POLITECNICO DI MILANO
Master's degree in Computer Science and Engineering
Department of Electronics, Information, and Bioengineering

Design and Implementation of Robot with Automatic Cruise and Garbage Sorting Functions

AIRLab
Artificial Intelligence and Robotics Laboratory
of the Politecnico di Milano

Master Thesis of:
Yuxiang Long, matricola: 904240
Wenshuai Tang, matricola: 894181

Supervisor: Prof. Andrea Bonarini

Academic Year 2019-2020

Abstract

This thesis focuses on designing and implementing a robot which can identify the types of garbage through the color of the garbage bag and has an automatic cruise function in a limited room, including obstacle detection, approaching, and avoiding. The implementation of the robot is made in C language running on the Arduino Mega microprocessor. The color sensor provides data to identify the type of the garbage back thanks to an algorithm based on machine learning classification algorithms which includes Logistic Regression, Support Vector Machine. The algorithm implementation is based on Python and the data-set for learning has been collected by ourselves presenting samples from hundreds of objects with different colors. The distance detection algorithm and the functions mentioned before are based on the three sonars on the robot, pointing to different directions. This thesis includes the pre-knowledegments needed to implement the functions, the hardware and software developments, the final structure of the robot, the final mathematical models and results of the color detector get from the classification algorithms. this work also leaves some promising directions for future developments.

Sommario

Questa tesi si concentra sulla progettazione e realizzazione di un robot in grado di identificare i tipi di immondizia attraverso il colore del sacco della spazzatura e ha una funzione di crociera automatica in una stanza limitata, compreso il rilevamento di ostacoli, avvicinamento, ed evitare. L'implementazione del robot è realizzata in linguaggio C in esecuzione sul microprocessore Arduino Mega. Il sensore di colore fornisce i dati per identificare il tipo di immondizia grazie ad un algoritmo di classificazione di Machine Learning che include Logistic Regression e Support Vector Machine. L'implementazione dell'algoritmo è basata su Python e il set di dati per l'apprendimento è stato raccolto da noi stessi presentando campioni di centinaia di oggetti con colori diversi. L'algoritmo di rilevamento della distanza e le funzioni menzionate prima si basano sui tre sonar del robot, che puntano in direzioni diverse. Questa tesi include le preconoscenze necessarie per implementare le funzioni, gli sviluppi hardware e software, la struttura finale del robot, modelli matematici finali e risultati del rivelatore di colore ottenuti dagli algoritmi di classificazione. Inoltre, questo lavoro presenta alcune promettenti indicazioni per sviluppi futuri.

Acknowledgement

On the occasion of the paper completed, we would like to thank those people who help and guide us in our study and research. We sincerely thank for their friendships, cares and well-intentioned criticisms to us.

First of all, we want to express the highest gratitude to our project advisor, Professor Andrea Bonarini, for the patience instruction and practical assistance. We thank him for provide us the opportunity to join in the AIRLab and complete our project. When robot performance is not as we expected, we bothered him several times, but he always replied us efficiently and effectively. His suggestion always point out the key problems and enlighten us how to improve our works.

Then we want to thank Ing. Giulio Fontana and Mr. Giacomo Colombo who are the colleagues of AIRLab. They gave us some valuable recommendations on how to manipulate the mechanics in the lab. The experience we learnt from them will definitely benefits us for the rest of our engineer career.

We would like to specially mention our roommate Mr. Liang Hu and Mr. Liang Dong. During the hard times when COVID-19 outbreaks, they gave us the encouragement and strength to carry on.

We should finally express our gratitude to our loved parents for their continuous support and efforts.

Contents

List of Figures

Chapter 1

Introduction

This chapter describes the motivations of our project, and the structure of the whole document.

1.1 Background

In the world nowadays, garbage sorting is a very meaningful topic, which has made outstanding contributions to environmental protection and resource reuse. In Italy, the waste sorting started from 1975, saved a lot of energy and resources, yet still need to face the rubbish crisis [1]. In our home country, China, Shanghai becomes the pioneer of waste sorting in 2019 [2]. It can be foreseen that more cities in China will join in the green recycle family in the future. Other countries, such as Japan and Singapore, also challenged by waste crisis and execute waste sorting policies to relieve the stress given by the rubbish. In conclusion, the waste sorting is a world-class challenge.

Unfortunately, sometimes the citizens put the garbage in the wrong trash bin by mistake, what's more is that sometimes the trash bin is lack of the corresponding identification of a certain type of garbage, which makes citizens difficult to throw the garbage in the right trash bin. The problems mentioned above will cause a lot of trouble for the staff responsible for garbage disposal and may cause problems below:

- 1. For staff responsible for garbage disposal: wasting time on checking the garbage and classifying them correctly, which increases a huge workload.

- 2. For the world and society: wasting resources caused by incorrect garbage classification, which reduces the efficiency of renewable and reusable resources.

- 3. For the humans: incorrect garbage may causes some health problems, for example, garbage containing heavy metals like batteries, if they are not properly classified and recycled, they will pose a potential and huge threat to human health.

1.2 Brief Description of the Work

With the situation faced, what we want to do is to design a new automated trash-bin which can correctly identify the type of garbage, if the garbage which is going to be thrown is the right type, the trash-bin will open the lid and accept it, otherwise the trash-bin cover will keep closed. In this way, we can greatly reduce the error rate of garbage classification.

The garbage sorting robot with automatic cruise function can automatically cruise in a certain area. Once it detects a target (for example, the people who want to throw the garbage, or obstacles) within a certain range, it will move towards the target, then stops within a certain distance from the target and turn on the garbage type detection module, regardless of whether the corresponding type of garbage is detected, the automatic cruise function will continue to be turned on after waiting for a period of time, seeking another potential user to put the rubbish in.

The project is based on Arduino Mega with C language, all the math models trained from the data-set are based on Python with Jupyter Notebook, the notebook can makes our work and result more intuitive and efficient.

1.3 Document Structure

This work is organized in the following chapters:

- Chapter 1: A brief introduction to the situation and the work-flow of this thesis.
- Chapter 2: The problems we need to solve and the solutions with principles.
- Chapter 3: The detailed description of hardware we use.
- Chapter 4: The detailed description of the software development.
- Chapter 5: The results of the math models and the realization of garbage sorting robot.
- Chapter 6: Sums up the conclusions and suggests possible ways to further develop this work.

Chapter 2

Conceptual Descriptions of the Problems Faced and the Solutions.

2.1 The Obstacle Detection Problem

The rubbish bin should detect the obstacle, in our case, the user who try to throw some rubbish. Once detected, it will move towards the user. Therefore, the bin should sense the direction and the distance between the bin and the user all the time, in order to control itself when and how to move or brake.

2.1.1 The Limitation of the Triangular Detection Method

Ideally, by using only two sonar sensors, we can detect the obstacle's distance and direction. Indeed, if we can obtain the two distances between the obstacle and the two sensors, with the relative sonar sensors position, a firm triangle can be derived. However, unlike antennas can have a wide angular range of 360 degrees, the sonar sensors only have a detection range of 20 degrees. For this reason, the triangular detection method limits the measured position, makes this measure strategy not suitable.

Figure 2.1 shows how the triangular detection method will make the most of area become a "dead" area, where it is unable to obtain the position of the obstacle.

Because of this, a simple 2-sonar trial will be impossible to complete the functions we initially desired.

2.1.2 The Three-Sonar Detection Method

As is mentioned in the fail of triangular detection method, if the obstacle is not properly aligned within the detecting angle range, simply the detector will not detect the obstruction and will risk a collision between our bin-base and the object.

This risk could be avoided by enhanced the detecting angle range. In practice, our solution is using 3 sonar sensors, arranged in fairly decent positions. The detecting angle range will therefore be approximately $\pm 30°$. Figure 2.2 shows this kind of position arrangement.

From this arrangement, it can be discovered that compared with the triangular method, the detecting angle range improve quite a lot. The "dead" zone around the bin base

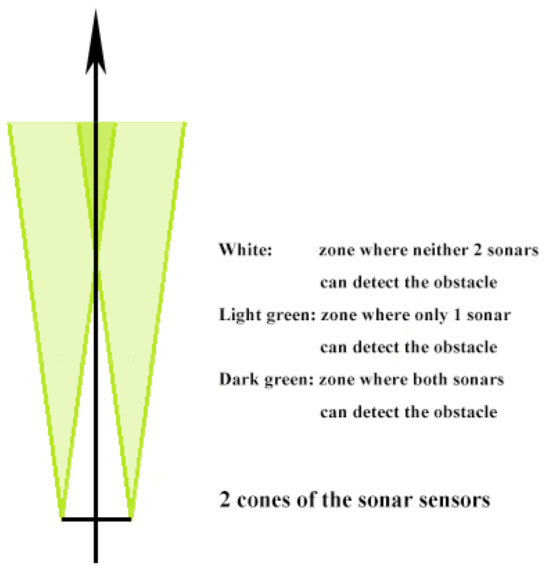

White: zone where neither 2 sonars can detect the obstacle
Light green: zone where only 1 sonar can detect the obstacle
Dark green: zone where both sonars can detect the obstacle
2 cones of the sonar sensors

Figure 2.1: The Triangular Detection Method Limits the Detection Area.

shrinks, make the collision less possible to happen, also considering that the robot mainly moves forward with relatively slow rotations.

