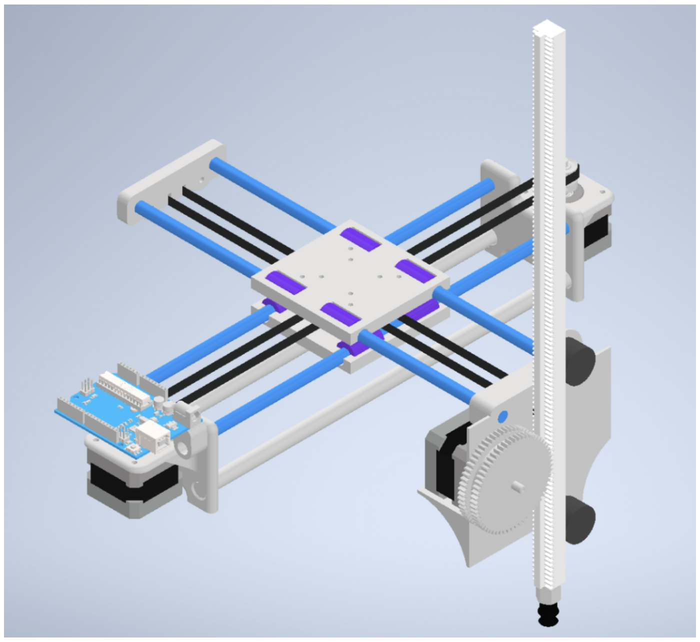


Figure 4: Design Rendering

7.2 Test for Real-time Monitoring

In order to ensure that our web application meets our requirements, we plan to measure two things: (i) latency between messages and (ii) speed of bounding box drawing.

7.2.1 Latency Between Messages

We plan to use iPerf to measure the speed of data transfer of two different formats of data: (i) RGB images (provided from the camera) and (ii) structured metadata (the predicted bounding box). For RGB images, we will be using iPerf's TCP mode to determine maximum bandwich utilization and for structured metadata will be using iPerf's UDP mode to measure jitter and packet loss.

7.2.2 Stress Testing Bounding Box Drawing

Since the maximum number of objects on the conveyor is variable, as we would like to simulate how actually waste management facilities are, we need to be to ensure bounding boxes are draw at a relatively fast pace to ensure real-time monitoring. To stress test this mechanism, we plan to continuously send dummy bounding boxes of all sizes to web application to see the rate at which they show up. More concretely, we have to measure how fast a batch of dummy bounding boxes are rendered after they are send from the Jetson Orin Nano.

7.3 Test for Real-time Detection

In order to ensure that the deep learning model meets our requirements, we plan to roll out a three-phase testing plan: (i) visualizing predictions on single object toy problems, (ii) visualizing predictions on multi-object toy problems, and (iii) visualizing predictions on real-life toy problems. Only after these three tests pass, we will then integrate the deep learning model into the overall system.

7.3.1 Single Object Toy Problems

Once the deep learning model has reached adequate performance on our train data set, we will then transition into analyzing its performance on our test data set. In particular, we are going to look at how the model generalizes on "easy" problems. We define "easy" to be images with only one object in focus. For this stage, we are choosing to ignore images that have improper lightening and images where objects are obscured by the background. The model will only pass this phase if the computed centroid is within ±5 pixels of the of the true centroid.

7.3.2 Multi-Object Toy Problems

Once the deep learning model passed the selected images from the *Single Object Toy Problems* stage, we can then transition into analyzing its performance on multi-object toy problems. In particular, these images will be much harder than the images from the previous test set (i.e. the background is able to obscure the object, objects are allowed to overlap, and etc.). The idea behind this stage is to test how the model will perform on images that are indicative of images the model would see in a waste management facility. Similarly to the last stage, if the computed centroid is within ±5 pixels of the of the true centroid.

7.3.3 Real-Life Toy Problems

Once the deep learning model passed the selected images from the *Multi-Object Toy Problems* stage, we can then transition into analyzing its performance on real-time images. Images for this dataset will be collected as followed: objects will be placed on the conveyor belt in different orientations and locations, random lightening will be selected, and then the photo will be taken. From here the images will be human-annotated with a bounding box. The idea behind this stage is to test how the model will perform in our system. Similarly to the last stage, if the computed centroid is within ±5 pixels of the of the human-annotated centroid.

