Portland State University
Maseeh College of Engineering and Computer Science
Electrical and Computer Engineering Department
EE478 - Intelligent Robotics
Fall 2023

SiriusX

A prototype for satellite debris removal robot
(lower earth orbit)

Designed by: **Wa'el Al Kalbani**
Instructor: **Dr. Marek Perkowski**
Date: December 1st, 2023

Table of Content

1. Introduction

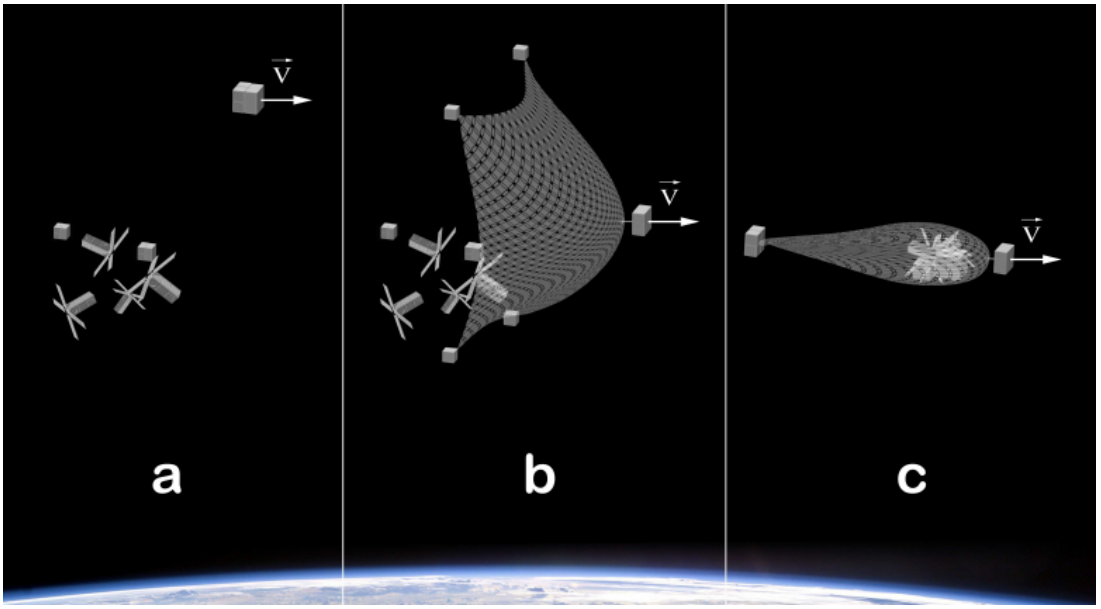
1.1. Problem Statement

Developing an Autonomous Satellite Debris Removal System Using MasterPi Robot: A Solution for Enhancing Space Sustainability and Lower Earth Orbit Management

1.2. Background research

Because of the uncertainty of their population, trajectory, mass, size, and other characteristics, space debris pose a growing threat to space infrastructure and operations. The amount of space debris has increased significantly over the past few decades, and it is expected to increase exponentially over the next few decades (Ivanov et al., n.d.).

Many ideas have been proposed to mitigate the space junk issue, which include intelligence swarm satellites as shown in *Figure 1*. The project mainly uses two types of small satellites. The first one acts as an observer that detects space debris using sensors. The second satellite is an aggregator, which will gather and remove debris using a swarm configuration.



Source:Ivanov et al., n.d. <spacemic.net>

Figure 1: Phases of aggregating space debris using intelligent satellite swarms

1.3. Scope of project

This project, centered around the MasterPi robot, marks a pivotal advancement in the realm of satellite management and debris mitigation. The robot's cutting-edge capabilities address a critical need in the removal of defunct satellites from lower Earth orbit. By harnessing sophisticated camera-based analysis and refined obstacle avoidance techniques, the MasterPi demonstrates the potential to revolutionize the way we approach space sustainability. This innovation heralds a promising future, one where advanced robotics play a key role in safeguarding our celestial environment.

2. Machine learning Algorithm

As *Figure 1* displays, a Machine Learning Algorithm implemented using Orange Data Mining software. The algorithm is designed to process images sourced from a directory titled 'debris,' comprising eight distinct folders labeled Astronaut, CubSat, LaunchVehicle, Meteoroid, Payload, Planet, Satellite, and Tool. These images undergo a series of transformations, beginning with Image Embedding widgets, followed by data testing and scoring. The algorithm utilizes six learners - Support Vector Machine (SVM), Random Forest, AdaBoost, Logistic Regression, Naive Bayes, and Neural Network - to analyze and classify the images. Upon completion, the algorithm employs a Confusion Matrix to evaluate the effectiveness of the classification process.

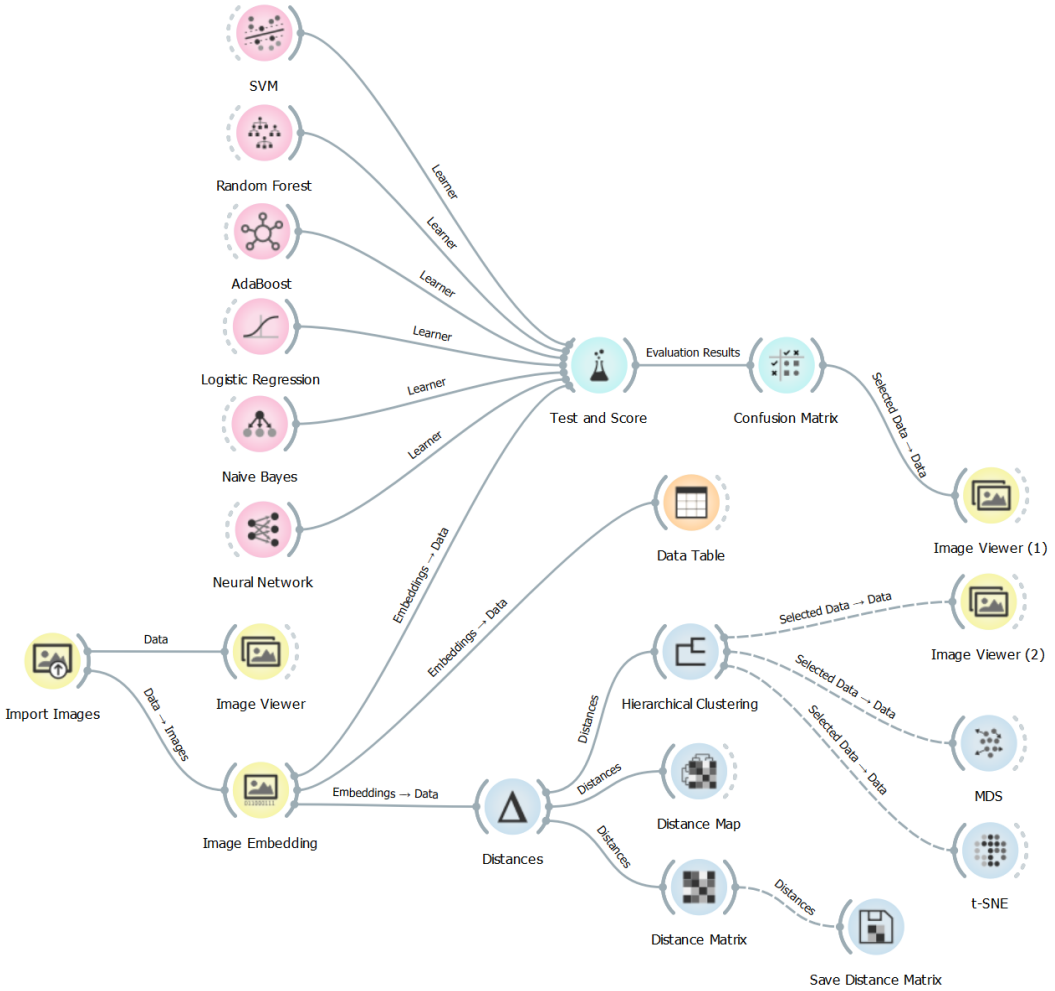


Figure 2: Orange Data Mining Algorithm for Image Processing

Table 1 demonstrates the decimal values for Area under ROC is the area under the receiver-operating curve. Classification accuracy is the proportion of correctly classified examples. F-1 is a weighted harmonic mean of precision and recall. Precision is the proportion of true positives among instances classified. Recall is the proportion of true positives among all positive instances in the data. The Matthews correlation coefficient (MCC), instead, is a more reliable statistical rate which produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements in the dataset.