2.2 The Bin Base Dynamic Control Problems

The bin base dynamics control strategy is a feedback logic. The sonar sensor values give a feedback signal, to which the base motors react.

From the procedure point of view, the base will undergoes three main steps:

(1) Before the obstacle is sensed, the bin base will move randomly.

(2) Once the user is detected, the bin will move to the user, which might consist of some left or right rotation and forward movement, and brake when the distance between the bin and user is less than 25cm.

(3) After the user put the rubbish in, or the bin denies the rubbish, the bin should move away from the user and start a new loop.

The most critical procedure is the second step, since the bin is both rotating and forwarding. Therefore, it has to be balanced between rotating and forwarding. If there is more rotating than forwarding, then the bin dynamic would vibrate, act like shaking left and right. Vice-versa, if the forwarding movement overweight, the trace of the bin will be less turned, and this results in a risk of collision.

Our solution is to apply a PID control to the rotation. The main issue is to find the suitable P coefficient, which is the proportional part.

For example, an obstacle is detected to the left of the bin, the bin should react with a left turn. If we apply a PID control strategy, if the obstacle is to the left but far from the

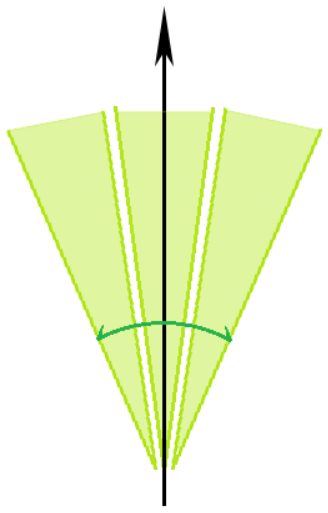


Figure 2.2: Angle Range Enhanced by Using Multiple Sonar Detectors.

bin, a slight left rotation would be enough to bent the trace. However, the obstacle is to the left, but quite close to the bin, an urgent sharp left turn should happen.

We also have to discuss the I and D coefficients of the PID control. The I part is used to correct the small offset error of the trace. Even though we place the center sensor on the middle of the bin, the sonar sensor still does not sense the target in the mid area, due to the sonar's trigger and echo cylinders are not symmetric. Therefore an offset is present and needs to be corrected. The D part is used to overcome the large inertia of the system, and improve the time response.

Notice that the output of PID control is:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{2.1}$$

Where K_p, K_i and K_d are the PID coefficients, and e(t) is the error of the controlled factor at time t.

However in our case, we have to change the equation into a discrete formula:

$$u(k) = K_p e(k) + K_i \sum_{i=1}^{k} e(i) + K_d[e(k) - e(k-1)] \tag{2.2}$$

Where K_p, K_i and K_d are the PID coefficients, and e(k) is the error of the controlled factor sampled in a duration k.

2.3 The Classification Problems

We want the robot can detect the types of garbage, for example: metal, plastic or food residue, there exists a problem is that it is hard to analyse the ingredient of the garbage;

fortunately, in the same region,fixed color garbage bags are defined to correspond to different types of garbage, so with the help of people putting garbage in the correct color garbage bag, we can turn this problem into a problem of identifying colors, this is a classical classification problem, where it is possible to use some machine learning technologies.

2.3.1 Data Collection and Preparation

We collected samples as the data set including different objects with different colors, the object with color yellow marked as "1" and others marked as "0", we let the numbers of the RGB as three inputs and the type "is yellow" as output. In order to test our models' performances, we divided the data set into training set (80%) and test set (20%). Due to the number of the samples is not so big, we drop the validation set and use cross validation to check the performance [3].

2.3.2 Logistic Regression

The logistic regression [4] is one of the efficient ways to do the classification, we compute the probability of assigning a label (in our situation, the label is yellow or not yellow) to identify whether the color that the sensor detects is what it needs or not; the key function of logistic regression is called sigmoid function:

$$\begin{aligned} P(\hat{y}_i = +1|x_i) &= \frac{1}{1+e^{-\omega^T x}} \\ P(\hat{y}_i = 0|x_i) &= 1 - P(\hat{y}_i = +1|x_i) \end{aligned} \tag{2.3}$$

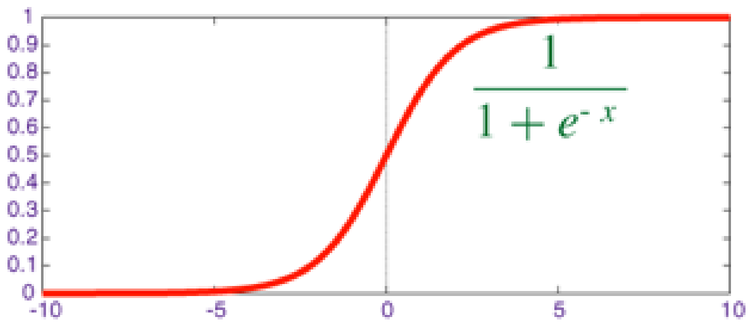


Figure 2.3: Sigmoid Function

The output of the sigmoid function is in (0,1), through this feature we can compute the probability of the object which the sensor samples belongs to the class the machine needs, usually if the probability is greater than 0.5, we take that the object belongs to the class 1, otherwise it belongs to class 0.

As we all know, we use loss function to judge the performance of the model, in logistic regression, we can think the sigmoid function as the probability density function, so we use likelihood function to compute the loss function; in order to compute it more conveniently, we usually use the log likelihood function:

$$
\begin{aligned}
L(\omega) &= \prod_{i=1}^{N}[p(x_i)]^{y_i}[1-P(x_i)]^{1-y_i} \\
l(\omega) &= log(L(\omega)) \\
&= \sum_{i=1}^{N}[y_i log p(x_i) + (1-y_i) log(1-p(x_i))] \\
&= \sum_{i=1}^{N}[y_i log\frac{p(x_i)}{1-p(x_i)}] \\
&= \sum_{i=1}^{N}[y_i(\omega_i \cdot x_i) - log(1+e^{\omega_i \cdot x_i})]
\end{aligned}
\tag{2.4}
$$

The goal is finding the parameters ω which make the loss function to be maximum. In our situation, we have 3 inputs so the matrix is:

$$
\omega^T \cdot x = \begin{pmatrix} \omega_1 & \omega_2 & \omega_3 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}
$$

If we take the average log-likelihood loss over the entire data set, we can get:

$$
\begin{aligned}
J(\omega) &= -\frac{1}{N} log(L(\omega)) \\
&= -\frac{1}{N}\left(\sum_{i+1}^{N} y_i(\omega_i \cdot x_i) - log(1+e^{\omega_i \cdot x_i})\right)
\end{aligned}
\tag{2.5}
$$

2.3.3 Stochastic Gradient Descent

We can find the descending direction of the function through partial $J(\omega)$ over partial ω and iteratively update the parameters until the function converges, we can update the parameters through:

$$
g_i = \frac{\partial J(\omega)}{\partial \omega} = (p(x_i) - y_i)x_i \tag{2.6}
$$

$$
\omega_i^{k+1} = \omega_i^k - \alpha_i g \tag{2.7}
$$

k represents the number of the iterations and the function converges when:

$$
\left| J(\omega^{k+1}) - J(\omega^k) \right| < \delta \tag{2.8}
$$

δ is the threshold we set to stop the iteration, and α represents the descending rate. When the function converges, we can get the parameters make the loss function local optimal but not global optimal.

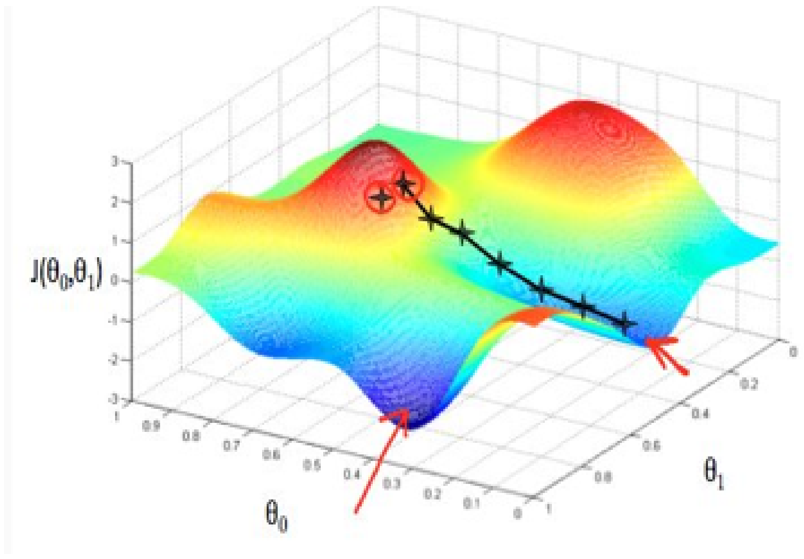


Figure 2.4: The Process of SGD

2.3.4 Batch Gradient Descent

The batch gradient descent is based on the stochastic gradient descent, on the same time, it updates the parameters using the data set in order to find the global optimal solution[5], so the update strategy becomes:

$$\omega_i = \omega_i - \alpha \frac{1}{m} \sum_{j=0}^{m} (p_\omega(x_0^j, x_1^j, , , x_n^j) - y_j) x_i^j \tag{2.9}$$

With this strategy, we can find the optimal solution, but at each iteration, it will use all the data-set, for this reason, the training will become very slow if the data-set is large.