8 PROJECT MANAGEMENT

8.1 Schedule

Our schedule is split into four main sections: the xy movement of the gantry, the z-axis movement, the web application, and the deep learning model. We have made the schedule such that the entire system should be fully assembled and essentially complete by the beginning of April. The reasoning behind this is that we would like to have it mostly complete before the interim demo, and that month of April can be used to refine the end-to-end system. However, if necessary, this extra time will provide slack in case certain parts of the project fall behind schedule. We are on track as of now and are maintaining a steady pace to reach our April goal. The full schedule is shown in Fig. 5.

8.2 Team Member Responsibilities

We have divided team member tasks and responsibilities based on each member's past experience in each field and current interests. Teddy Lin would be working mainly on designing a custom z-axis end effector that can satisfy the design requirements mentioned above and programming movement into the x, y, z directions. Alejandro Miranda will be working on developing a web stack necessary for real-time live-streaming and creating a user interface that helps visualize of the system, as well as obtaining a depth map from the stereo camera to obtain z-coordinates for the gantry movement. Ethan Lu will be working on implementing a YOLOv8 OBB model and optimizing model performance for image classification accuracy and bounding box mIoU. The construction of the full system as well as end-to-end testing and verification will be done together as a team.

8.3 Bill of Materials and Budget

The current Bill of Materials for SortBot is shown in Table 3. It includes all currently purchased parts, their quantities, vendors, and final prices.

8.4 TechSpark Useage Plans

We plan to meet in TechSpark once a week on Wednesday during the allotted registrar timeslot for 18-500: ECE Design Experience (10:00 a.m. to 11:50 a.m.). At TechSpark, we hope to take advantage of the many manual tools — wrenches, screwdrivers, and etc. — to assemble our project and as well use the additional to perform end-to-end testing.

8.5 Risk Mitigation Plans

8.5.1 Suction-Type End-Effector Not Performant

It is possible that a suction-cup end effector might not exert enough force to consistently pick up objects even when a vacuum is pulled, or that it might not be able to deal with very rough or porous surfaces. If this happens, we will move to using a claw-shaped end effector that will either "grab" or "scoop" up objects.

8.5.2 Estimated Position Is Not Reliable

The current plan is to use our knowledge of the speeds of the conveyor belt and the gantry, along with the time that an object is detected, in order to determine the object's location so it can be picked up. However, if this method does not yield accurate enough results to reliably pick up objects, we could potentially stop the conveyor belt when an object is detected so that the position of the object after detection does not change.

8.5.3 Web Socket Communication Is Too Slow

Currently, we plan to use web sockets to establish a communication protocol between the Jetson Orin Nano and the web application. Web sockets offer a more sleek approach to monitoring as the user does not need to be close to the system to see its progress. Unfortunately, there are a lot of variables that introduce overhead this communication method that is not in our control. This like reliability of internet connection and potential congestion would directly impact our streaming. One solution to this problem would be to switch from web socket-based communication to a more local serial-based communication. Serial communication would remove all these previously mentioned variables as it is just one device communicating with another directly, and speed will be determine solely based on bit throughput.

8.5.4 Deep Learning Model Is Too Slow

While the YOLO architecture is optimized architecturally, combining bounding box localization and class prediction into one task, there is still a possibility that YOLO will not be fast enough to run on the Jetson Orin Nano. In this scenario, we will need to look into hardware-level optimization using SDKs such as TensorRT and ONNX. These SDKs are designed for high-performance deep learning inference, as they focus on aggressive optimizations for Nvidia graphics chips. Respectively, these two these techniques offer up to a 2x and 1.5-2x speed up for YOLO-based architectures.

