Domestic Helper Bot

A Design Project Report

Present to the School of Electrical and Computer Engineering of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Honglei Huo, Zhiming Xie

MEng Field Advisor: Joseph F. Skovira

Degree Date: January 2023

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: Domestic Helper Bot

Author: Honglei Huo, Zhiming Xie

Abstract: Due to COVID-19, some workers started working from home and spent more time on home renovation. Therefore, our team decided to build the Domestic Helper Bot to serve as an assistant to many homeowners. The goal of DHB is to perform robot-to-human handover in a household. DHB consists of a Roomba base mounted with a robotic arm with a camera, another camera mounted on the Roomba Robot, and a USB microphone. The system will run on Python 3. The robot will deliver tools that are requested by the owner through voice commands; recognize its owner's voice; locate the tool location; pick up the tool with its robotic arm and then deliver the tool to the owner. The project could offer a lot of benefits to make life easier.

Introduction

This project has three main parts: Vision, robotic arm manipulation, and Voice Recognition. Our team uses the SpeechRecognition package to determine a user's command. The user will send the signal via a USB microphone, and the microphone will record the voice that will go through the speech recognition algorithm to determine which tool is requested. After understanding the request, the mobile robot will move to the toolbox that is given as a waypoint. By using the computer vision algorithm, when the robot gets close enough to the toolbox. It will pause and start to use the camera which is on the robotic arm to look for the tool using OpenCV. We use color algorithms primarily in conjunction with the inverse kinematic of the robotic arm. The advantage of implementing color recognition is to achieve instant recognition without Neural Networks training, which consumes a lot higher computational power.

System Requirements

A. Roomba Assembly

Roomba Create2 is the base platform for the entire system as shown in Figure 1. Additionally, three laser-cut acrylic layers are used to build the structure for carrying different parts of the system. For top layer is used to support the robotic arm and camera. The second layer is to carry the power system. The bottom layer is to fix other layers without damaging the Roomba.

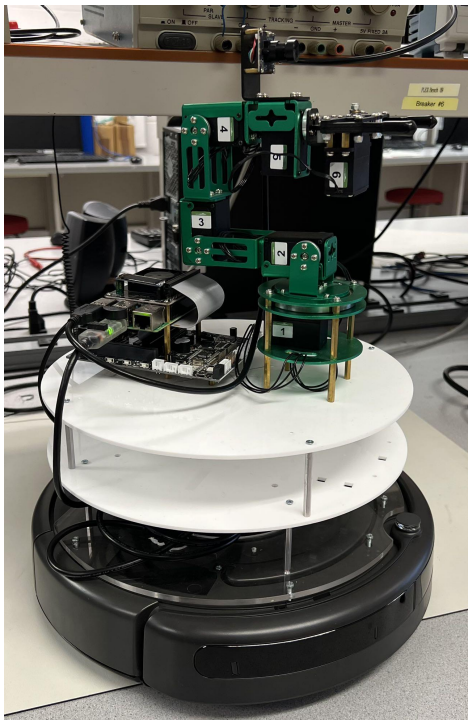


Figure1. Roomba Assembly

B. Device Connections

The Raspberry Pi 4B and expansion board is mounted on the acrylic platform, and connected by using a Ribbon cable to support power and data transmission. The robotic arm is also connected to the expansion board by wires. Therefore, it is only necessary to connect the expansion board to the power supply to drive the entire robot arm system, and there is no need to provide additional energy equipment for the Raspberry Pi and the robot arm. The camera on the robotic arm, additional camera, and a microphone for voice recognition are all connected to the Raspberry Pi via USB ports of the Raspberry Pi.