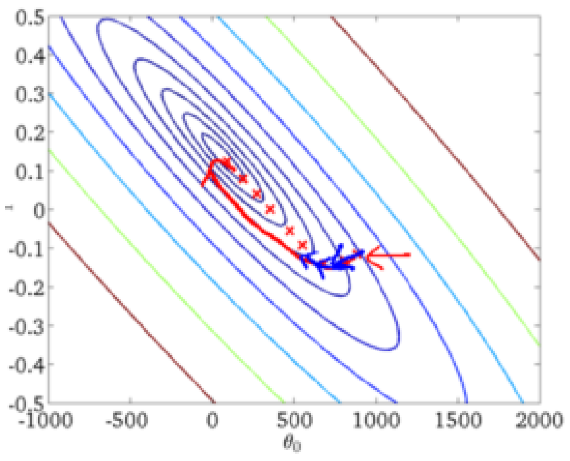


Figure 2.5: The Process of BGD

2.3.5 Min-batch Gradient Descend

The-Min batch Gradient Descend takes the advantages of both the BGD and the SGD algorithms: at every iteration only takes a fixed number of samples (randomly) to help update the parameters, for example 20; in this case, the MBGD can get a better result than SGD algorithm without adding much training time:

$$\omega_i = \omega_i - \alpha \frac{1}{20} \sum_{j=0}^{m} (p_\omega(x_0^j, x_1^j, , , x_n^j) - y_j) x_i^j \tag{2.10}$$

2.3.6 Regularization

To avoid over-fitting, we need regularization to limit the complexity of the model, there are two usually used regularization methods called L1 and L2 regularization, the Lasso regression add a priori probability to the model: the ω obeys the zero-mean Laplace distribution:

$$f(\omega|\mu, \lambda) = \frac{1}{2\lambda} e^{-\frac{|\omega-\mu|}{\lambda}} \tag{2.11}$$

Then the likelihood function becomes:

$$\begin{aligned} L(\omega) &= P(y|\omega, x)P(\omega) \\ &= \prod_{i=1}^{N} p(x_i)^{y_i}(1-p(x_i))^{1-y_i} \prod_{j=1}^{d} \frac{1}{2\lambda} e^{-\frac{|\omega_j|}{\lambda}} \end{aligned} \tag{2.12}$$

Then take the log of the loss function:

$$logL(\omega) = \sum_{i=1}^{N} [y_i logp(x_i) + (1-y_i) log(1-p(x_i))] + \frac{1}{2\lambda^2} \sum_j |\omega_j| \tag{2.13}$$

Another one is Ridge regression, it adds a priori probability to the model: the ω obeys the zero-mean normal distribution:

$$f(\omega|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\omega-\mu)^2}{2\sigma^2}} \tag{2.14}$$

Then the log loss function is:

$$logL(\omega) = \sum_{i=1}^{N} [y_i logp(x_i) + (1-y_i) log(1-p(x_i))] + \frac{1}{2\sigma^2} \omega^T \omega \tag{2.15}$$

2.3.7 Bagging Algorithm

Normally, we can build a strong classifier with several weak classifiers, according to the results of the weak classifiers through voting, to get the final result of the strong classifier. More details on [6].

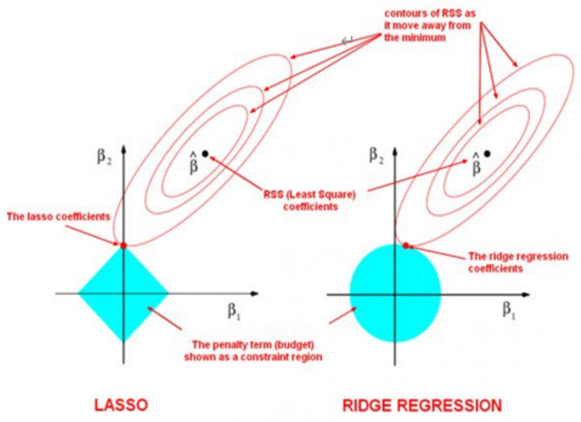

contours of RSS as it move away from the minimum
β_2
$\hat{\beta}$
RSS (Least Square) coefficients
The lasso coefficients
The ridge regression coefficients
β_1
The penalty term (budget) shown as a constraint region
LASSO
RIDGE REGRESSION

Figure 2.6: The Comparison Between Lasso and Ridge Regression

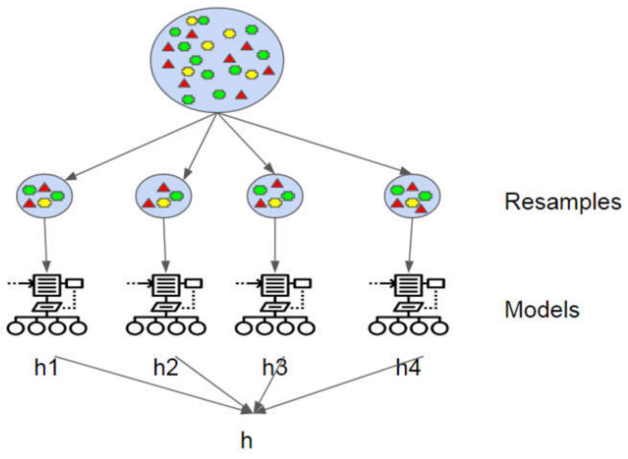


Figure 2.7: Bagging Algorithm

2.3.8 Support Vector Machine

Support vector machine is one of the best methods used for classification, so we tried to use support vector machine to find a surface that could separate the yellow points and not-yellow points; more details on [7].

As the picture shows in the figure below,there are two kinds of points, blue and red, if the domain of the space is 2, we can use only a straight line to separate the blue and red points if the points are linearly separable.

If the points are not linearly separable, we can use a curve to separate them, but the issue is that there exists an infinite number of lines that can separate them, for example see Figure 2.8 and Figure 2.9.

In order to find the best line which can separate the points as far as possible, in other words, we need to find the line which has the biggest margin within the red and blue points. So the constraint is:

$$\gamma = \min_i y_i(\omega^T x_i + b) \tag{2.16}$$

So, we want to find the the maximum margin γ, then our goal is to find the ω that

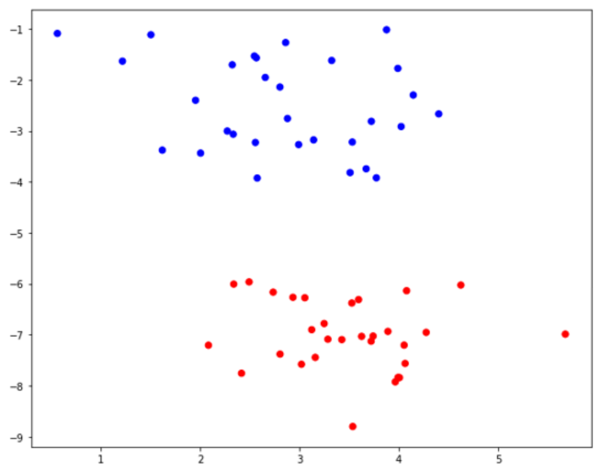


Figure 2.8: The Data Distribution Example

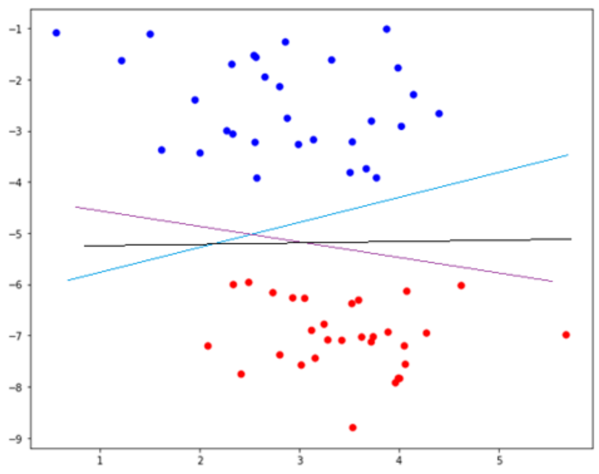


Figure 2.9: Lines Separate the Two Classes

satisfies:

$$\omega = \arg\min_{\omega,b}\left(\frac{1}{|\omega|}\min_{i}(y_i(\omega^T x_i + b))\right) \tag{2.17}$$

According to the duality principle the solution becomes:

$$\min_{\omega,b} \frac{1}{2}|\omega|^2 \tag{2.18}$$

$$s.t. \quad y_i(\omega^T x_i + b) - 1 \geq 0$$

Then using the Lagrange Multiplier method we can get the duality problem:

$$L(\omega, b, \alpha) = \frac{1}{2}|\omega|^2 + \sum_{i=1}^{n} \alpha_i(1 - y_i(\omega^T x_i + b))\alpha_i \geq 0 \tag{2.19}$$

Through taking partial L over partial ω and partial b: $\frac{\partial}{\partial \omega}L(\omega, b, \alpha)$ the question becomes:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i, \alpha_j, y_i, y_j, k(x_i, x_j)$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{2.20}$$

$$\alpha_i \geq 0, i = 1, 2, 3, ..., N$$

Then we can get the optimal solution of α then get the optimal solution of ω and b:

$$\alpha^* = (\alpha_1^*, \alpha_2^*, ..., \alpha_N^*)$$

If we use linear kernel k, the optimal surface is:

$$\omega^* x + b = 0$$

.

