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: In response to the COVID-19 pandemic, many individuals have transitioned to remote work arrangements, resulting in increased time spent on home renovation projects. Consequently, our team has developed the Domestic Helper Bot (DHB) as a versatile assistant to address the needs of modern homeowners. The primary objective of the DHB is to facilitate efficient robot-to-human handover tasks within a domestic environment.

The DHB system comprises a Roomba base, equipped with a sophisticated robotic arm featuring an integrated camera, an additional camera mounted on the Roomba robot, and a USB computer speaker for audio output. The entire system operates on Python 3 programming language. The DHB is designed to respond to voice commands, accurately recognize the owner's voice, identify

the location of the requested tool, retrieve the tool utilizing its robotic arm, and subsequently deliver the tool to the owner in a timely manner. This innovative project holds significant potential to enhance the quality of life by streamlining household tasks and fostering greater convenience for homeowners.

Introduction

The present project integrates three core components—Vision, Inverse Kinematics, and Voice Recognition—to create a seamless user experience. Our team employs the SpeechRecognition package to accurately discern user commands delivered via a USB microphone. Upon interpreting the command, the mobile robot navigates to a predetermined toolbox waypoint. As the robot approaches the vicinity of the toolbox, it momentarily halts and initiates the computer vision algorithm, utilizing the camera integrated into the robotic arm to identify the desired tool. This process primarily involves OpenCV-based color recognition methods in tandem with the inverse kinematics of the robotic arm. The primary advantage of implementing color recognition lies in its capacity for instantaneous tool identification without necessitating extensive Neural Network training, which would entail significantly greater computational demands.

System Requirements

A. Roomba Assembly

As depicted in the accompanying figure, the Roomba Create2 serves as the foundational platform for the entire system. Three additional laser-cut layers have been incorporated to support the various components of the system. The top layer is designed to accommodate the robotic arm and camera, while the second layer houses the power system. The bottom layer functions to secure the other layers in place without causing any damage to the Roomba.

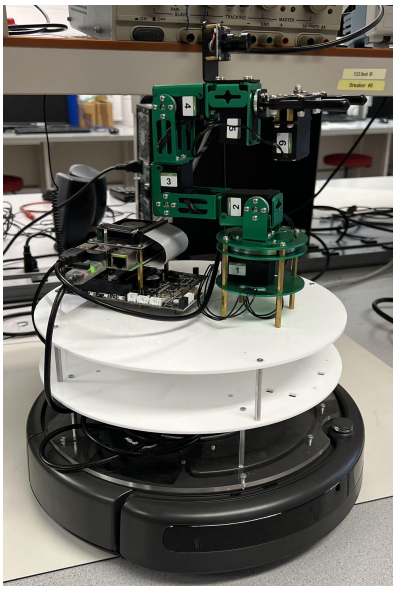


Figure1. Roomba Assembly

B. Device Interconnectivity

The Raspberry Pi 4B is securely mounted onto the expansion board and employs a ribbon cable to enable efficient power and data transmission. The robotic arm is also integrated with the expansion board through a wired connection. Consequently, providing power to the expansion board is adequate for operating the entire robotic arm system, thereby negating the necessity for auxiliary energy resources for the Raspberry Pi and the robotic arm. The camera affixed to the robotic arm, the supplementary camera, and the microphone utilized for voice recognition are all interconnected with the Raspberry Pi via USB ports, ensuring seamless communication and functionality.

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.