Achievements and Current Progress

1. Computer Vision Algorithm

The computer vision algorithm is the core control algorithm of the entire robot project. We use the functions from OpenCV to select a specific color, and then combine it with the robot's control algorithm so that Roomba can find and move to the position of the toolbox, and the robot arm can recognize and pick up corresponding items.
The core idea of this algorithm is to first convert the picture captured by the camera from the BGR color space to the HSV color space, so that the computer can process the color more accurately. Next, find the part of the image between the upper and lower thresholds of the specified color (that is, the part we need). The third step is to erode the converted image to remove excess noise and avoid the algorithm being affected by noise. To erode, we choose 3x3 kernels and 2 iterations. Because these two parameters can reduce the noise without causing major changes to the image.

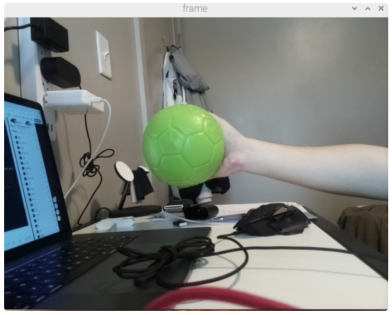
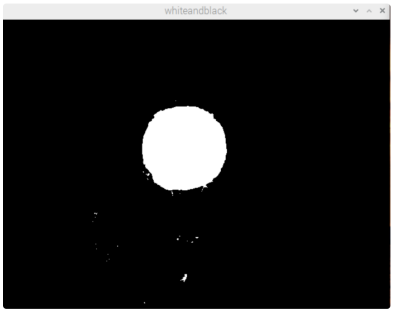
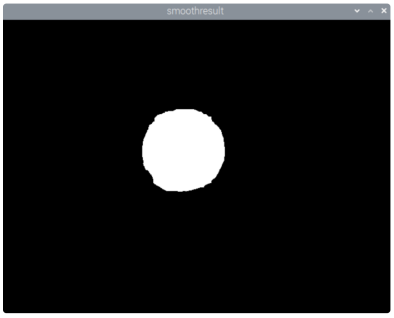
Frame captured by camera	Image only remain the green object	Image after erode
		

Table 1. Image Processing

2. Robotic Arm Control Algorithm

For the control part of the robot arm, we use the output of the computer vision algorithm as the input of the robot arm control algorithm, and divide the picture captured by the camera into three regions.

As Figure 1 shown, the size of the frame is 640x480 pixels, since we need to make sure the center of the target object is in the middle of the camera screen, so as to ensure that the grabbing direction of the robotic arm and the target object are on the same straight line. Because the Raspberry Pi and the operating system we use cannot achieve the performance of a real-time operating system, we cannot make the robotic arm be perfectly aligned with the centerline of the object. According to the robotic arm API Arm_serial_servo_write(id, angle, time) , in order to avoid the robotic arm losing target, we set the time as 1000, which is the running time of the servo. within the effective range, the servo rotates at the same angle. The smaller the input running time, the faster the steering gear moves. Additionally, we set the angle of each rotation of the servo is 0.5 degrees and 0.5 second sleep time between each rotation. We also draw a range at a distance of 6 pixels from both sides of the centerline of the screen. If the center of the target object appears to the left of Xmin or the right of Xmax, the control algorithm will use the servo to rotate the arm horizontally until the center of the object is between Xmin and Xmax. Additionally, in order to avoid mistakes, the algorithm will only be executed correctly if the selected color area is greater than the preset threshold, and any results smaller than this threshold will be ignored. In this case, the orientation of the robotic arm is guaranteed to be correct and the grasping operation can begin.

During our development of the robotic arm control algorithm, the main aim is to guarantee the success rate, only trying to slow down the moving speed of the arm. As a part of future work, a fixed input for FPS could be determined, and also reduce the running time. Use the combination of these two parameters to optimize the operating speed of the robotic arm on the basis of ensuring the success rate.