Table 1: Average Probabilities of Image Classes

Model	AUC	CA	F1	Prec	Recall	MCC
Logistic Regression	0.953	0.793	0.788	0.787	0.793	0.760
Naive Bayes	0.873	0.141	0.124	0.384	0.141	0.200
Neural Network	0.946	0.786	0.782	0.782	0.786	0.754
AdaBoost	0.694	0.472	0.473	0.480	0.472	0.389
Random Forest	0.782	0.479	0.463	0.478	0.479	0.394
SVM	0.924	0.603	0.574	0.732	0.603	0.567

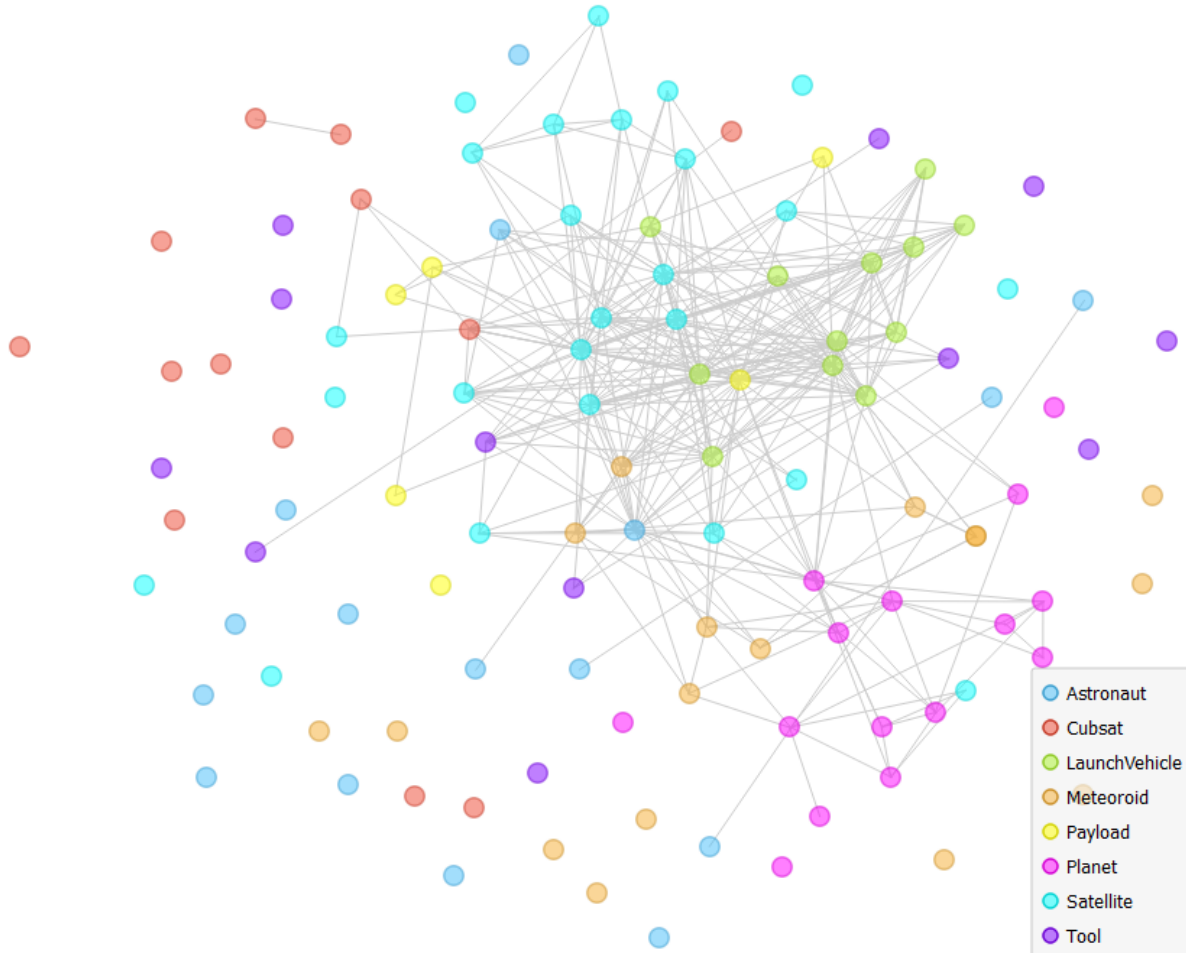


Figure 3: MDS visualization for all image classes

Presented here is an MDS (Multidimensional Scaling) data visualization, an unsupervised technique, illustrating the relationships and disparities among various image classes: Astronaut, CubSat, LaunchVehicle, Meteoroid, Payload, Planet, Satellite, and Tool. This visualization, created using the Orange Data Mining program, offers a comprehensive overview of the dataset's structural organization. Employing the neural network contrasts and intersections, the MDS representation delineates similarities and dissimilarities between different image categories. Through this visual exploration, the MDS visualization offers insights into the underlying patterns, similarities, and distinctions among the diverse image classes, facilitating a deeper understanding of their inherent relationships within the dataset.