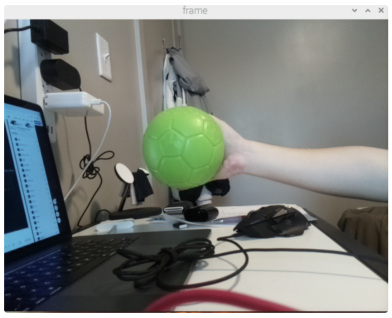
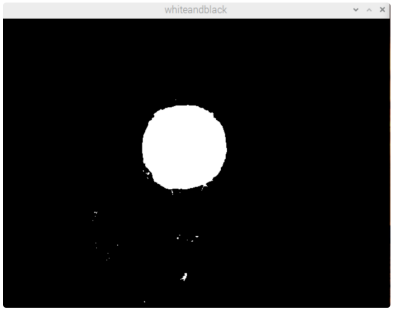
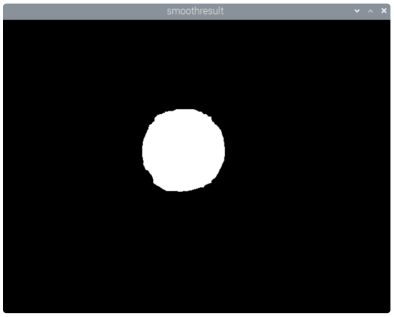
Frame captured by camera	Image only remain the green object	Image after erode
		

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. So we draw a range at a distance of 16 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.

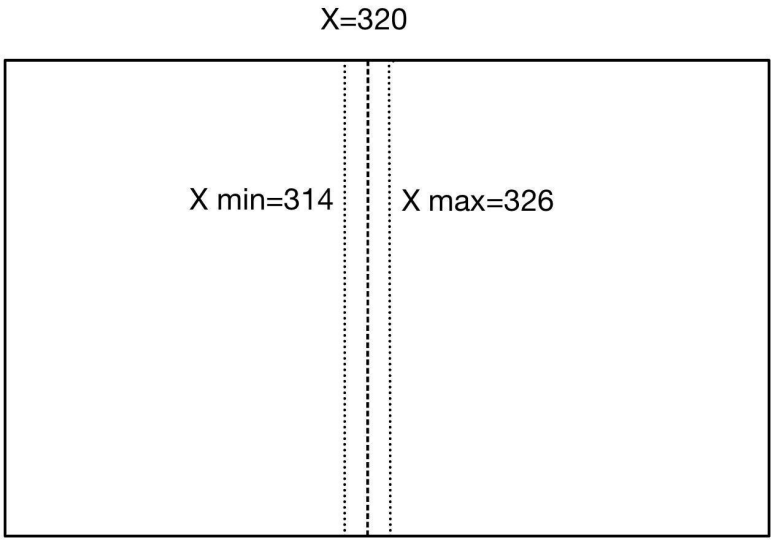


Figure 2. Schematic diagram of the logic of the robotic arm control algorithm

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 a lower x bias, which is 100. This bias value is obtained from the experimental result when a red shoebox was used as a toolbox. By adding this bias value to the current x coordinate value, which is given by the function cv2.minEnclosingCircle(area), the Roomba camera can locate the x-center of the toolbox more conveniently. 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 3)