Finally we use sign function to assign the label of the data:

$$y = sign(\omega^* x + b)$$

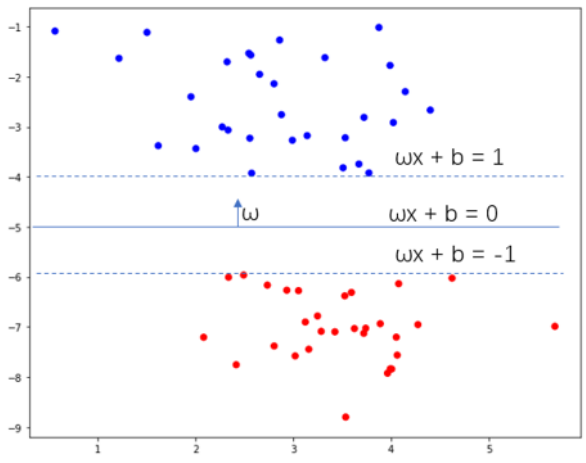


Figure 2.10: Support Vector Machine

2.3.9 Noise Handing

The ideal linearly separable nearly does not exist in reality, in most cases there is some noise like in Figure 2.11:

If we try to separate them completely with curve, those noise points will obviously make the model over-fitting, so we add a slack variable $\eta_i \geq 0$ to the constrain, then we get:

$$\min_{\omega,b,\eta} (\frac{1}{2}|\omega|^2 + C \sum_{i=1}^{N} \eta_i) \tag{2.21}$$

$$s.t. \qquad y_i(\omega^T x_i + b) \geq 1 - \eta$$

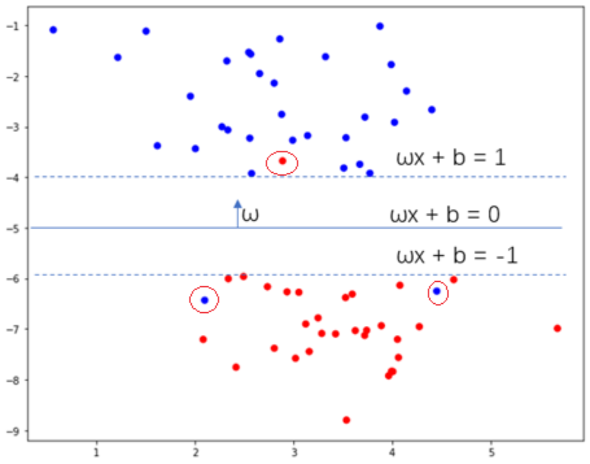


Figure 2.11: Noise in SVM

The parameter C is called penalty coefficient, it represents the tolerance to the noise, the bigger is C, the lower the tolerance to the noise. So we also use the Lagrange Multiplier to transfer the problem into a dual problem:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i, \alpha_j, y_i, y_j, k(x_i, x_j)$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{2.22}$$

$$C \geq \alpha_i \geq 0, i = 1, 2, ..., N$$

The final model is:

$$y = sign(\omega^* x + b) \tag{2.23}$$

The $k(x_i x_j)$ is the kernel function with many types, in our example show in the Figure, it is linear linearly separable, so the kernel function is linear kernel function, below are some common kernel functions:

$$LinearKernel : k(x_i, x_j) = x_i \cdot x_j$$

$$PolynomialKernel : k(x_i, x_j) = (x_i \cdot x_j)^d, d \geq 0$$

$$GaussianKernel : k(x_i, x_j) = e^{-\frac{|x_i - x_j|^2}{2\sigma^2}}$$

$$SigmoidKernel : k(x_i, x_j) = tanh(\beta \cdot x_i x_j + \theta), \beta{>}0, \theta{<}0$$

The kernel function also can be the linear combination of arbitrary kernel functions:

$$k(x_i, x_j) = \alpha_1 k_1(x_j, y_j) + \alpha_2 k_2(x_j, y_j) \tag{2.24}$$

Chapter 3

Detailed Description of Hardware

This chapter introduces the hardware part of the trash-bin project. This hardware part will be carefully analyzed from its working principles, and then we discuss what we have done to achieve our aims.

3.1 Wheels of the Bin Base

As it is shown in Figures 3.1 and 3.2, the wheel has two type of components. The majority of wheels can be driven by a motor, and this defines its active direction. In omniwheels there are several small free-rolling barrels, placed orthogonal to the active direction, we can call as passive direction.

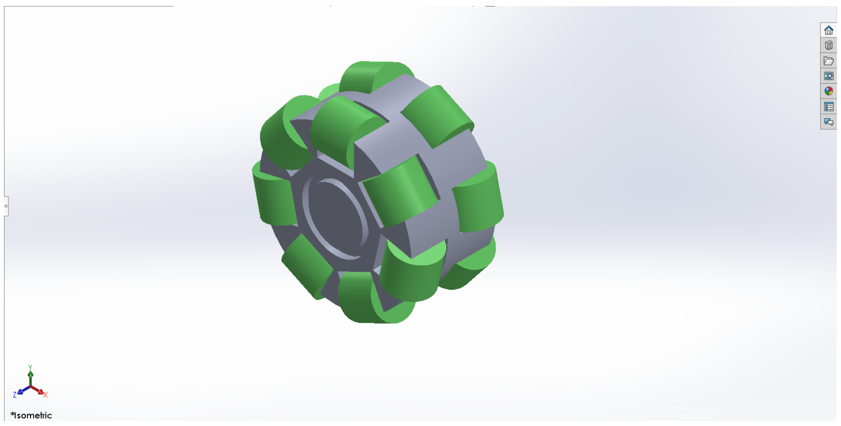


Figure 3.1: A 3D View of A Single Wheel (sketch in SolidWorks).

In reality, this wheel act as a universal wheel. The vectors used to describe the wheel dynamics, such as velocity vector and acceleration, can be resolved orthogonal to the active and passive directions.

3.1.1 The Three-wheel Base Motion

The base is composed by three wheels in a triangular distribution. Obviously, it can be easily find that the base can act two kinds of dynamics, rotation and slide. Of course it can behave more since it is only a linear system, therefore the a more complicated movement

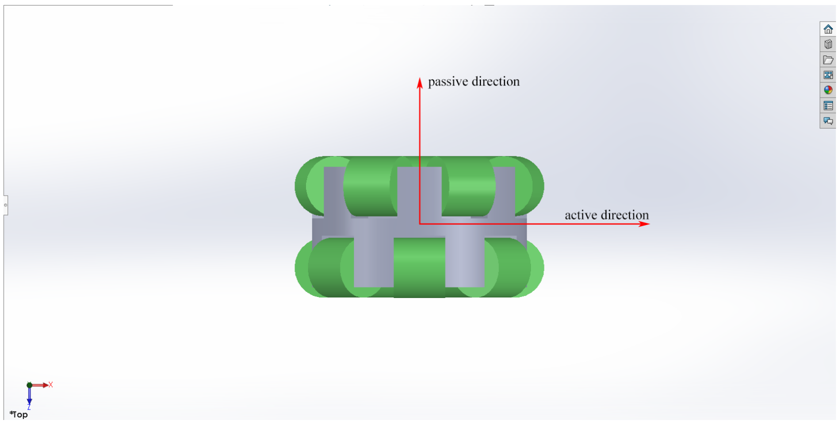


Figure 3.2: Vertical View of a Single Wheel (sketch in SolidWorks).

is only some superposition of some simple movements. Before our analysis, we assumed that the wheels are well in contact with ground, in other words, the frictions between the wheels and ground are static fictions.

3.1.2 Rotation of the Base

Compared to slide movement, rotation is much easier to realize in our scenario. Three wheels moving clockwise or counterclockwise simultaneously can make the base rotate. Now we introduce the maximum velocity given by a single wheel, v_{max}, as well as the distance between the wheel and the center of the base, R, we can easily derive that the maximum rotation angular velocity is

$$\omega = \frac{v_{max}}{R}$$

Although we can control our motors to make the angular velocity less than its maximum, we want our response fast thus in our application we always try to hit this ceiling.

3.1.3 Slide Movement

The slide dynamic is tricky to analyze because, given a slide direction, this direction has to be resolved in three orthogonal vector pairs, related to the three wheels.

Starting with the easiest circumstance moving forward or backward, as shown in the Figure 3.3, wheels #1 and #2 move forward as two driving wheels, while wheel #3 holds still as a passive wheel.