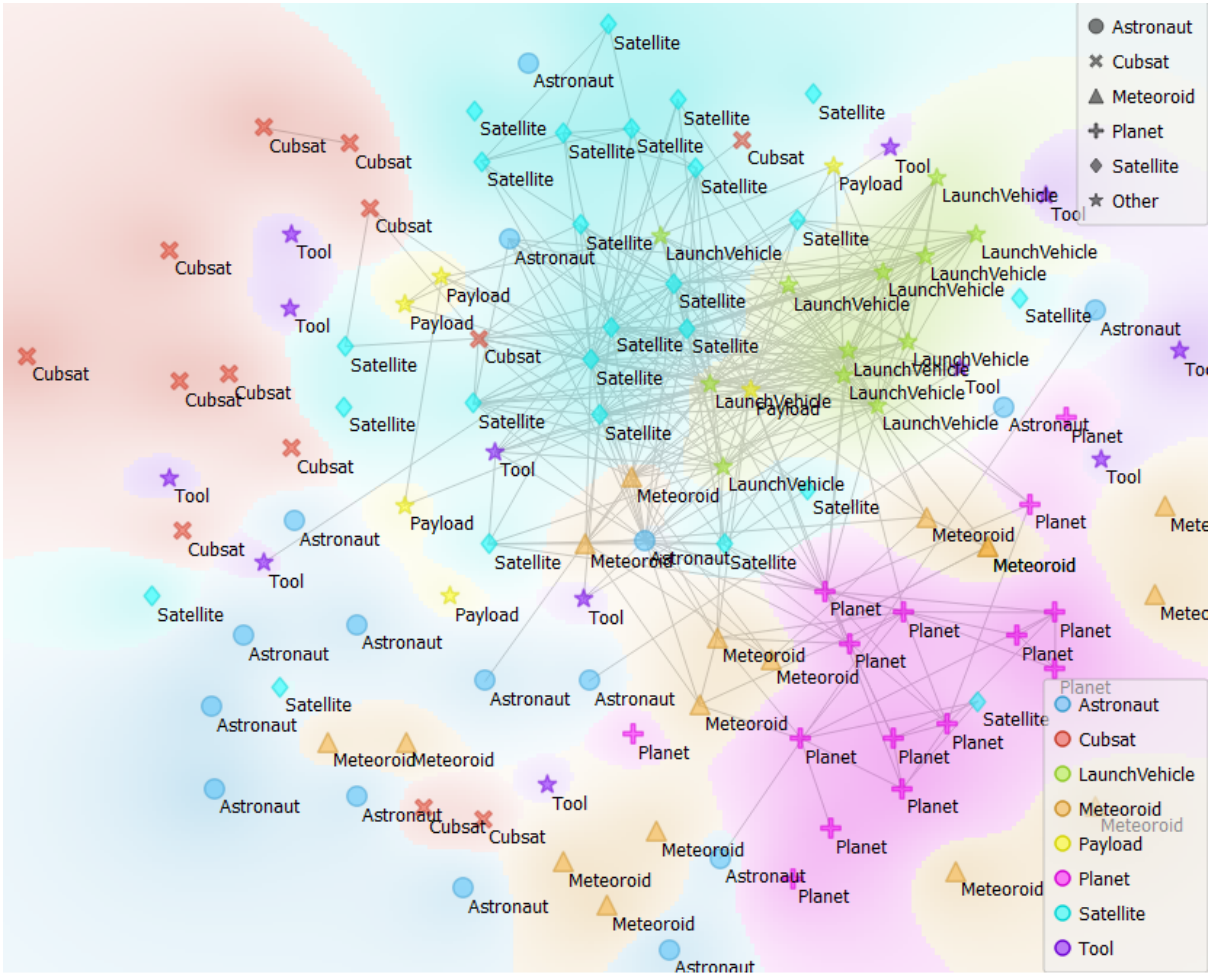


Figure 4: MDS visualization for all image classes with color coding

Predicted

Actual	Astronaut	Cubsat	LaunchVehicle	Meteoroid	Payload	Planet	Satellite	Tool	Σ
Astronaut	37	0	0	0	0	0	3	0	**40**
Cubsat	1	21	0	0	0	0	8	0	**30**
LaunchVehicle	0	0	26	0	4	0	0	0	**30**
Meteoroid	0	1	0	37	0	0	2	0	**40**
Payload	0	1	5	0	7	0	0	7	**20**
Planet	1	0	0	2	0	36	0	1	**40**
Satellite	4	4	0	2	0	5	44	1	**60**
Tool	0	0	0	0	1	0	7	22	**30**
Σ	**43**	**27**	**31**	**41**	**12**	**41**	**64**	**31**	**290**

Figure 5: Confusion matrix for Logistic Regression (showing number of instances)

The Confusion Matrix results table for Logistic Regression provides a comprehensive overview of the model's predictive performance across various classes. It presents the number of instances where the model correctly or incorrectly predicted each class against the actual labels. This matrix serves as a vital tool in evaluating the classifier's effectiveness in distinguishing between different categories. Observing the diagonal values signifies the correct predictions made by the algorithm for each class, while the off-diagonal elements reveal misclassifications. For instance, the model accurately predicted 37 instances of the 'Astronaut' class out of 40, while misclassifying 3 'Astronaut' images as 'Satellite'. Similar analysis across all classes aids in understanding the strengths and weaknesses of the Logistic Regression model in accurately categorizing the images, guiding potential improvements for enhanced predictive accuracy and reliability. The figure below displaying the Orange algorithm's classification of 'Payload' images during a 'LaunchVehicle' detachment is a crucial test scenario. It showcases the algorithm's ability to differentiate and sort objects in complex visual contexts. The associated confusion matrix further evaluates the algorithm's precision in correctly identifying 'Payload' instances. This visual demonstration, along with the matrix analysis, provides valuable insights into the algorithm's effectiveness for real-world image classification, particularly in space-related contexts.

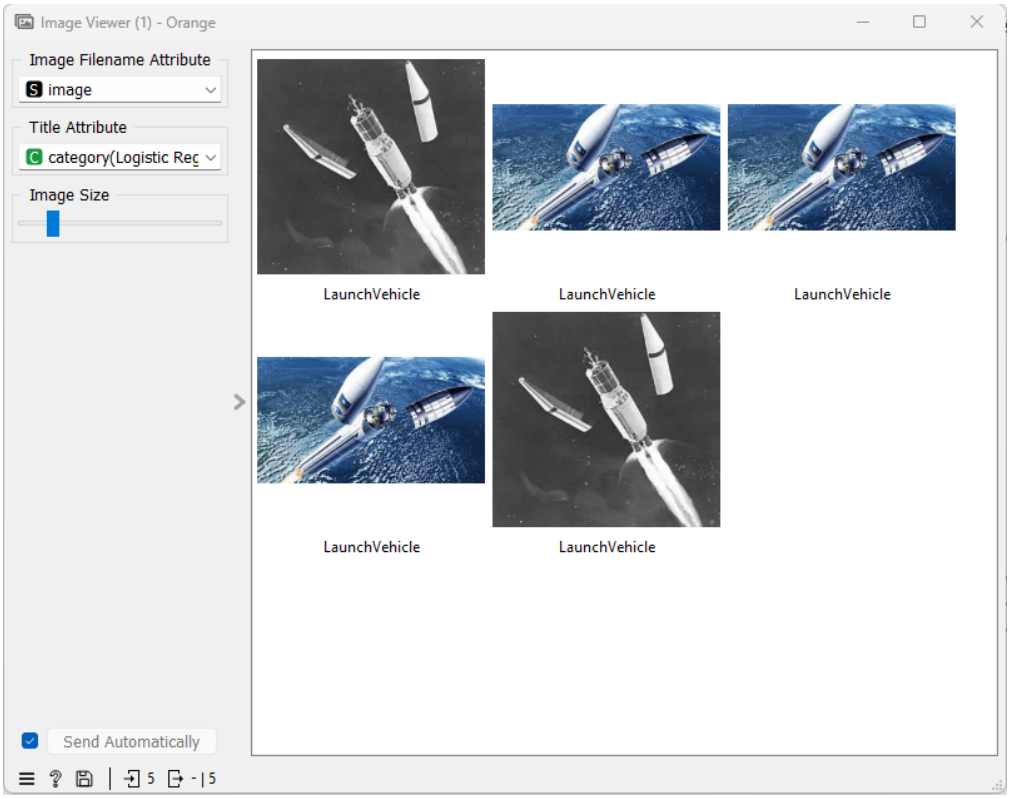


Figure 6: Image example after confusion matrix for Logistic Regression

3. Background

Mobile robots with arms, or mobile manipulators, combine mobility with the ability to manipulate objects. They excel in tasks requiring complex interactions, such as grasping and positioning. Their applications span industries like manufacturing, logistics, healthcare, and research, automating tasks that once relied solely on human intervention. Advanced control algorithms and sensing technologies have propelled the adoption of these robots in various dynamic environments.

Figure 7: MasterPi Hiwonder Robot

3.1. Primary Reference Robot

For this project, *MasterPi Hiwonder* will be utilized as a base robo. Which is a robot driven by a Raspberry Pi 4B, it features a *mecanum* wheel chassis, 5DOF robotic arm, and an HD camera. Utilizing OpenCV, it excels in tasks like color sorting, target tracking, and line following. With an RGB glowy ultrasonic sensor, it can handle obstacle avoidance and light control. Expand its capabilities with additional sensors. Abundant tutorials and Python codes make it a perfect tool to dive into AI and robotics.

4. Applications and Benefits

The robot's advanced capabilities hold great potential in the field of satellite management. Serving as a prototype for an innovative object collection arm, it offers a promising solution for the critical task of removing defunct satellites from lower Earth orbit. Its sophisticated camera-based analysis enables intelligent decision-making, allowing the robot to determine whether an object should be collected based on stored data. Moreover, its obstacle avoidance feature ensures safe and efficient operation, distinguishing between collectible and non-collectible objects with precision and reliability. This technology presents a significant leap forward in satellite debris mitigation efforts.

5. System Components

- **Processor:** Raspberry Pi 4Bb
- **Chassis:** Mecanum wheel design
- **Robotic Arm:** 5 degrees of freedom (5DOF)
- **Camera:** HD, 120° visual angle (180° tilt angle when combined with mecanum chassis)
- **Ultrasonic Sensor:** Glowy ultrasonic sensor for obstacle detection and light control
- **Control Ports:** 6-channel PWM servo ports with over-current protection
- **Additional Features:** Independent power switch, MCU for workload offloading, RGB light, buzzer, LED light, 3 IIC ports, multiple GPIO pins, 4PIN anti-reverse ports compatible with Hiwonder sensors.

6. Functional Overview

6.1. Decomposition Overview

Overall, an array of technical functionalities is established as shown in *Appendix.A*, each contributing to its exceptional performance. From omnidirectional mobility to precise object manipulation and advanced vision recognition, it represents a leap forward in robotic innovation.

6.2. Omni-directional Movement

As *figure 2* shows, the robot showcases omni-directional movement, a feat achieved through its innovative mecanum wheel design. This allows the robot to seamlessly move in any direction, including forward, backward, sideways, and even diagonally, providing unparalleled agility. The accompanying image vividly illustrates this dynamic movement capability, showcasing the robot's versatility and adaptability in navigating through intricate routes with ease.