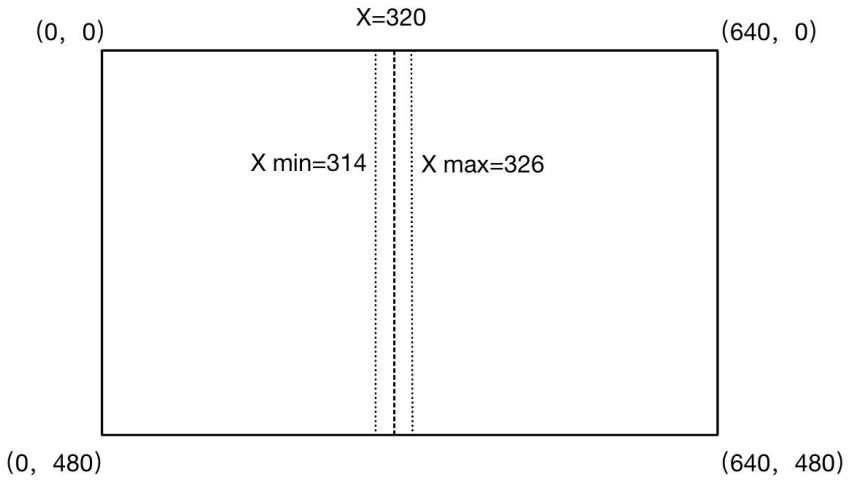


Figure 2. Schematic diagram of the logic of the robotic arm control algorithm

3. Roomba Control Algorithm

camThread class is built and implemented to allow the raspberry pi to control two cameras in different channels. Port 0 connects with the robotic arm camera and port 2 connects with the Roomba camera which is mounted on the top front platform. The ports are interchangeable, and the algorithm can activate a specific camera when the command is passed through from the terminal/voice command. The Roomba camera uses the same algorithm as the robotics arm algorithm but with an addition of x bias value of 100. The purpose of inserting the bias shown in Figure 3 is to enlarge the center of mass region from xmin to xmax to (xmin - xbias) to(xmax + xbias). With a 100 pixel wider center of mass searching region, the Roomba camera can locate the x-center of the toolbox more conveniently, and the Roomba would not need to make unnecessary small-angle left and right shifting to align itself to the toolbox center. This x bias value is obtained from the experimental result when a red shoebox was used as a toolbox. The shoebox size is approximately 10 x 5 square inches.

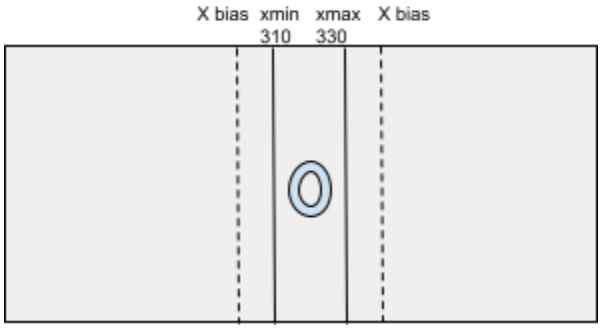


Figure 3: Toolbox drawing with xmin, xmax, and bias

Having the bias can avoid adjusting the Roomba toward the left or right of the toolbox center frequently. As a result, the Roomba can seemingly travel in a straight line when the Roomba camera sees the toolbox from a distance. In the experiment, we placed the Roomba 3 to 4 feet away from the toolbox. (Figure 4)