Again now we consider the relation between the velocity of moving forward (or backward) with the maximum velocity of the wheel, v_{max}. We immediately notice that the wheel #1 and #2 are symmetric so that their contribution are identical, as a result:

$$v_{fb} = v_{max}/cos(\frac{\pi}{6}) = \frac{2\sqrt{3}}{3} v_{max}$$

$$v_{wheel1} = v_{wheel2} = V_{max} \tag{3.1}$$

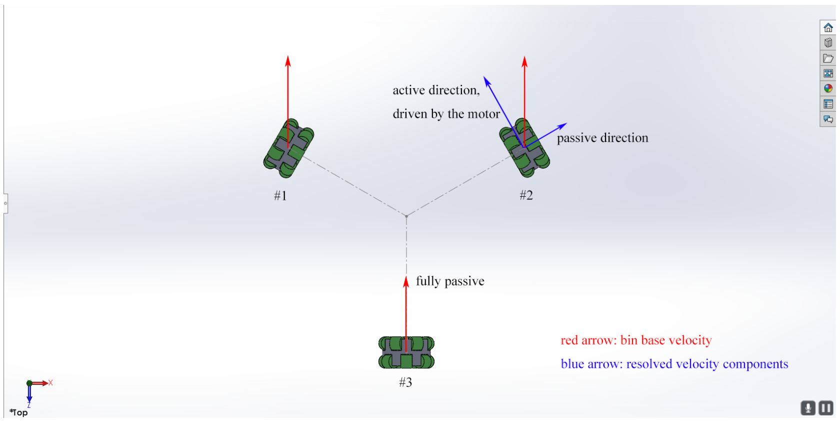


Figure 3.3: Base Forwarding and Velocity Resolution (sketch in SolidWorks).

$$v_{wheel3} = 0$$

Let us make some comment on this result, to control the base to move forward or backward is fairly elementary. Especially when we want our bin move at its highest speed, since the moving velocity and the wheel velocity have a constant ratio. Moreover, we only drive wheel #1 and #2, and leave #3 untouched.

Now we discuss another case, the base sliding left or right, as it is performing a shuffle dance. This case is shown in Figure 3.4.

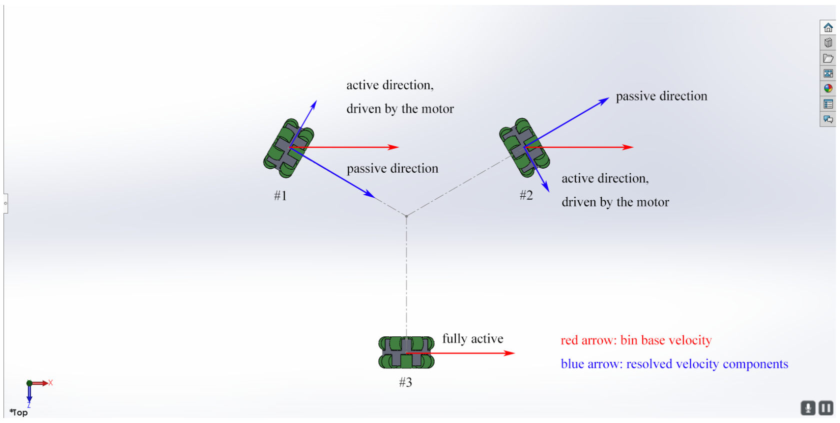


Figure 3.4: Base Shuffle and Velocity Resolution (sketch in SolidWorks).

We resolved the velocity vector in 3 axis and discovered that wheel #3 fully contributes its motor velocity, meanwhile wheel #1 and #2 have their corresponding resolved components. Again we have to consider the constraint about the maximum velocity of the wheels. In this shuffle motion circumstance, wheel #3 limits the shuffle velocity. Therefore, as a result of the simple geometry:

$$v_{shuffle} = v_{wheel3} = V_{max} \tag{3.2}$$

$$v_{wheel1} = v_{wheel2} = V_{max}cos(\frac{\pi}{3}) = \frac{1}{2}V_{max}$$

Again, let us make some evaluation to this result. Firstly, compared with the velocity of forward and backward movement, the left and right shuffle mode provides less velocity. Secondly, we have to take control to wheel #1 and #2, limit them to only half of the velocity of wheel #3 to match the shuffle movement.

After the analysis of the 2 cases, we question ourselves, since moving forward and backward provides greater velocity and is easier to realize, is it always better to use only the forward and backward mode? Indeed, from the previous evaluation we immediately find two drawbacks, although they might be an advantage in disguises. In general cases, motor is driven by a DC motor and use the strategy of PWM control. Therefore, from the point of power dissipation, the power dissipation of forward and backward movement:

$$P_{fb} \propto v_{max}^2 + v_{max}^2 + 0 = 2v_{max}^2 \tag{3.3}$$

While the power dissipation of shuffle:

$$P_{shuffle} \propto (\frac{1}{2}v_{max})^2 + (\frac{1}{2}v_{max})^2 + v_{max}^2 = 1.5v_{max}^2 \tag{3.4}$$

That is, the power dissipated when the base move in shuffle mode is less than that in forward and backward movement. This result is pretty intuitive, since the shuffle speed is lesser, the battery delivers lesser electric energy translated to kinetic energy of the base. Vice-versa, if there appears a power bottleneck of the battery, working in shuffle mode will give greater velocity.

Let us give a final comment on these two modes. It is very interesting that we find the two modes are similar to the manual gears of an automobile. When the road is flat and you are aiming to get the maximum velocity, it is better to find your direction and move forward. When there is a slope, you would better to slow it down and enter the shuffle mode because you dissipate less power, and, because in shuffle mode the base is driven by three wheels instead of only two.

We still have to ask a final question, is there is another mode can provide us a faster speed or is there another mode dissipate less energy? After analysis via an universal velocity vector, the answer is no, which confirms our assumption above.

3.2 Ultrasonic Detectors HC SR-04

The ultrasonic detectors, also called as sonar sensors in short, is applied to detect the distance between the sensor and the possible obstacle. In our application we use HC SR-04 as our sonar sensors. Figure 3.5 shows the sonar sensor we adopted.

The working principle is based on the speed of sound: under the condition of the media of air and room temperature, is a constant approximately equals to 340m/s. When a sonar detector is triggered, it emits several pulses of ultrasonic signal. The ultrasonic signal hits the target and echos. The echo is received by the sonar detector, and the time difference between the trigger and the received echo is the transmission time. Since the sound speed is constant, the distance between the ultrasonic sensor and the targeting object can be derived as:

$$D = \frac{1}{2}v_{sound}T \tag{3.5}$$

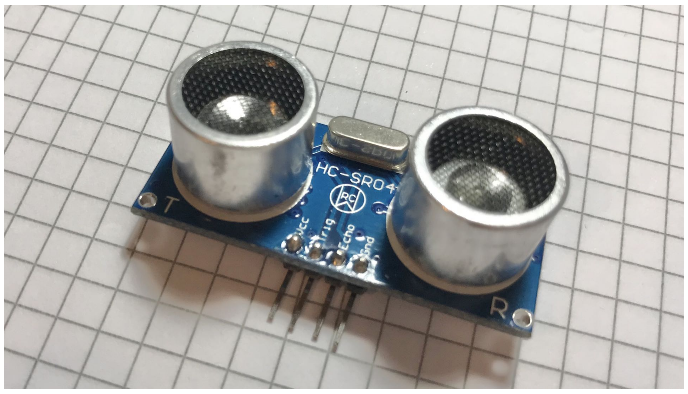


Figure 3.5: Sonar Sensor HC SR-04, 4 Pins are VCC, TRIG, ECHO and GND.

Where T is the transmission time, or in another word, the time of flight. Since the pulsed signal undergoes is a sending and receiving process, there is a 1/2 factor in the equation. Figure 3.6 shows the geometry of sonar sensor detection.

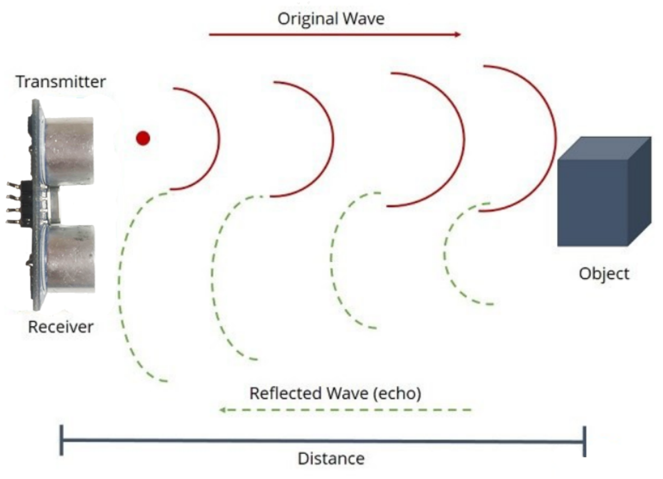


Figure 3.6: The Geometry of Sonar Sensor Detection

From the figures shown above, we notice that the sensor have some cylindrical trigger and receiver structure. This kind of topology will limit the detecting angle, usually the angular range is $\pm 10°$. Therefore, the position of the sonar sensors should be selected carefully to limit the ”dead” zone and the overlapped zone.

3.2.1 Motor Driver L298N

With a motor driver, we can use the control logic signal to control the motor driving signal. That is, using Arduino output 5 volts, to control a high voltage 12 volts. In our application, we use the L298N chip as motor drivers. It is shown in the Figure 3.7.

The L298N chip is a dual-channel H-Bridge motor driver capable of driving two DC motors simultaneously. In our case, since we only use 3 motors to drive the wheels, we use two chips to drive the motors, with one channel left unconnected.