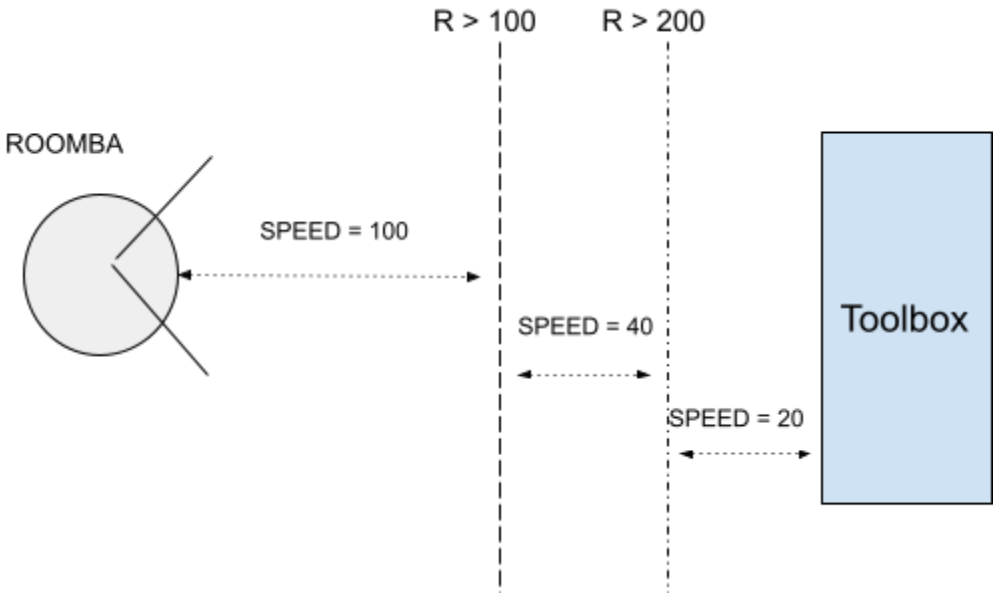


Figure 3: Roomba Traveling Velocity Graph

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) 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 and 100, the Roomba travels with a speed of 100 rev/min, and when the radius is larger than 100, the Roomba will slow down to 40 rev/min, and when the radius is larger than 200, the Roomba will travel at a speed to 15 rev/min for 2 seconds and stop in front of the toolbox. The stopping distance is about 2 centimeters away. 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.

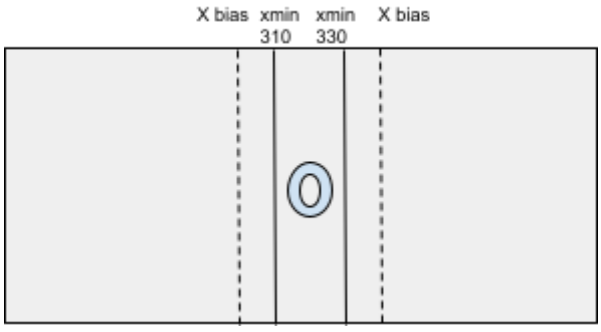
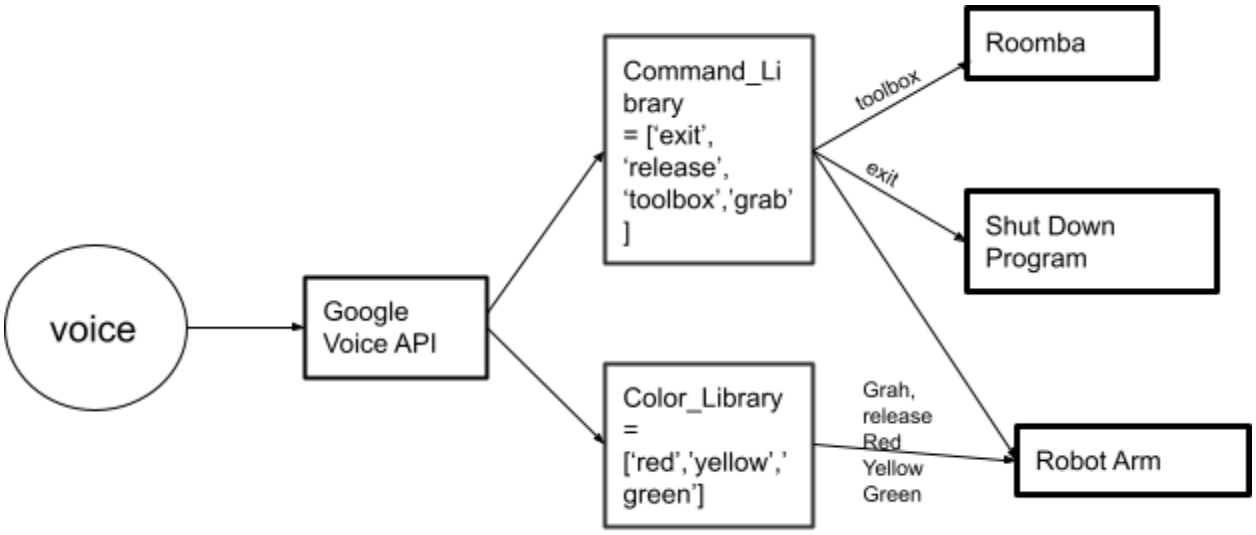


Figure 4: Toolbox drawing with xmin, xmax, and bias

4. Voice Recognition

Speech_recognition 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. 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.