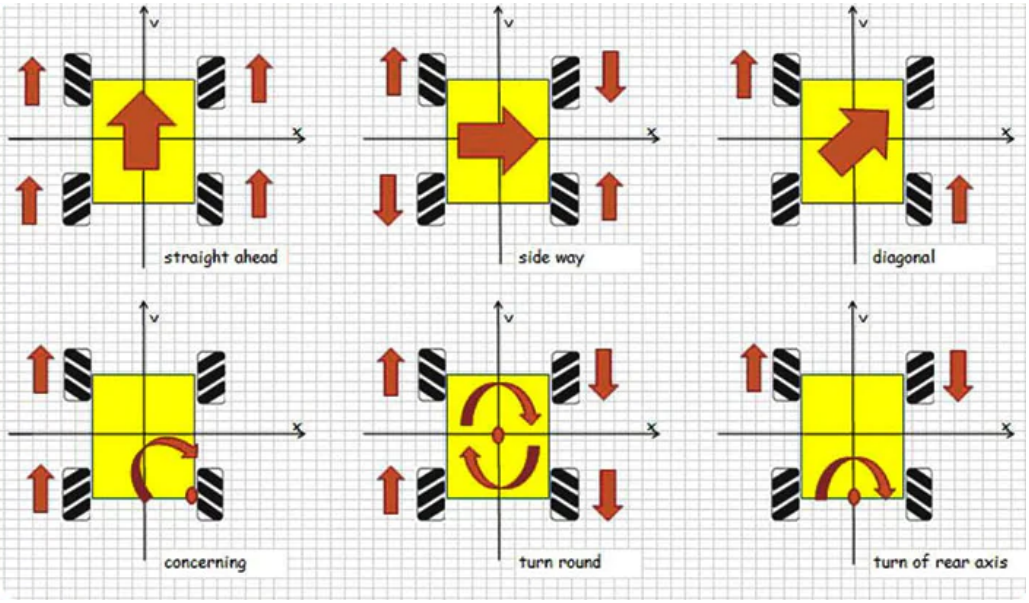


Figure 8: Omni-directional Movement Diagram

6.3. Control System

The combination of RaspberryPi 4B and RaspberryPi expansion board will be implemented in this project to significantly enhance performance.

- Boots processing speed, multimedia performance, memory, and connectivity.
- Paired with the expansion board, it enhances MasterPi's AI capabilities significantly.
- It ensures stable power supply with an independent switch.
- Offers 6-channel PWM servo ports with over-current protection.
- An integrated MCU reduces Raspberry Pi's workload, optimizing resources.
- Includes RGB light, buzzer, LED light, and more for added functionality.
- Three IIC ports enable expansion possibilities.
- Accessible GPIO pins for seamless development.
- 4PIN anti-reverse ports for Hiwonder sensor compatibility.

7. Technical Specifications

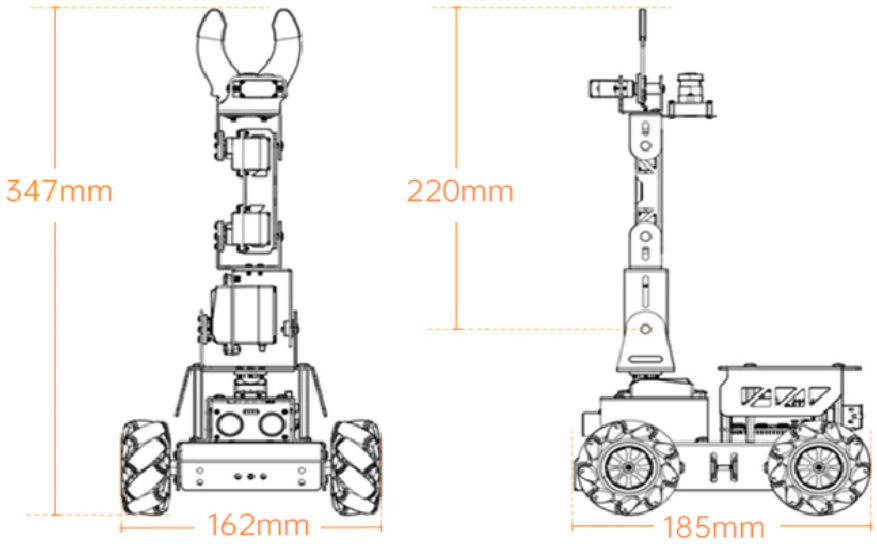


Figure 9: MasterPi Hiwonder Dimensional Diagram

7.1. Object Analysis and Collection

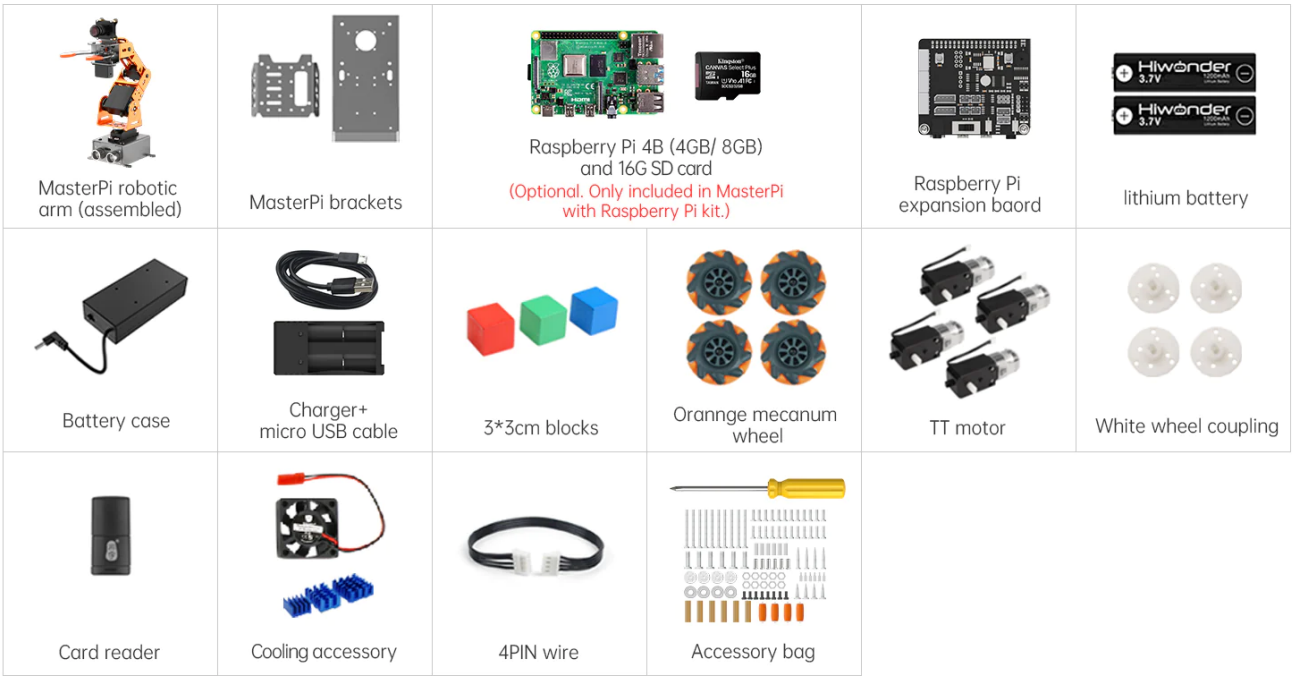
In the term of object analysis and collection, MasterPi employs a multi-faceted approach. Firstly, it utilizes advanced vision positioning and inverse kinematics to impeccably track designated objects. Moreover, for color-based sorting tasks, MasterPi harnesses the power of OpenCV and Python codes, ensuring precise recognition and sorting capabilities. To round off its skill set, MasterPi leverages OpenCV for region of interest (ROI) extraction and adept image processing, enabling seamless and efficient line following operations. The capability of visual optics will ensure accurate target tracking, color sorting, and line following.

7.2. Obstacle Avoidance

MasterPi integrates an ingenious obstacle avoidance system, bolstered by a glowy ultrasonic sensor. This sensor acts as a vigilant sentinel, detecting obstacles and triggering adjustments in light colors for heightened situational awareness. Importantly, this system maintains peak performance efficiency

through its low CPU consumption, ensuring smooth and uninterrupted operation even in complex environments.

8. Bill of material



Total Price	Provider
$329.99	Hiwonder.com

Figure 10: Bill of material and package set

Upon acquiring the robot kit inclusive of all components depicted in Figure 9, the assembly process typically spans approximately three to four hours. Subsequently, an additional two hours are devoted to coding and configuring the servos. While the robot's instructions are documented, a significant hurdle emerged due to the majority of documentation being in Chinese, necessitating further translation efforts for technical functionality. Moreover, the foundational coding libraries are primarily programmed in Chinese, posing challenges in comprehending and implementing them effectively, thus adding more time to the implementation process.