It also needs to be carefully mounted, since the pins on the back of the chip are left

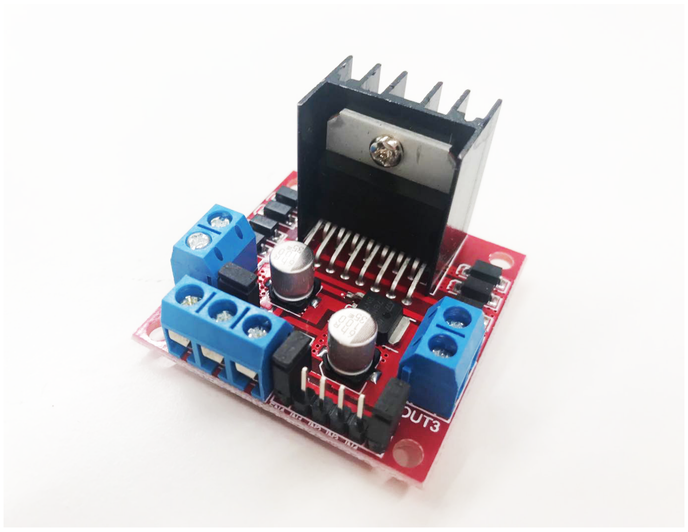


Figure 3.7: Dual-channel Motor Driver L298N.

quite exposed. If this chip is mounted on a metal surface, there will be possible short circuit between the VCC and GND. Because the motor driver has 12 volts, the short circuit will be devastating. In reality, we use a thin plastic film as the insulator to avoid the possible short circuit.

The L298N uses a H-Bridge topology to control the motor forward rotation and reversed rotation [8]. As is shown in Figure 3.8, the H-Bridge is composed by four switching elements, in the case of L298N they are MOSFETs, as well as the motor controlled placed in the center, forming a H-like shaping. By activating the diagonal pair of the switches, the motor is forward rotating. On the contrary, if the other pair of switches are on, the motor rotation direction will be reversed.

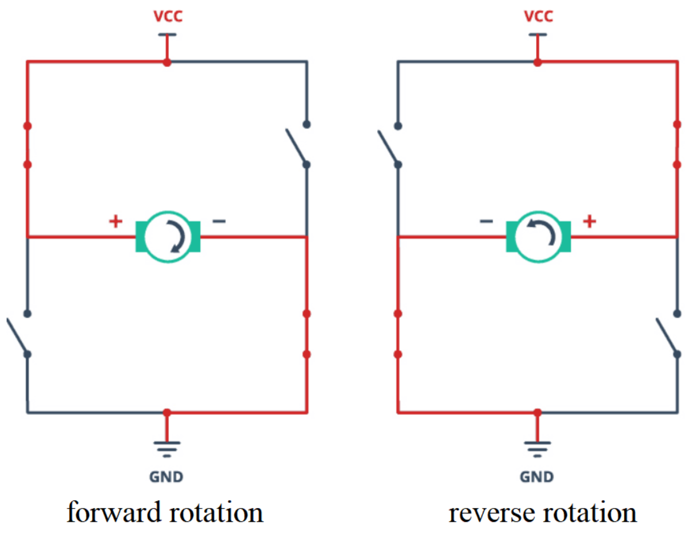


Figure 3.8: H-Bridge of the L298N Motor Driver .

As is mentioned above, the switches are realized by MOSFETs, thus they can be controlled by logic. The table shows the motor behavior with respect to the control logic.

$Input1$	$Input2$	$Spinning\ Behavior$
$LOW(0)$	$LOW(0)$	$Brake$
$HIGH(1)$	$LOW(0)$	$Forward\ Rotation$
$LOW(0)$	$HIGH(1)$	$Reverse\ Rotation$
$HIGH(1)$	$HIGH(1)$	$Brake$

3.2.2 TCS3200 Color Sensor

The TCS3200 color sensor [9] is a "all-color" detect sensor, including a TAOS TCS3200RGB induction chip and 4 white LED lights. The TCS3200 chip can detect all types of the visible light so it is used widely in many fields for example: medical and computer vision.

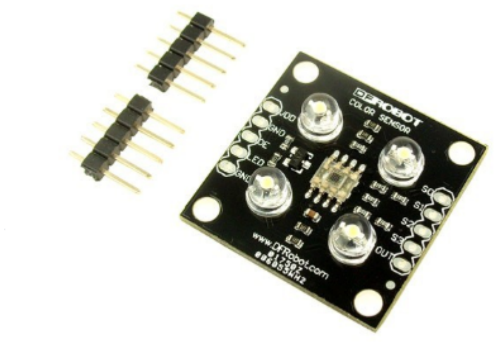


Figure 3.9: Color Sensor TCS3200.

It integrates a configurable silicon photodiode and current-frequency converter on the segmented CMOS circuit, and integrates three red, green and blue (RGB) filters on the split chip, which is the industryâs first digital. The output signal of the compatible TCS3200D is digital and can drive standard TTL or CMOS logic inputs, so it can be directly connected to a converter or other logic circuit. Because the output is digital, and each color channel can achieve a conversion accuracy of more than 10 bits, no A/D conversion circuit is required, thereby simplifying the circuit.

In practice, the color sensor should be used in a dark environment; therefore, when we mount the sensor on our assembly, it is necessary to make some cover in order to filter the ambient light source which distorts the signal. Figure 3.10 shows how we embedded the sensor in a cylindrical cover structure.

3.2.3 Chip Parameters

$Pin\ Name$	I/O	$Description$
$GND(4)$		$GND\ Reference\ Voltage$
$OE(3)$	$Input$	$Enable\ f0\ (active\ low)$
OUT	$Output$	$Output\ Frequency\ f0$
$S0, S1(1,2)$	$Input$	$Output\ frequency\ scaling$
$S2, S3(7,8)$	$Input$	$Photodiode\ type\ selection$
$VDD(5)$		$Power$

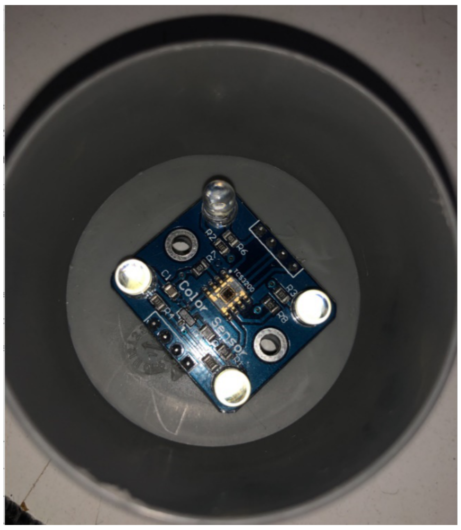


Figure 3.10: Color Sensor Embedded in a Cylindrical Cover Structure.

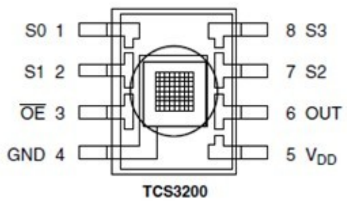


Figure 3.11: Color Sensor Schematics.

According to the table and the chips pin, we connect the chip with the Arduino Mega like Figure 3.12.

3.2.4 Chip Principles

As we know, all the visible colors can be obtained by combining the 3-basic colors, called RGB (R: red, G: green, B: blue). The TCS3200 color sensor can detect the intensity of a specific color by activating different color filters; if we get the intensity of the RGB channels, then we can get the color of the object. Here are the different combinations of the color filters. The color goes into the photodiode and then goes through the current-frequency converter, we can get the square wave signal of the color.

The TCS3200D outputs square waves of different frequencies (gain of 50%). Different colors and light intensities correspond to square waves of different frequencies. The output frequency is linearly related to the light intensity. The user can select the output scale factor of 100%, 20% or 2% through two relative refractive indexes.

$S2$	$S3$	$Color\ Filter$
L	L	Red
L	H	$Blue$
H	L	No
H	H	$Green$

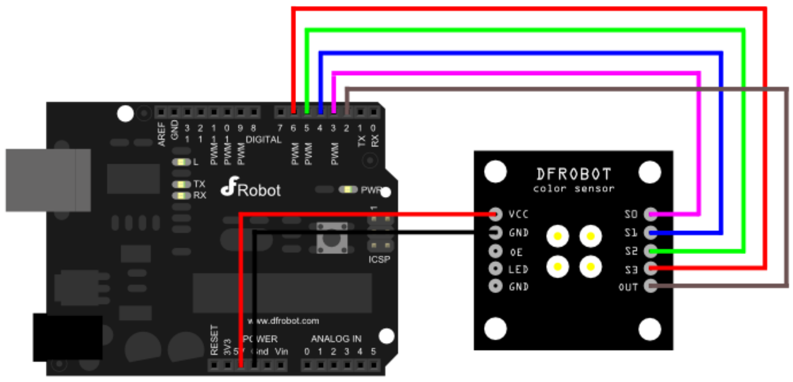


Figure 3.12: Color Sensor Pin Connections.

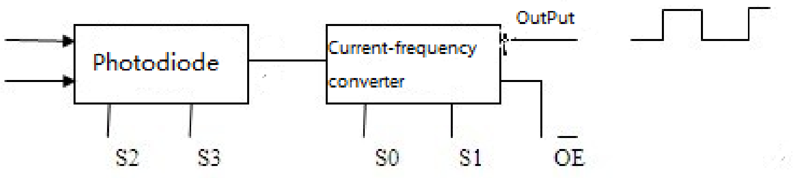


Figure 3.13: Color Sensor Pin Connections.

$S0$	$S1$	$Output\ Frequency$
L	L	$Power\ Off$
L	H	2%
H	L	20%
H	H	100%

3.3 Other Hardware

Beside the hardware mentioned above, there are some other hardware pieces we installed in order to complete our applications.