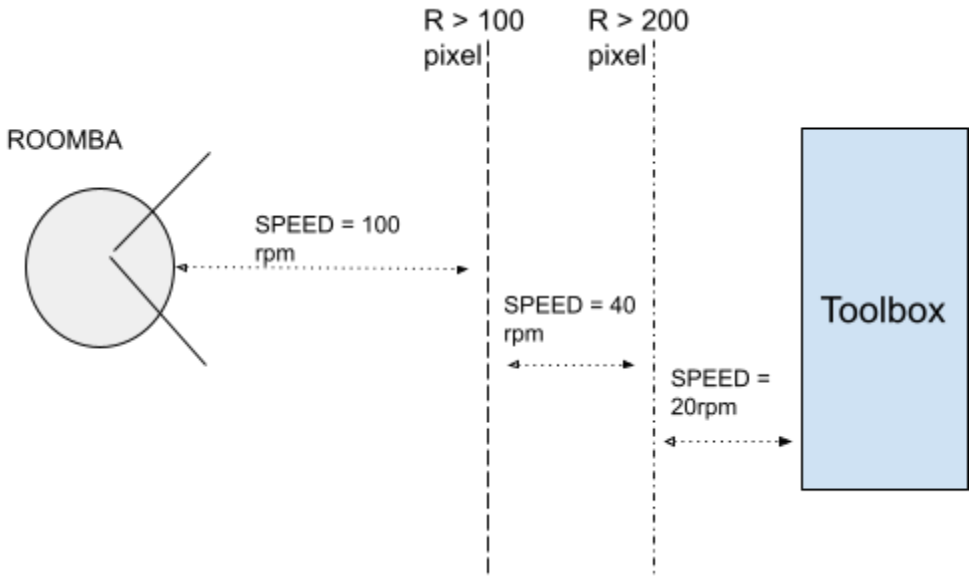


Figure 4: Roomba Traveling Velocity Graph

In addition to adding the bias value, a value of 20 is added to the left wheel velocity of the Roomba to balance the uncoordinated speed between the left and right wheels.(Figure 4). This value is a fixed value and it is only applicable to the Roomba we are given for this project. Other values for tuning the drifting behavior of other Roomba robots should be obtained from experiments. To have the Roomba stop right at the toolbox, we include two safety features in the algorithm. First of all, when the Roomba camera is activated, it will obtain the real-time value of the toolbox's radius, given by the function cv2.minEnclosingCircle(area). (Figure 3) When the radius is in the range between 20 pixels and 100 pixels, the Roomba travels with a speed of 100 rev/min, and when the radius is larger than 100 pixels, the Roomba will slow down to 40 rev/min, and when the radius is larger than 200 pixels, the Roomba will travel at a speed to 15 rev/min for 2 seconds and stop in front of the toolbox. The traveling time of 2 seconds is an arbitrary value, and the original plan and code were to have the Roomba stop 2 centimeters away from the toolbox when the detected toolbox radius is larger than 350 pixels. However, due to the time delay in obtaining the actual radius of the toolbox, we altered the code to have the Roomba run for 2 seconds after detecting the toolbox radius that is over 200 pixels. Thus, we can make sure the stopping distance is about 2 centimeters away. This distance is considered the optimal distance for the robotic arm’s moving radius. When the Roomba has arrived at the toolbox, the algorithm will switch the camera port to activate the robotics arm camera and commands the arm to recognize the target object and obtain the object. The target object was set in a fixed location on the toolbox for the current stage. Further improvements on the robotic arm algorithm and Roomba stopping algorithm will be discussed in the future work session.

4. Voice Recognition

SpeechRecognition3.9.0 - Google Cloud Speech API is implemented in the algorithm to recognize basic commands such as toolbox and color names. Each tool corresponds to a unique color, and the user only needs to input the color name, and the robot will grab the correct item. The user will need to speak directly to a USB microphone that is connected to the raspberry pi. The voice command will serve as an input source(audio) and will be passed to the function r.recognize_google(audio). After the voice is recognized and outputted as a word, the algorithm will check if such a word exists in the pre-defined vocab library, which is an array of valid commands.(Figure 5) If the word matches one of the strings in the array, then the algorithm will pass the word as input to the corresponding function and activate the robot arm or Roomba to perform certain actions. If the word does not appear in the given vocab library, then the user will be asked again to input the correct message until the command matches a string in the library.