9. Results and discussion

These figures and tables below contribute to showcasing the diverse aspects of the project, including control interfaces, servo movements, programming snippets, image processing settings, and the execution of major functions, providing comprehensive insights into the project's technical aspects.

9.1. Software and set up

HiWonder software interface (documentation) for controlling various actions of the robotic arm, providing an overview of the control layout and functionalities.

Figure 11: Robotic Arm actions control screen

9.2. Robotic Arm actions

Here, the data related to the servo movements for the 'head shaking' action is presented, showcasing the servo behavior during this specific motion.

	Index	Time	ID:1	ID:3	ID:4	ID:5	ID:6
	1	2000	1500	590	2500	700	1500
	2	800	1500	590	2500	700	550
▸	3	100	1500	590	2500	700	550
	4	1000	1500	590	2500	700	2230
	5	100	1500	590	2500	700	2230
	6	1000	1500	590	2500	700	550
	7	100	1500	590	2500	700	550
	8	1000	1500	590	2500	700	2230
	9	500	1500	590	2500	700	1500

Figure 12: Head shaking action servo data

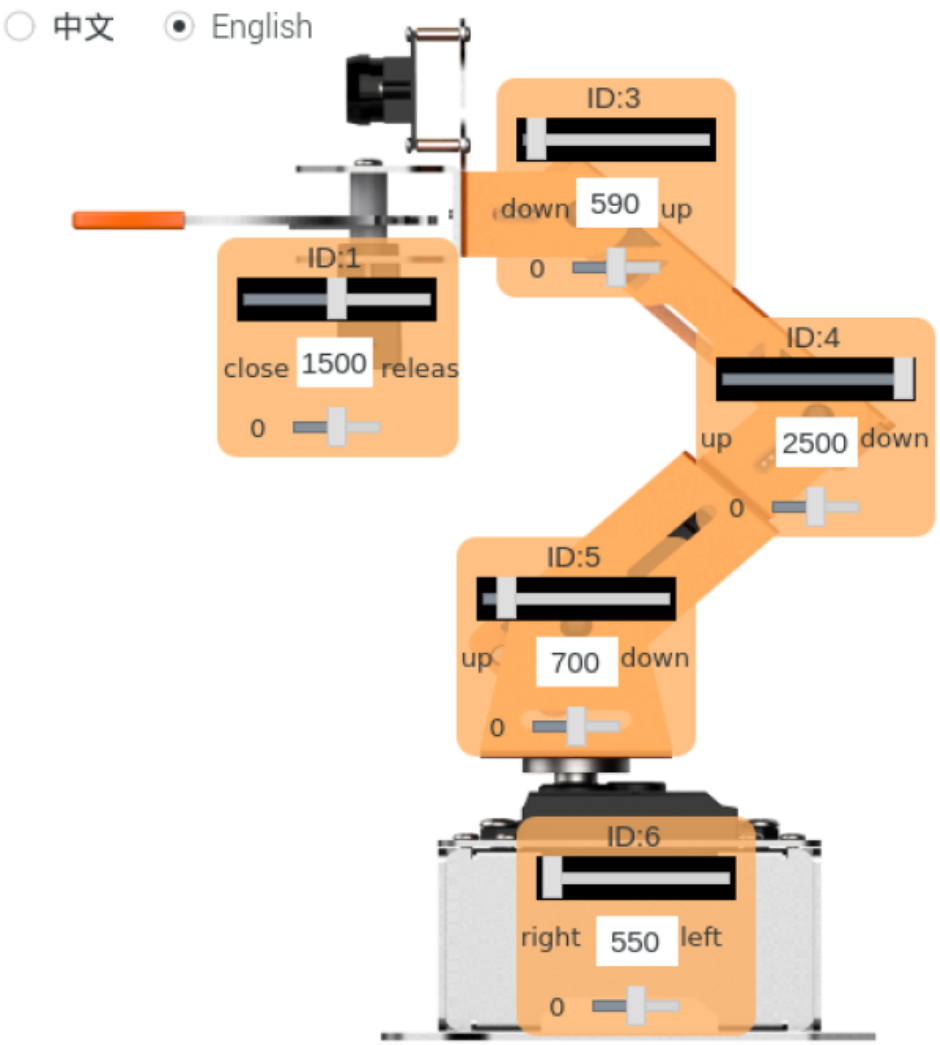


Figure 13: Head shaking action servo orientation

The figure above, illustrates the orientation specifics of the servo involved in the 'head shaking' action, providing insights into its positioning and movement pattern.

	Index	Time	ID:1	ID:3	ID:4	ID:5	ID:6
	1	2000	1500	590	2500	700	1500
	2	200	1500	500	2500	700	1500
	3	200	1500	800	2500	700	1500
	4	200	1500	500	2500	700	1500
▸	5	200	1500	800	2500	700	1500
	6	200	1500	590	2500	700	1500

Figure 14: waving gesture action servo dat

Moreover, the above data regarding servo movements for the 'waving gesture' action is depicted here, offering a detailed look at the servo behavior during this action.

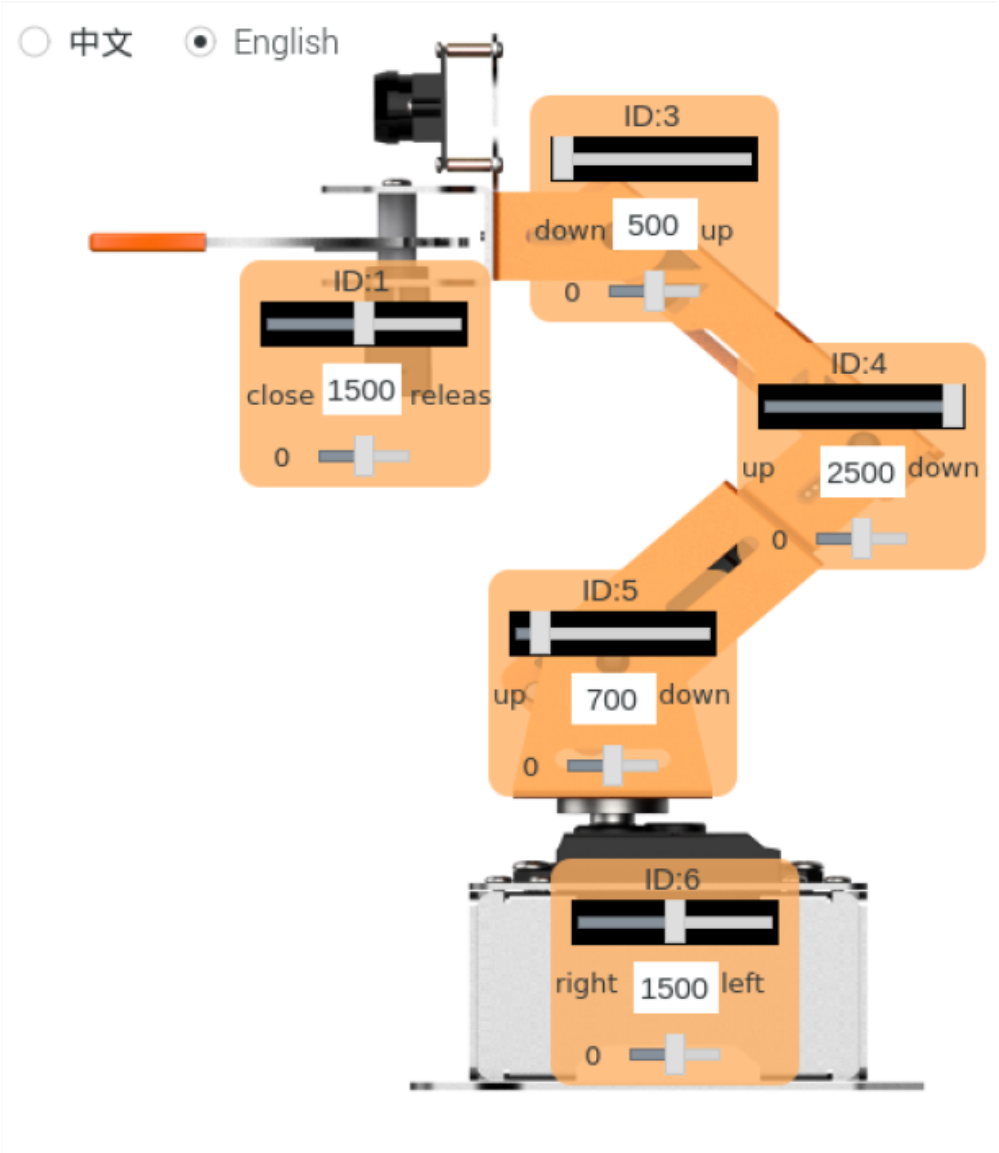


Figure 15: waving gesture action servo orientation

The orientation specifics of the servo engaged in the 'waving gesture' action, detailing its orientation and movement pattern.