Arduino Mega 2560 Micro-controller

The well-known Arduino mega 2560 is used as the micro-controller [10].

In our application, we have to pay attention to the pins used as input and output, especially those output pins that can be used as PWM outputs.

The Arduino mega 2560 board can simultaneously output 15 PWM signals, which are related pins to 2-13 and 44-46. However, these PWM output pins might be disabled due to the libraries used in our application. Therefore we have to be aware of pins we select, as well as in software construction, attach and detach the pins in order to disable and enable the PWM outputs.

3.3.1 Servomotor for the Bin Lid Open/Close

Since we want to realize a rubbish bin which can detect the rubbish type, the bin lid should be controlled open and close if the certain incoming logic signal is met.

The bin lid is driven by a servomotor, with some lever mechanism. Then the control logic was applied to the servomotor, and as a result, controlled the bin lid.

The servomotor can control the rotation angle precisely. In this open and close application, if the servomotor rotates for 0 degree, in another word, stay still, the bin lid will also stay still, remains in a close situation. While the servomotor rotates for 180 degrees, the lever supports the lid will be pulled, thus the lid will open.

3.3.2 Voltage Converter

The ultra battery elimination circuit plays a role of voltage converter, which converts 12V from battery to the 5V needed for the other circuits. Usually the voltage provided by the battery will decay as the time goes. By using the ultra battery elimination circuit, we can provide the whole system a better rejection to battery voltage fluctuations.

3.3.3 Battery

The battery is a high discharge Li-Po battery. It has the maximum voltage of 11.1V (when charged to 3 cells), and the charge amounts to 5000mAh.

Chapter 4

Software Development

In this section, we introduce how we developed the software part in the robot with Arduino Mega in C language.

We used some machine learning technologies to train the model in order to find a mathematical model to help the robot distinguish the RGB value in output from the TCS3200 color sensor in order to let the robot open the trash cover when detects the yellow color, otherwise it keeps it closed.

On the automation control part, it can be divided into two major parts. The first one is obstacle sensing and distance measurement. On this part we should handle the 3-sonar logic in order to determine the object's direction and distance. The other part is the bin motion that should react according to the signal it receives, such as some turning and braking motions. For this part we use a PID to control the dynamics of the bin.

4.1 Obstacle Distance and Direction Sensing Using Sonars

In this section, we basically discuss the software realization of distance and direction detection using the ultrasonic sensors.

4.1.1 Trigger and Echo Method and Its Limitation

As it is mentioned in the previous hardware chapter, the sonar sensor measures the distance by triggering a signal and receiving its echoes. The time difference is translated to the distance. The trigger signal duration should be at least 10ms. In fact, the ultrasonic might be absorbed by some cloth material, thus the duration has to be prolonged in order to strengthen the trigger signal. This method is fine when only one sonar is used.

However, in our application, we need 3 sonar sensors. These 3 sonar sensors should not be triggered simultaneously, otherwise these 3 sonar sensors may have cross-talk to each other. The remaining choice is to trigger the sensors one after the other. This trail eliminates the cross-talk, but cascades the delay of the trigger and receiver. This delay will be 3 times larger than the single sonar sensor delay. Unfortunately, this results in the sampling rate reduced to 1/3 the previous rate. Since the bin dynamic is controlled by the feedback of the sonar sensors, the large delay will make the dynamics difficult to manage.

4.1.2 Sonar Sensor Detection Using NewPing Library

The NewPing library [11] can solve the dilemma mentioned above. This library have many features, in which the most important one is the use of timer interrupt method for event-driven sketches instead of using pulseIn. Other features such as apply digital filtering method and ease of using multiple sensors were also very helpful for us to design the sonar sensor array.

Before we applied the NewPing library, the detection delay was 60ms per sonar, therefore, 3 sonars needed 180ms, while, using the NewPing library, each iteration will only have 60ms delay. Therefore there is plenty of margin to control the bin base.

4.2 Dynamic Control of the Bin Base

The bin base dynamics can be divided into 2 parts: random walk before obstacle detected, obstacle approaching and post-detection fade away. The overall procedure is shown in the Figure 4.1.

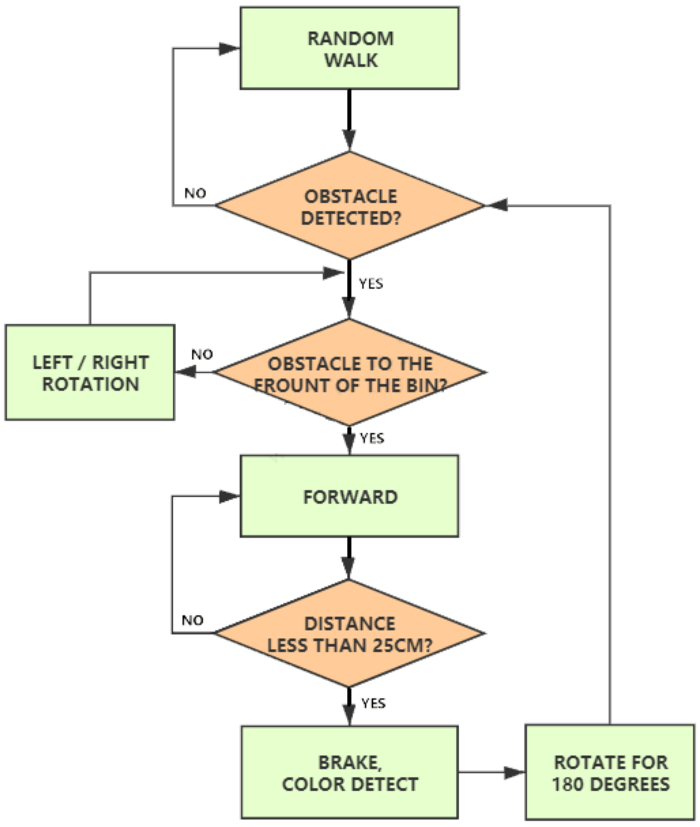


Figure 4.1: The Movement Procedure of the Bin

These dynamics are composed by elementary movements: forward(), left_rotation(), right_rotation() and brake().

4.2.1 Random Walk Movement

The robot's randomwalk() function can be achieved by taking the "Ranged Random Number"; a different number corresponds to different actions (like turn left, go forward and so on). The most critical part is "how to cast the dice". In order to generate a random number, the program should include a head file <math.h>. With this head file, random(max) function can be used to generate an int number between 0 and max [12].

With this random core, the random walk dynamics can be constructed by some case and switch structure.

4.2.2 Obstacle Approach and Avoid Movement

If there is an obstacle in the detection range of the sonar sensors, the bin will move towards the obstacle. The most important part is to adjust the ratio between forwarding and rotating. If the middle sonar reads the minimum distance, that means the obstacle is to the right front of the bin, then the bin will simply move forward. However, if the left most sonar reads the smallest distance, therefore, the bin should turn left, face the obstacle, and move forward. In this case, the rotation time is what we can apply a PID control.

The PID calculation function can be realized by:

```
void calculate_pid(){
    P = error;                          //propotional
    I = I + error;                      //integration
    D = error - previous_error;  //derivative
    PID_value = (Kp * P) + (Ki * I) + (Kd * D);
    previous_error = error;
}
```

The case of a left obstacle will enters in the branch is shown below:

```
else if((sonar_distance[0])< sonar_distance[1] &&
        sonar_distance[0] < sonar_distance[2] ){
    error = 70 - sonar_distance[0];
    brake();            // brake for a short period
    delay(25);          // of time to avoid inertia
    left_rotation();
    delay(10 + PID_value); // PID control on rotation time
    Serial.println("turn left");
}
```

When the distance between the bin and the obstacle is less then 25cm, the bin will brake and activate the color detection function. After the color detection the bin will right rotates for 180 degrees in order to fade away from the obstacle and start a new loop. The relative code is shown below:

```
if (sonar_distance[1] <= 25){
    brake();
    delay(3000);
```

```
        colordet();
        delay(4000);
        right_rotation();
        delay(1550);// the bin rotates 180 degrees
    }
```

4.3 Software Development of Color Sensor

In this section, we mention some software developments needed to make the color sensor work well, including how to measure the right RGB values and how to develop it in the Arduino code and the introduction which we need.

4.3.1 White Balance of TCS3200

As it mentioned before, the principle of the color sensor is counting the different color light pulse which reflect from the object, so if we want to measure the values of color red, green and blue, we need to count those three lights' pulse independently with different color filters through set the value of the S0 to S3. However, the count value is linearly related to the counting period which can not guarantee the measured value is in the range 0 to 255, what's more is that the three color filters have different sensitivities to the respective colors, so that, even using it to measure the corresponding pure color, the count will not be the same (considering the measure error); the responds of the filters is shown in Figure 4.2.