8.5.5 Deep Learning Model Is Not Precise

In the speed and accuracy trade offs, we chosen to side in favor of speed by using a one-stage detector. Architecturally, one-stage detectors will never be as good two-stage detector due to their lack of a dedicated bounding box refinement process. If the YOLOv8 OBB model can not predict precise enough bounding boxes, we will need to move over to a two-stage detector like Cascade R-CNN [7]. While Cascade R-CNN will be more precise with its bounding box localization, it is important to note that it will introduce additional latency that can not be optimized and we would need to identify other bottlenecks in the deep learning pipeline to optimize out in order to achieve real-time speeds.

9 RELATED WORK

9.1 AMP Delta

AMP's Delta robotic sorter has a very similar design and purpose. It's a robot made to sort recyclables from trash, which uses a suction-type end-effector to pick up objects and move them away from the conveyor belt. However, this robot uses the "Delta" gantry design, which uses three pairs of rods in a triangular shape that change in angle in order to move the end-effector where it needs to go.

9.2 Zen Robotics

Zen Robotics has made a waste sorting machine meant for lifting heavy-duty trash, such as wood and stone. They do use a linear gantry system, of using a suction-type end effector, they use a large, excavator-like claw to scoop and grab heavy items and throw them away from the main conveyor belt. Our design will be similar, however it will be intended for lightweight trash sorting.

10 SUMMARY

Overall, we propose a design that we hope will be able to automate the process of sorting trash to make it more efficient, economical, and safe. Our custom-trained YOLO model using oriented bounding box prediction will allow the robot to quickly and accurately identify the existence and location of recyclables on the conveyor belt. The use of a Cartesian/Linear robotic gantry should provide the robot with a large and highly precise range of motion, and the vacuum end effector should be able to pick up a wide variety of recyclable items with ease. Lastly, our web interface will allow the user to monitor the robot's tracking and history in real time.

Some key challenges of our design will be configuring the end-effector to work on objects with irregular surfaces, generating rapid and accurate motion for Cartesian movement, and reducing latency on both the detection of the objects and the transmission of data to the user interface.

Glossary of Acronyms

- COCO - Common Objects in COntext dataset
- CPU - Central Processing Unit
- DoF - Degrees of Freedom
- GFLOPS - Giga Floating Point OPerations Per Second
- GPU - Graphics Processing Unit
- I/O - Input/Output
- mAP^{VAL} - mean Average Precision for VALidation dataset
- Mbps - Megabits per second
- MBps - MegaBytes per second
- mIoU - mean Intersection over Union
- OBB - Oriented Bounding Box
- ONNX - Open Neural Network eXchange
- RAM - Random Access Memory
- R-CNN - Region-based Convolutional Neural Networks
- RGB - Red Green Blue
- SDK - Software Development Kit
- SSD - Single Shot multibox Detector
- TCP - Transmission Control Protocol
- TOPS - Tera Operations Per Second
- UDP - User Datagram Protocol
- USB - Universal Serial Bus
- YOLO - You Only Look Once