9.3. Programming and Functions

Below is a snippet of Python code showcasing major functions involved in controlling the robot's movements, providing an insight into the programming structure.

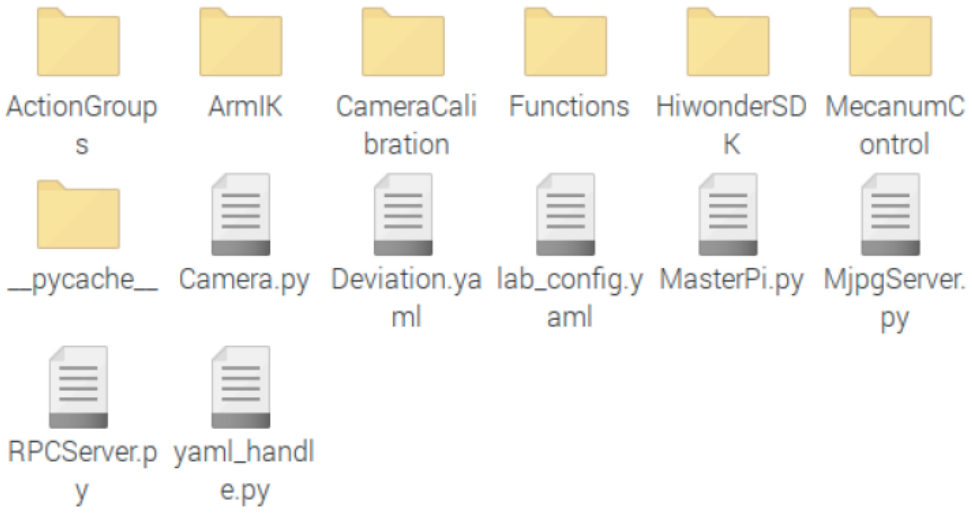


Figure 16: Python coding directory for major functions

```
#!/usr/bin/python3
# coding=utf8
import sys
sys.path.append('/home/pi/MasterPi/')
import time
import signal
import HiwonderSDK.mecanum as mecanum

if sys.version_info.major == 2:
    print('Please run this program with python3!')
    sys.exit(0)

'')

chassis = mecanum.MecanumChassis()

start = True
#Pre-closing processing
def Stop(signum, frame):
    global start

    start = False
    print('Closing...')
    chassis.set_velocity(0,0,0) #Stop all motors


signal.signal(signal.SIGINT, Stop)

if __name__ == '__main__':
    while start:
        chassis.set_velocity(50,90,0)
```

```
    time.sleep(1)
    chassis.set_velocity(50,0,0)
    time.sleep(1)
    chassis.set_velocity(50,270,0)
    time.sleep(1)
    chassis.set_velocity(50,180,0)
    time.sleep(1)
chassis.set_velocity(0,0,0)  #Stop all motors
print(Closed)
```

9.4. Image Processing

Figure 17: Color sorting settings for blue

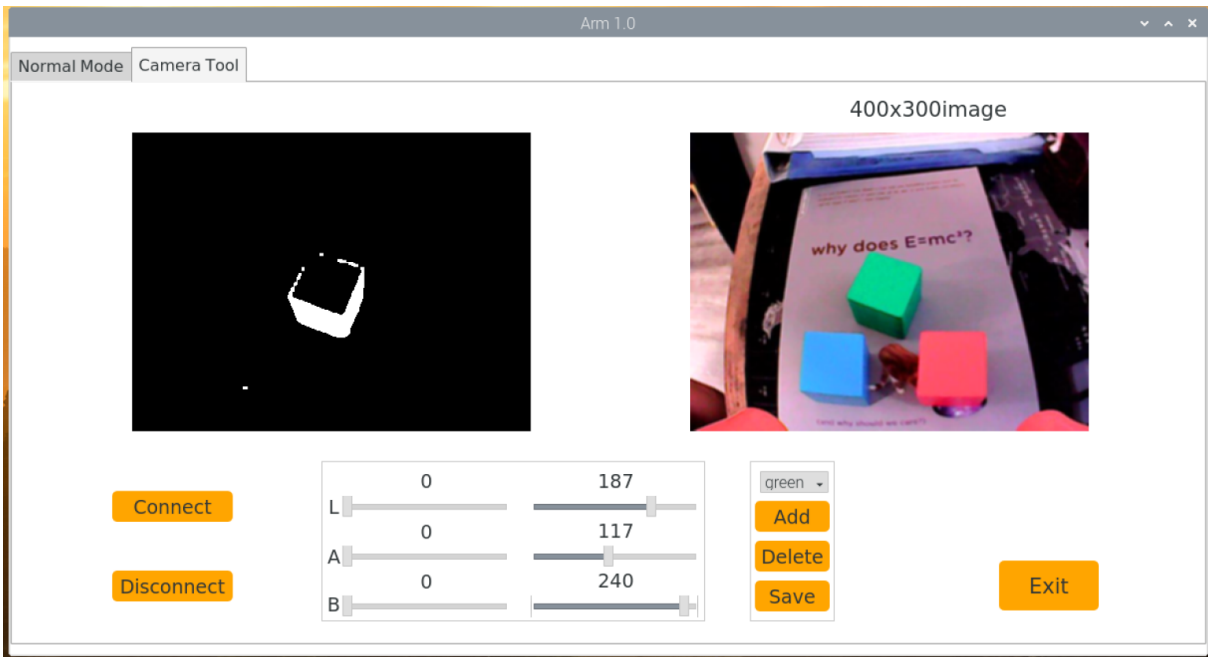


Figure 18: Color sorting settings for green

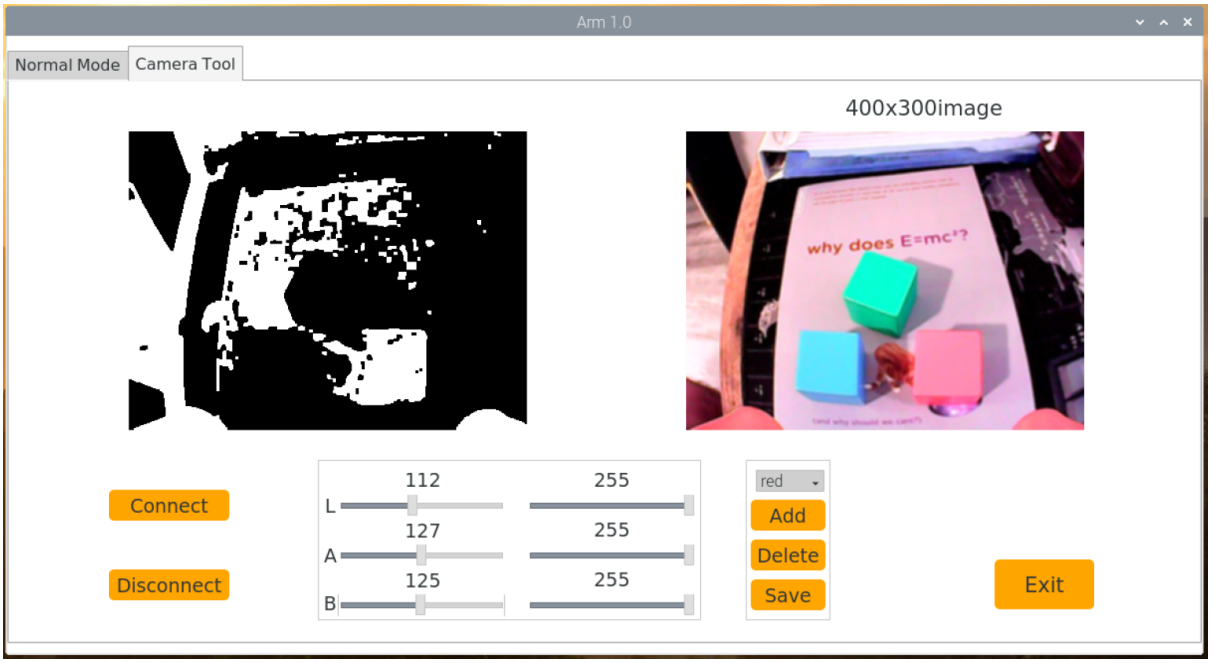


Figure 19: Color sorting settings for red

Figure 20: Color sorting settings for white

Color sorting settings for various color spectrums— 'blue' *Figure 17*, 'red' *Figure 19*, and 'white' *Figure 20*—are individually outlined, delineating the specific configurations employed for sorting tasks related to each color. Additionally, *Table 2* summarizes the settings for multiple coded colors, furnishing precise ranges across different color channels crucial for facilitating color-based sorting operations. Moreover, *Figure 21* displays the calibration image essential for resetting color configurations, serving as a reference point for calibration settings.