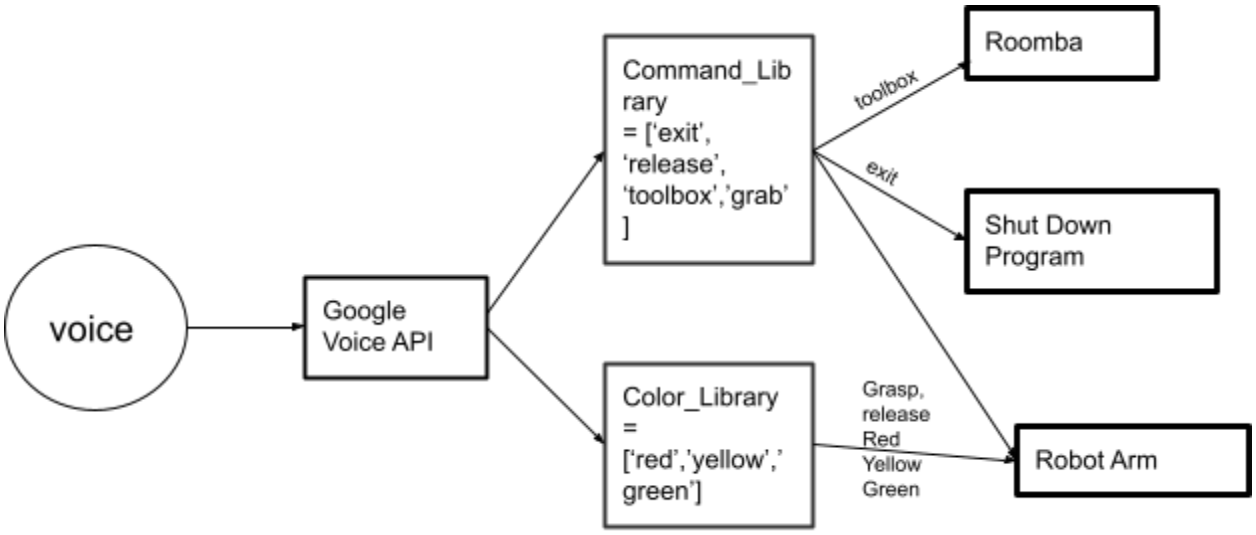


Figure 5: Voice Recognition Flowchart

5. Main Program Logic

The main flow of logic is shown in figure 6. Refer to figure 6, the user is first prompted to give commands to the raspberry pi. If the commands are recognized by Google Voice API, then the words will be passed to the command library and the color library. Assuming the words are “toolbox” and “yellow”. The algorithm will first check if “toolbox” is in the command library. If so, it will activate the Roomba and its front camera, and drive the Roomba to the toolbox. When the Roomba reaches 2 centimeters away from the toolbox, the program will pass the second word “yellow” to the color library. If the second word is in the color library, the algorithm will activate

the robotic arm and command it to obtain the target object. After obtaining the object, Roomba will return the object to the user. The returning part is labeled in dotted line because we have not completed the algorithm of that part, and we will elaborate the thought process in the future work section.