References

1 Por. "How Much Do We Waste? A Data-Driven Guide to Waste and Landfills - Meuresíduo." meuResíduo, 26 July 2022.

2 Liu, Wei, et al. "SSD: Single Shot MultiBox Detector.", EECV

3 Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection.", CVPR

4 Tan, Mingxing, et al. "EfficientDet: Scalable and Efficient Object Detection.", CVPR

5 Lin, Tsung-Yi, et al. "Microsoft COCO: Common Objects in Context.", EECV

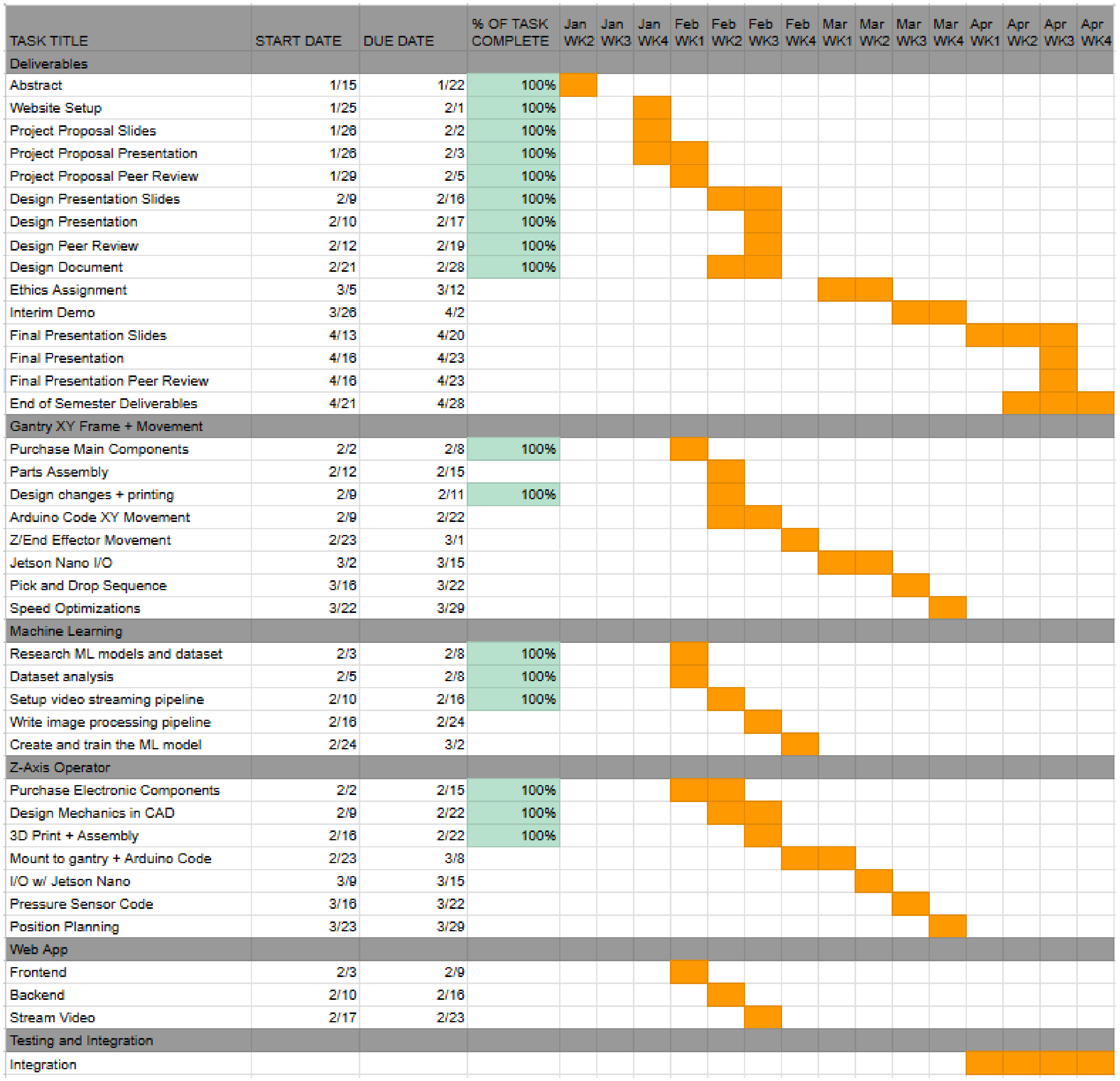
6 Trash Dataset for Oriented Bounded Box, Roboflow, `https://universe.roboflow.com/trash-dataset-for-oriented-bounded-box/trash-detection-1fjjc`