5. Main Program Logic

First, we define all of the above algorithms as functions with user input as connections. To form a complete loop to ensure that each function can be executed in the preset order.

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 severe collisions. 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. For example, the processing speed and accuracy of the speech recognition function are not as good as expected. Because the grabbing distance of the robotic arm is fixed, sometimes the grabbing fails.

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 lit and YOLO.

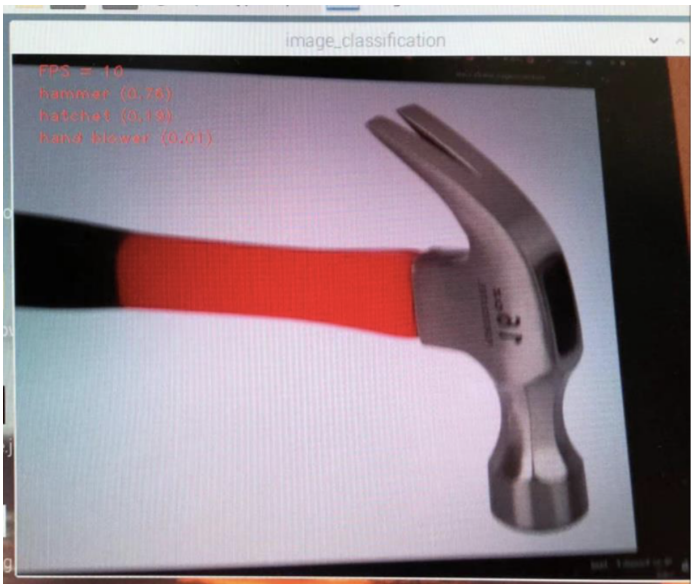


Figure 6.Identify the hammer shown on the screen

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

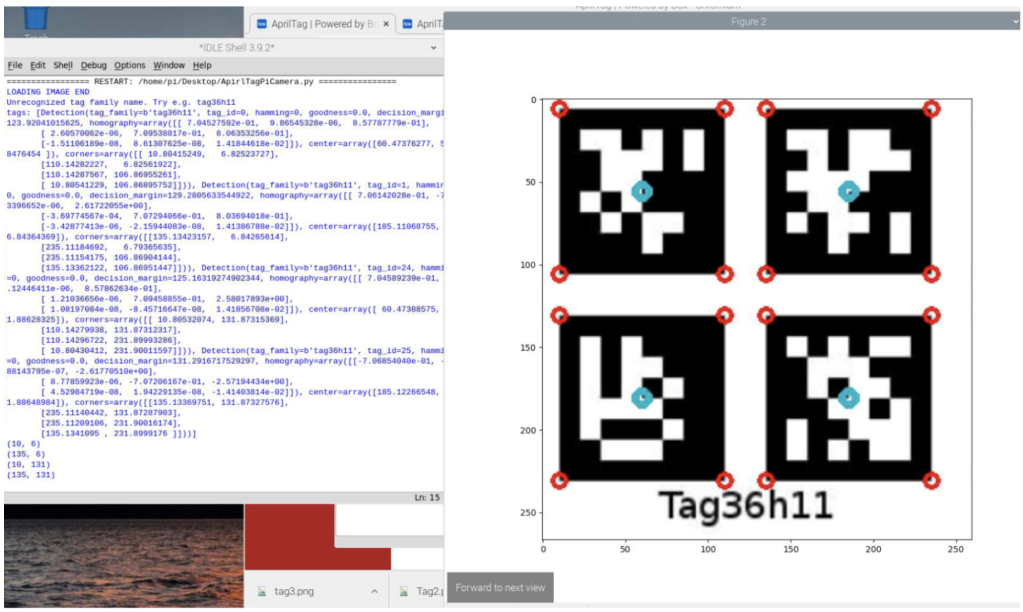


Figure 7.Identify different Apriltags in the same family

After our tests, these two methods are achievable, and will be more accurate and convenient than the current algorithm.

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 robot 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 robot arm system to 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.