Table 2: Color sorting settings for coded colors - Summary table

Lightness (L-Channel)	**Green-red (A-Channel)**	**Blue-yellow (B-Channel)**
Black		
Min: 0, Max: 115	Min: 0, Max: 135	Min: 0, Max: 135
Blue		
Min: 26, Max: 255	Min: 43, Max: 255	Min: 22, Max: 115
Green		
Min: 0, Max: 200	Min: 0, Max: 120	Min: 0, Max: 150
Red		
Min: 0, Max: 255	Min: 145, Max: 255	Min: 130, Max: 255

White		
Min: 193, Max: 255	Min: 0, Max: 255	Min: 0, Max: 255

Moving to the execution phase, *Figures 22* and *23* showcase the screen executions of major functions and line-following processes using Python, respectively. These visual representations offer insights into the actual execution processes, providing a comprehensive understanding of the practical implementation of significant functions based on color recognition.

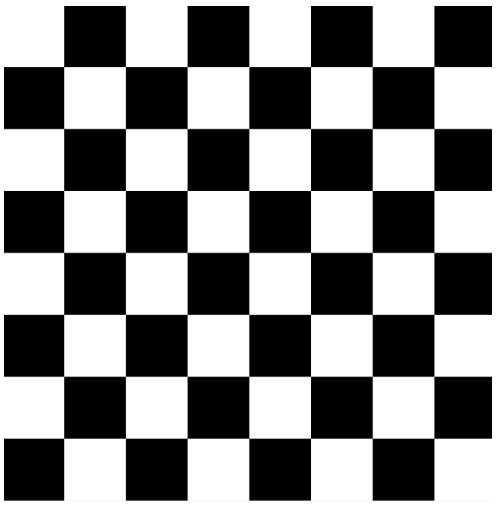


Figure 21: Calibration image for color setting reset

9.5. Performing functions

To execute the functions of the robot effectively, the Python prompt command serves as a pivotal tool. In order to access and activate these functions, the user navigates to the specific directory 'MasterPi/Functions/' using the command line. Once inside this directory, the command 'sudo python3 Avoidance.py' is inputted into the Python prompt. This command initiates the execution of the 'Avoidance.py' script, which triggers and operates the avoidance functionalities of the robot. Through this sequence of commands, users can precisely control and activate the desired functionalities essential for the robot's avoidance mechanism.

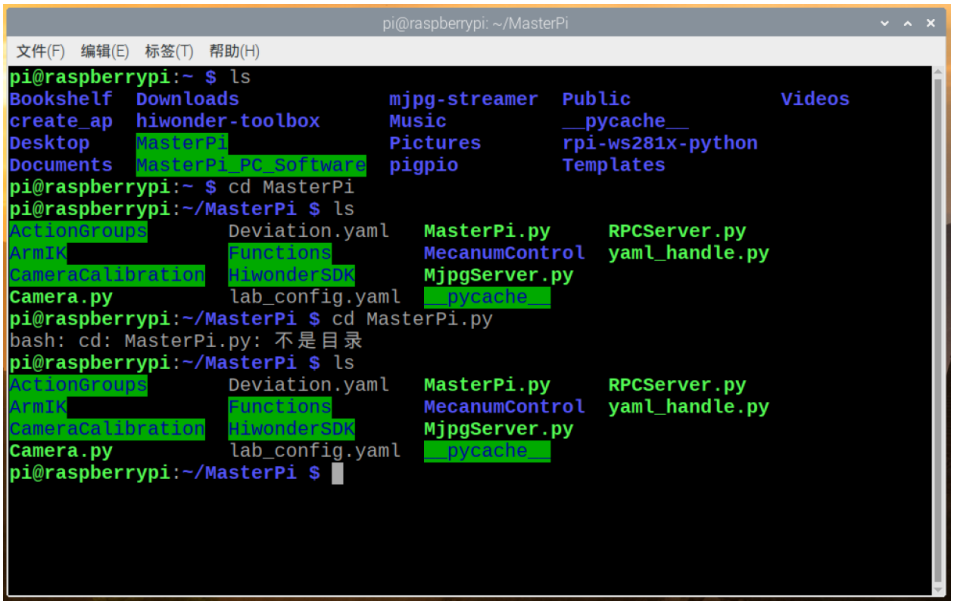


Figure 22: Execution screen for major functions using Python

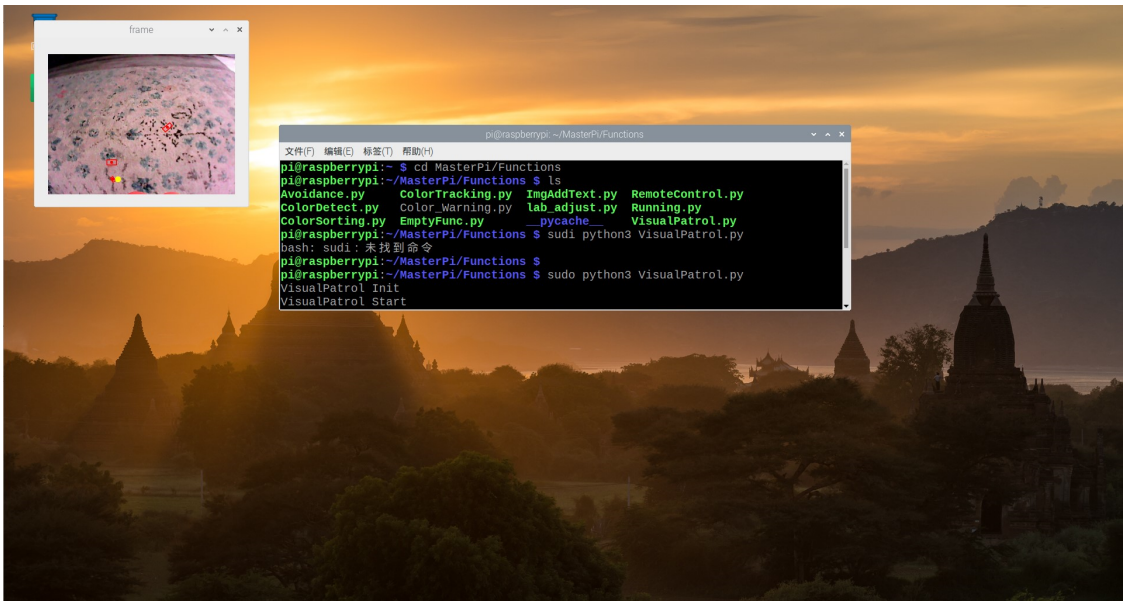


Figure 23: Execution screen for line following based on color

10. Presentation and Demonstration

In this section, Canvas is utilized as the primary platform for presenting the project findings, enabling the clear and concise display of data, figures, and analyses (Link to presentation). Additionally, VNC (Virtual Network Computing) serves as the access protocol to connect to the Raspberry Pi of the MasterPi robot access point. This method facilitates direct LAN (Local Area Network) connection, enabling seamless access and control of the robot's Raspberry Pi, thereby enhancing the efficiency of real-time monitoring, management, and potential modifications of the robot's