7 Cai, Zhaowei, and Nuno Vasconcelos. “Cascade R-CNN: Delving into High Quality Object Detection.” CVPR

Table 3: Bill of materials

Description	Manufacturer	Quantity	Cost @	Total
One Nema 17 Steppers	STEPPERONLINE	4	\$9.99	\$39.96
Two 8 mm Smooth Rod	McMaster-Carr	2	\$37.50	\$75.00
Eight LM8UU Bearings	MYCNCSHOP	1	\$10.99	\$10.99
Two 20-tooth GT2 Pulleys	WINSHINN	1	\$6.99	\$6.99
Ten F623ZZ Bearings	uxcell	1	\$8.99	\$8.99
One Arduino Mega 2560 Rev3	Ardunio	1	\$53.14	\$53.14
One CNC Shield and Two Pololu Stepsticks	ACERIRMC	1	\$9.99	\$9.99
One GT2 Belt	SeekLiny	1	\$9.99	\$9.99
Two M10 Threaded Rods 1000 mm	McMaster-Carr	2	\$14.86	\$29.76
Eight M10 Nuts	McMaster-Carr	1	\$3.77	\$7.54
One M3 Screw Kit	Kadrick	1	\$7.99	\$7.99
One 12V 2A Power Supply	NC	1	\$7.99	\$7.99
Two Bearing Wheels	AFUNTA	1	\$13.49	\$13.49
One Vacuum Cup	McMaster-Carr	1	\$11.02	\$11.02
One Vacuum Pump DC 12V 12W	NC	1	\$26.99	\$26.99
Four Relays	AEDIKO	1	\$6.99	\$6.99
One eYs3D Stereo Camera	eYs3D Microelectronics	1	\$350.00	\$0*
One Nvidia Jetson Orin Nano	Nvidia	1	\$249.00	\$0*
			Total	**\$326.82**

Note: The eYs3D Stereo Camera and Nvidia Jetson Orin Nano were provided to us by Lending



TASK TITLE	START DATE	DUE DATE	% OF TASK COMPLETE	Jan WK2	Jan WK3	Jan WK4	Feb WK1	Feb WK2	Feb WK3	Feb WK4	Mar WK1	Mar WK2	Mar WK3	Mar WK4	Apr WK1	Apr WK2	Apr WK3	Apr WK4
Deliverables																		
Abstract	1/15	1/22	100%	🟧														
Website Setup	1/25	2/1	100%			🟧												
Project Proposal Slides	1/26	2/2	100%			🟧												
Project Proposal Presentation	1/26	2/3	100%			🟧	🟧											
Project Proposal Peer Review	1/29	2/5	100%				🟧											
Design Presentation Slides	2/9	2/16	100%					🟧	🟧									
Design Presentation	2/10	2/17	100%						🟧									
Design Peer Review	2/12	2/19	100%						🟧									
Design Document	2/21	2/28	100%					🟧	🟧									
Ethics Assignment	3/5	3/12									🟧	🟧						
Interim Demo	3/26	4/2											🟧	🟧				
Final Presentation Slides	4/13	4/20													🟧	🟧	🟧	
Final Presentation	4/16	4/23															🟧	
Final Presentation Peer Review	4/16	4/23															🟧	
End of Semester Deliverables	4/21	4/28														🟧	🟧	🟧
Gantry XY Frame + Movement																		
Purchase Main Components	2/2	2/8	100%				🟧											
Parts Assembly	2/12	2/15						🟧										
Design changes + printing	2/9	2/11	100%					🟧										
Arduino Code XY Movement	2/9	2/22						🟧	🟧									
Z/End Effector Movement	2/23	3/1								🟧								
Jetson Nano I/O	3/2	3/15									🟧	🟧						
Pick and Drop Sequence	3/16	3/22											🟧					
Speed Optimizations	3/22	3/29												🟧				
Machine Learning																		
Research ML models and dataset	2/3	2/8	100%				🟧											
Dataset analysis	2/5	2/8	100%				🟧											
Setup video streaming pipeline	2/10	2/16	100%					🟧										
Write image processing pipeline	2/16	2/24							🟧									
Create and train the ML model	2/24	3/2								🟧								
Z-Axis Operator																		
Purchase Electronic Components	2/2	2/15	100%				🟧	🟧										
Design Mechanics in CAD	2/9	2/22	100%					🟧	🟧									
3D Print + Assembly	2/16	2/22	100%						🟧									
Mount to gantry + Arduino Code	2/23	3/8								🟧	🟧							
I/O w/ Jetson Nano	3/9	3/15										🟧						
Pressure Sensor Code	3/16	3/22											🟧					
Position Planning	3/23	3/29												🟧				
Web App																		
Frontend	2/3	2/9					🟧											
Backend	2/10	2/16						🟧										
Stream Video	2/17	2/23							🟧									
Testing and Integration																		
Integration													🟧	🟧	🟧	🟧

Figure 5: Gantt Chart