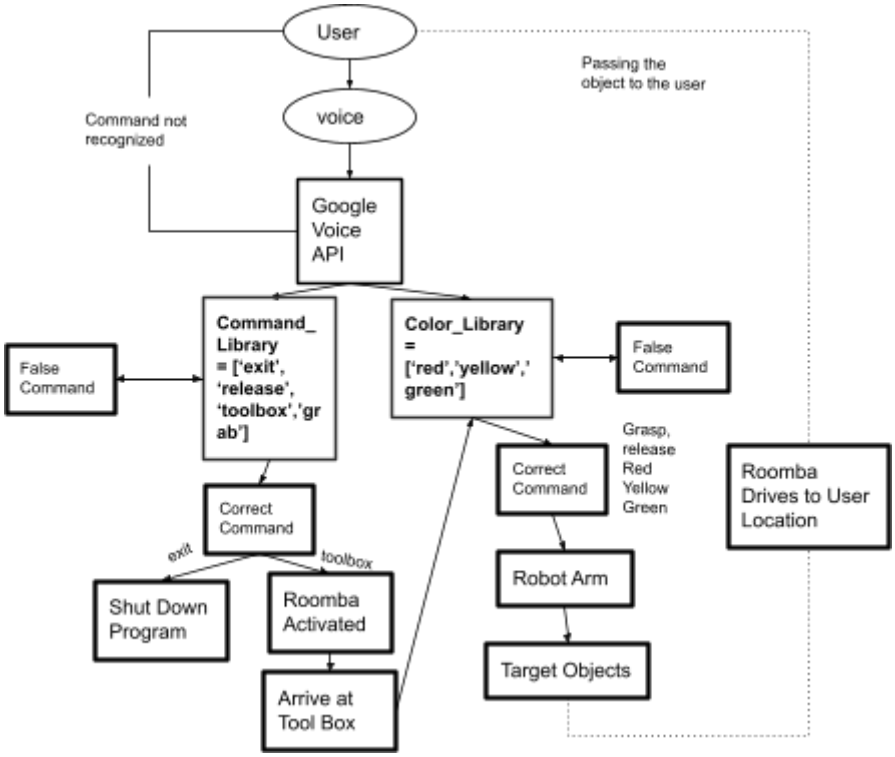


Figure 6: Flow Chart of Overall Project

Results

The results of the project are basically the same as our expectations, the robot can travel to the target location correctly and slow down at the appropriate location to avoid always hitting the toolbox. The robotic arm can also start to recognize the target object at the right time and rotate to the correct angle. However, there are also some shortcomings. In a very quiet environment, the success rate is 80%, however if the lab is crowded, the success rate will drop to 30% with long processing time. So during the following tests, we use the keyboard to input the commands. The success rate for driving to the expected location near the toolbox is 37.5%. If the robot reaches the correct location, the success rate of recognizing the block is 100%. Once the block is recognized, the success rate of picking up is 90%. The return process is not finished, it would be a part of future work.

Future Work

1. Computer Vision Algorithm
   Since we are using colors currently, we have to prepare many different colored blocks to represent different tools. When we developed the CV algorithm, we also did some tests on using machine learning to recognize different tools based on Tensorflow lite and YOLO.

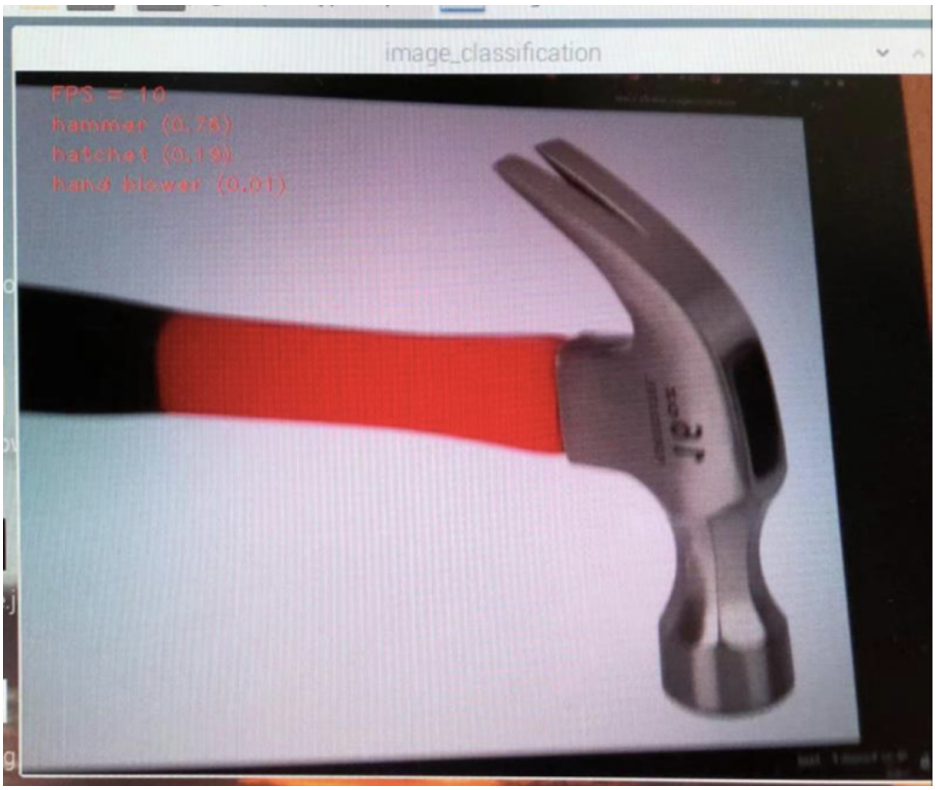


Figure 7.Identify the hammer shown on the screen

We also explored another way to identify different tools, which is using Apriltags.