functionalities. The demonstration conducted in class included line flowing, color detection, and obstacles avoidance (Link to demtration visuals). Based on the testing, it was noticed that the robot can get confused when being placed in an area with vlockings from both left and right sides. To solve this issue the python code for the movement function needs to be modified so it perform different direction if the robot is tuck more than one seconds:

```
def move_forward():
    motor1.forward()
    motor2.forward()

def move_backward():
    motor1.backward()
    motor2.backward()

def turn_left():
    motor1.backward()
    motor2.forward()

def turn_right():
    motor1.forward()
    motor2.backward()

def stop_motors():
    motor1.stop()
    motor2.stop()

def main():
    stuck_time = 0
    stuck_threshold = 2  # Stuck threshold in seconds
    directions = [move_forward, move_backward, turn_left, turn_right]

    while True:
        # Check distance measured by the ultrasonic sensor
        distance = ultrasonic_sensor.distance * 100  # Convert to centimeters

        if distance < 10:  # Change this threshold based on your requirement
            move_backward()
            stuck_time += 0.1  # Increment the time the car is stuck
        else:
            stop_motors()
            stuck_time = 0  # Reset the stuck time if not stuck

        # Check if the car is stuck for more than 2 seconds
        if stuck_time >= stuck_threshold:
            # Perform different maneuvers randomly
            for _ in range(5):  # Try 5 different directions
```

```
            random_direction = random.choice(directions)
            random_direction()
            sleep(1)  # Move for 1 second
            stop_motors()  # Stop after trying the direction
            sleep(0.5)  # Pause between different directions

        stop_motors()  # Stop after trying different directions
        stuck_time = 0  # Reset the stuck time

    sleep(0.1)  # Adjust the delay based on your requirements

if __name__ == "__main__":
    main()
```

11. Conclusion and Future Prospects

This project's innovative capabilities offer a solution to the pressing issue of removing defunct satellites from lower Earth orbit. By integrating cutting-edge camera-based analysis and precise obstacle avoidance, the MasterPi exemplifies the potential for a paradigm shift in our approach to space sustainability. Looking forward, enhancing this project by evolving it into a robotic drone with an advanced arm could further amplify its impact. This transformation would enable even greater mobility and versatility, opening up new avenues for efficient satellite collection and debris management. With continuous innovation and strategic adaptations, the MasterPi project has the potential to redefine the landscape of space technology and contribute significantly to a more secure celestial environment.

12. References

Godec, P., Pančur, M., Ilenič, N., Čopar, A., Stražar, M., Erjavec, A., Pretnar, A., Demšar, J., Starič, A., Toplak, M., Žagar, L., Hartman, J., Wang, H., Bellazzi, R., Petrovič, U., Garagna, S., Zuccotti, M., Park, D., Shaulsky, G., & Zupan, B. (2019). Democratized image analytics by visual programming through integration of deep models and small-scale machine learning. *Nature Communications*, *10*(1). https://doi.org/10.1038/s41467-019-12397-x

Ljubljana, B. L., University of. (n.d.). Widget catalog. Orangedatamining.com. https://orangedatamining.com/widget-catalog/

Ivanov, S., Tsokov, S., Ivanov, T., Zlateva, V., Avramov, M., Kanev, T., Ivanov, B., Neshev, N., Vassilev, V., & Dankov, P. (n.d.). Space Debris Identification, Classification and Aggregation with Optimized Satellite Swarms. Retrieved December 5, 2023, from https://www.spacemic.net/pdf/mic4/finalist/08.StoilIvanov_abst.pdf

Marek Perkowski. (n.d.). Intelligence Robotics Class Lectures. Web.cecs.pdx.edu. Retrieved November 23, 2023, from https://web.cecs.pdx.edu/~mperkows/

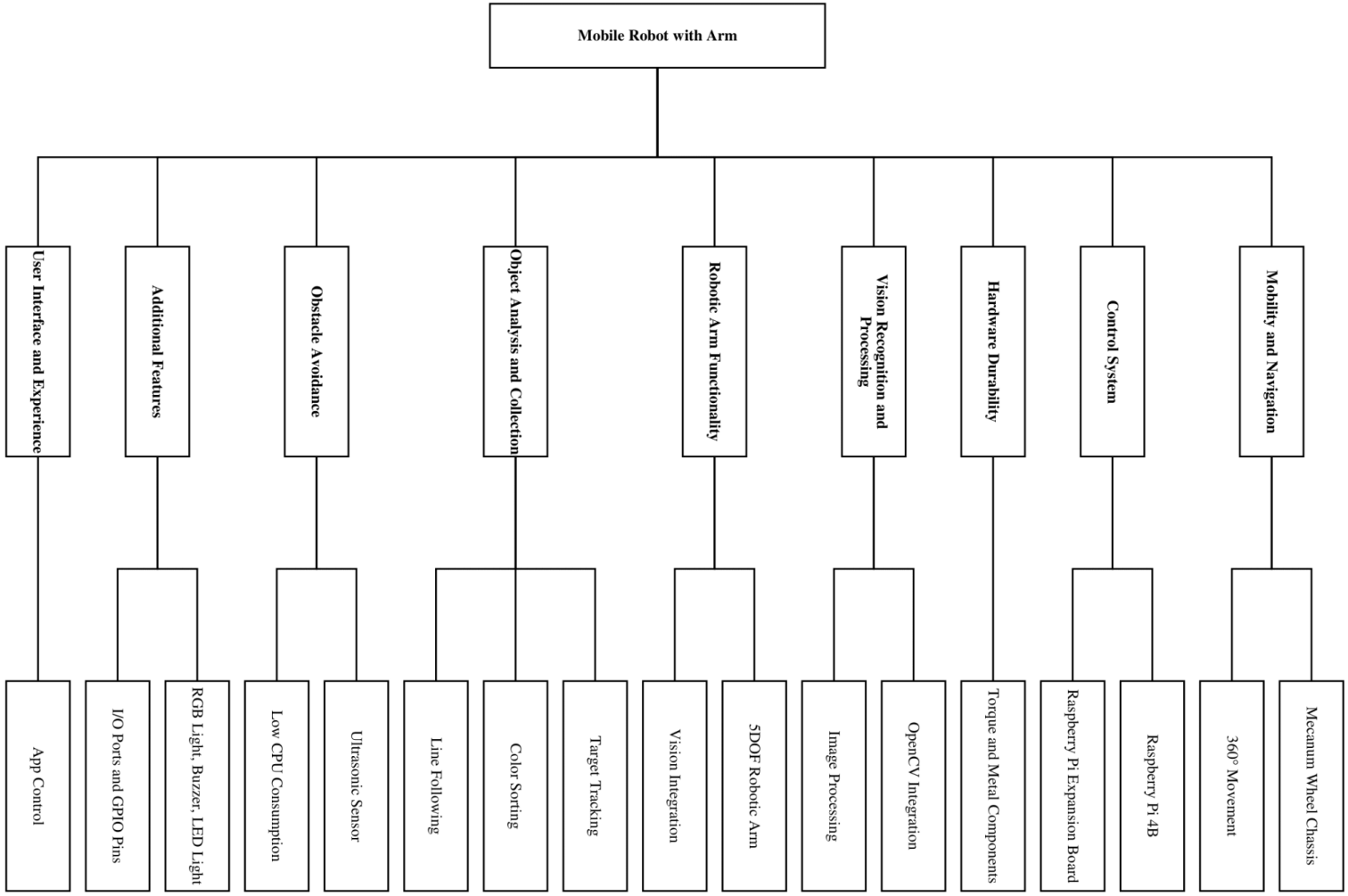
MasterPi - Google Drive. (n.d.). Drive.google.com. Retrieved December 5, 2023, from https://drive.google.com/drive/folders/19wOOF4T_N37y_SSklHbo7DnL76_P_NWb?usp=sharing

Python Image Processing Cookbook. (n.d.). Subscription.packtpub.com. Retrieved December 4, 2023, from https://subscription.packtpub.com/book/data/9781789537147/1/ch01lvl1sec03/transforming-color-space-rgb-lab#:~:text=The%20CIELAB%20(abbreviated%20as%20Lab

Test & Score — Orange Visual Programming 3 documentation. (n.d.).
Orange3.Readthedocs.io.
https://orange3.readthedocs.io/en/3.5.0/widgets/evaluation/testandscore.html

Zupan, Demsar: Introduction to Data Mining. (2018).
https://file.biolab.si/notes/2018-05-intro-to-datamining-notes.pdf

Appendix .A | Tentative Functional Decomposition



Appendix .B Orange Hierarchical Clustering