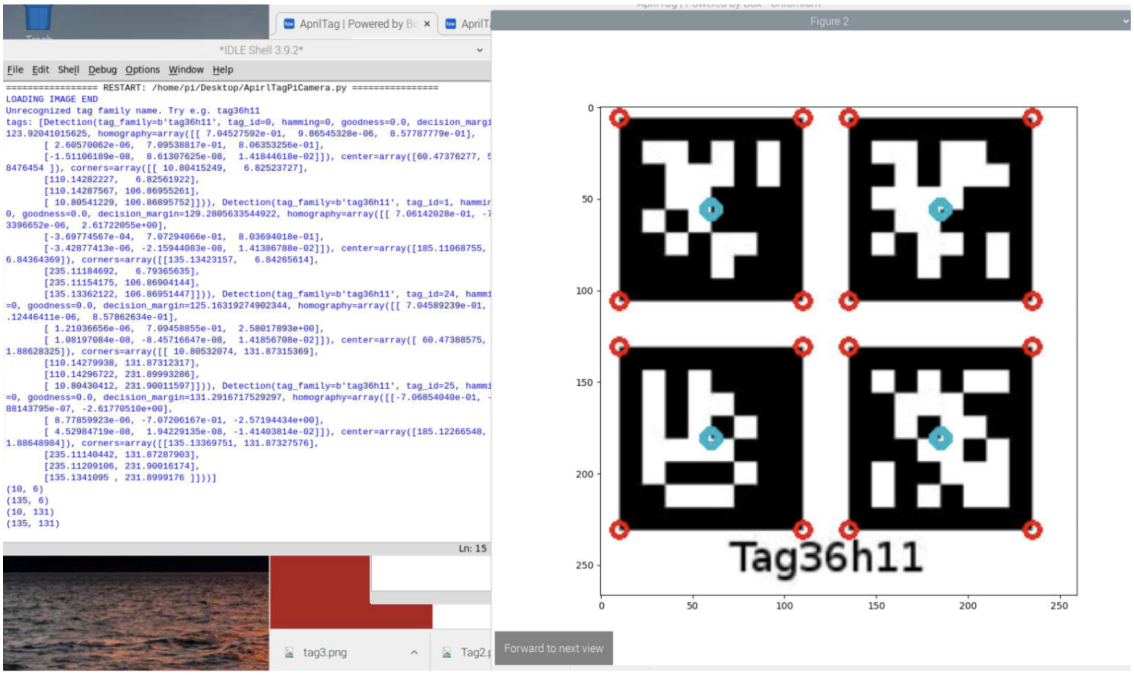


Figure 8.Identify different Apriltags in the same family

After our tests, these two methods are achievable, but we prefer the Apriltag method to be developed in the future, since the correct rate of recognizing a specific Apriltag is 100%, and the processing time is much less than recognized by Tensorflow lite.

2. Robotic Arm Control Algorithm

   At present, the robotic arm can correctly identify the direction of the target object and rotate to align with the center of the target. However, it can only grasp objects at a relatively fixed distance. If the distance between the object and the robotic arm is too far or too close, the robotic arm will fail to grasp.

   For this part, the solution we thought of was to install a top-down camera above the toolbox. The camera can transmit the position of the recognized tool back to the Raspberry Pi through the wireless network. The target object can be successfully grasped by the inverse kinematics robotic arm, which is no longer limited by the distance of the object.

3. Voice Recognition

   Our speech recognition function currently has two major problems, one is slow processing speed, and the other is that it is easily affected by environmental noise. We think ambient noise may also be an important reason for the slow processing speed. The method we think of is that the continuous noise in the environment can be reduced through the algorithm,such as a filter, and the confidence level can be adjusted to judge whether it is noise.

4. Roomba control Algorithm
   The navigation method we currently use is based on computer vision algorithms, but the disadvantage of this navigation method is that once the robot cannot see the target position within the field of vision or encounters obstacles during the travel process, it is difficult for the robot to move correctly to the target position. In future development, there are two potential solutions. One is to further develop the computer vision algorithm so that it can detect obstacles and guide the robot to avoid them, and the other is to use laser radar or distance sensors to avoid direct collisions between robot and obstacles.

5. Power Supply
   All testing at this stage has been done with a power cord connecting the expansion board to wall power. Therefore, the movement area of the robot is greatly limited. Because Roomba has a built-in battery, we only need to find a 12V 5A mobile power supply to connect to the expansion board, which is connected with Raspberry Pi and robotic arm. The expansion board would provide 5V power to Raspberry Pi. So the robot can get rid of the limitation of the length of the power cord.

Individual Works

Honglei Huo:
- Developed computer vision algorithms
- Combining computer vision algorithms and robotic arms for target detection
- Main program logic integration
- Modification and assembly of robot parts.

Zhiming Xie:
- Developed robotic arm control algorithms
- Servos Control Algorithm of Robotic Arm for Target Grasping
- Voice recognition and roomba algorithm
- Main program logic coding
- Use laser cutting to make the structural parts needed for the robot.

Bill of Materials (list of parts to build the system)

A. Hardware:
   Raspberry Pi 4B
   Roomba Create2
   Yahboom Raspberry Pi Robotic Arm
   Webcam
B. Software:
   Raspberry Pi OS (Bullseye 32-bit)
   OpenCV
   Google Voice API

References:

iRobot, “iRobot® Create® 2 Open Interface (OI) Specification based on the iRobot® Roomba® 600”, July 19, 2018

iRobot, “Roomba Create2 Manual”, https://edu.irobot.com/learning-library/getting-started-with-create-2

Yahboom, “ Raspberry Pi AI Robotic arm”, https://github.com/YahboomTechnology/dofbot-Pi

Raspberry Pi, “Operating system images”, https://www.raspberrypi.com/software/operating-systems/

Raspberry, “Starter code for raspberry pi camera”,https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/2

Cherie Tan, “Set up OpenCV on Raspberry Pi 4”, https://littlebirdelectronics.com.au/guides/165/set-up-opencv-on-raspberry-pi-4

Adrian Rosebrock, “AprilTag with Python”, https://pyimagesearch.com/2020/11/02/apriltag-with-python/

Audas, “Speech Recognition With Raspberry Pi”, https://www.youtube.com/watch?v=R1SFP3t7Gwo

J. Skovira, “ECE 5725 Lab1”, Cornell University, Lab Instruction, Spring 2022

J. Skovira, “ECE 5725 Lab3”, Cornell University, Lab Instruction, Spring 2022

A. Chang, C. Chu, C. Qian, K. Lu, R. Peng, Y. Jiao, “I-ROBOT CREATE 2 SYSTEM FOR DEEP UV LIGHT DISINFECTION OF SURFACES” , Cornell University, Design Project Report, May. 2022.

OpenCV, “G-API Image processing functionality”, https://docs.opencv.org/4.x/d2/d00/group__gapi__imgproc.html

Kevin Walchko, “pycreate2 0.8.0”, https://pypi.org/project/pycreate2/

Sam Westby Tech, “How to Install Tensorflow 2 on a Raspberry Pi”, https://www.youtube.com/watch?v=QLZWQlg-Pk0&t=1s

Appendix A:

All codes can be found in this GitHub Repository link.
https://github.com/hongleihuo/Domestic-Helper-Bot