So we need to normalize the measure values and redress the index to make sure the filters have the same sensitivities to the different colors; this method is called White Balance. Before we start measure the object color value, we need to let the color sensor measure pure white object, get the measure values, then normalize the values, get the indexes of the different colors, through this way we can limit the measure values are in the range 0 to 255, also, make sure the measure values of the white object is (255,255,255), which is important for getting the right values of the measured object.

$$index[color] = 255/white[color]$$

$$Output[color] = measure[color] * index[color] \tag{4.1}$$

4.3.2 Software Development of Arduino

In the Arduino implementation of the color sensor, as we mentioned before, we needed a timer to set the sampling period, the TimerOne Library [14] uses the hardware Timer1 for finer PWM control and/or running a periodic interrupt function, in this project, we used the functions below:

```
Timer1.attachInterrupt(); /the function call when interrupt
Timer1.setPeriod(value); / set the interrupt period value
Timer1.initialize();     / initial the Timer
Timer1.stop();           / stop the timer
```

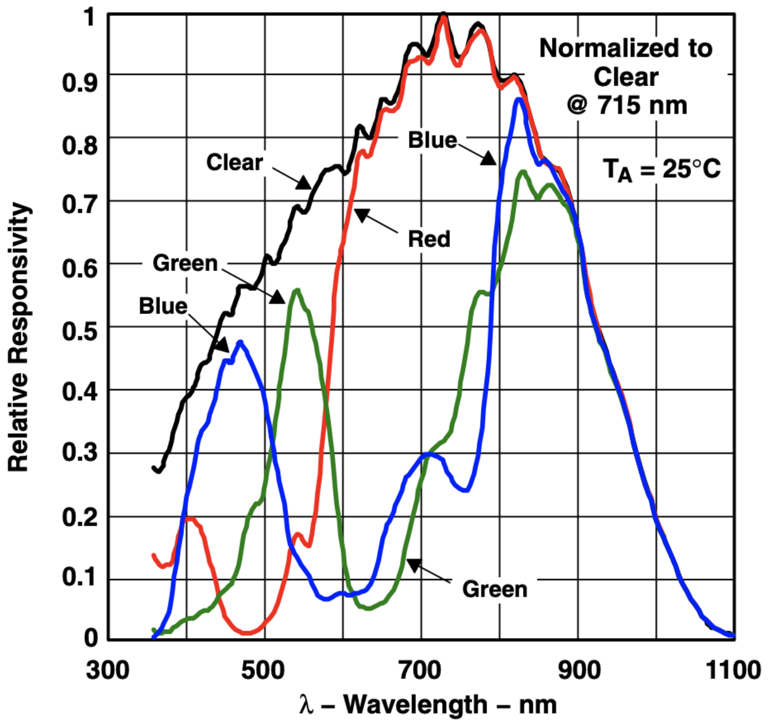


Figure 4.2: The Responds of the Different Colors

We need to set the color filters of the TCS3200 and count the color pulse using the code below:

```
switch(g_flag) {
    case 0:
        Serial.println("--->WB Start Filter");
        TSC_WB(LOW, LOW); // filter red active
        break;
    case 1:
        Serial.print("->Frequency R=");
        Serial.println(g_count); // print the red value.
        g_pulse_number_array[0] = g_count;
        // save the value into the array.
        TSC_WB(HIGH, HIGH); // filter green active.
        break;
    case 2:
        Serial.print("->Frequency G=");
        Serial.println(g_count);
        //print the value of green.
        g_pulse_number_array[1] = g_count;
        // save the green value. into the array
        TSC_WB(LOW, HIGH); // filter blue active
        break;
    case 3:
        Serial.print("->Frequency B=");
        Serial.println(g_count); // Print the blue value.
        Serial.println("--->WB End Filter");
        g_pulse_number_array[2] = g_count;
        // save the blue value into the array.
        TSC_WB(HIGH, LOW); // no filter active
        break;
        default:
        g_count = 0; // count set to 0
        break; } }
```

4.3.3 Direct Detection on Color Sensor

The first method is getting the number of the RGB color which output from the TCS3200 color sensor, then compare the difference between the output and the standard color region, if the number is in the region, the machine thinks that the color of the garbage bag is yellow and accept it.

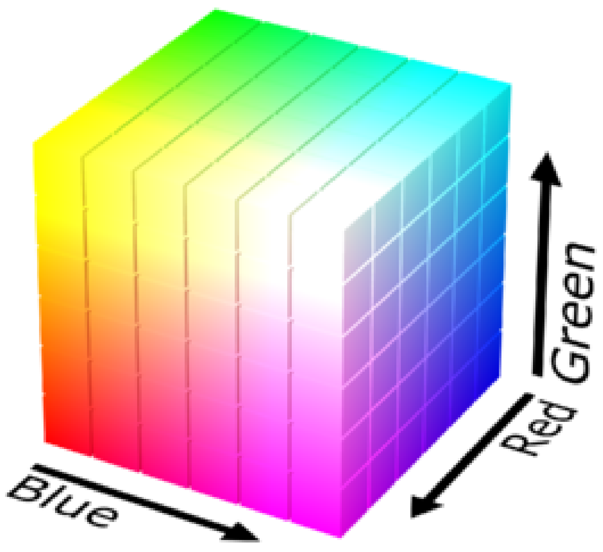
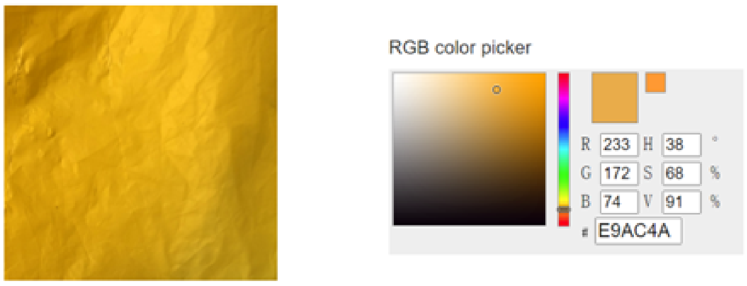


Figure 4.3: The 3D Model of RGB

Firstly, we create the 3D coordinates to represent the different color, then we find a region which probably represents yellow. For example, the region is: R: 200 255, G: 120 255, B: 0 30. After determining the "Yellow Region", what the machine do is checking the determining value is in the region or not, if the answer is yes, open the cover.

The problem is ensuring the region is so hard, the bound of the color is fuzzy, and different people has different sensitivity to the color, so the region is controversial and subjective. Also, in this region, not all the area represents the yellow, for example, RGB

= [200,255,0], it's obviously more similar to green. On the base of the previous way, we notice that the RGB number of the color in the region which is more next to the yellow all have a common characteristic: number of red greater than number of green, so we add a limitation: R value greater than G value. However, this way is not so precise, the RGB number we get in the figure below is [233,172,74] and Figure 4.4 shows the comparison between the standard color of the value and the object:



R 233 H 38 °
G 172 S 68 %
B 74 V 91 %
E9AC4A

Figure 4.4: The Comparison Between Standard Color the Object Color

We can achieve this function use the code below:

```
if R_L<=value_R<=R_H && G_L<=value_G<=G_H && B_L<=value_B<=B_H{
    /oopen the cover
}
else{
    /stand by for seconds and leave
}
```

4.3.4 Detection with Machine Learning

After collecting the data-set and we use some machine learning technologies to train some math models (with Python, code shows in Appendix.A) which can help the robot distinguish the color, different from judging the color directly, the three inputs RGB values will be put into the math models we train, through the sigmoid function then output the result, the pseudocode shows below:

```
float para1 = R*index_R1 + G*index_G1 + B*index_B1;
float para2 = exp(-para1);
float p1 = 1/(1+para2);


float para3 = R*index_R2 + G*index_G2 + B*index_B2;
float para4 = exp(-para3);
float p2 = 1/(1+para4);
if (p1 >= 0.5 && p2 >= 0.5){
    //open the garbage cover
}
else{
    // keep closed
}
```

4.4 Servomotor Control of the Bin Lid

In order to control the servomotor, a library <servo.h> [13] should be used in the code. Unfortunately, this library will disable the PWM output of pins 11-13. Therefore, it is better not to use these pins. However, in case use these pins as PWM output is inevitable, the function attach() and detach() could be use to enable and disable the servo library.

The control of the servomotor is quite easy. To open the bin lid, the servomotor rotates for 180 degrees. The corresponding code is shown below:

```
if (p1 >= 0.5 && p2 >= 0.5){    // color detected yellow
    Serial.print("lid open: ");
    Serial.println(1);
    mySerovo.attach(sermotor);  // attach the servomotor
    myServo.write(180);         // lid  open for 4 seconds
    delay (4000);
    myServo.write(0);           // close the bin lid
    delay (50);
    myServo.detach();           // detach, enable PWM output
}
```

Chapter 5

Results Obtained

5.1 Color Detector Math Models Results

We use three different methods (all machine learning technologies implemented with Python 3.6) to train the model and compare those results and accuracy, firstly we analyze the data and the data distribution shows in Figure 5.1.

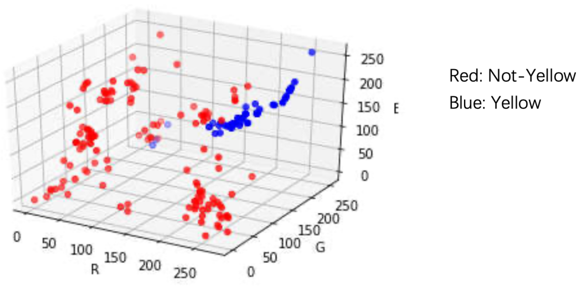


Figure 5.1: Data Distribution

5.1.1 Logistic Regression Model

We use Logistic Regression with SGD Algorithm to build the model, Also, with Bagging Algorithm to enhance the model, the final math model is :

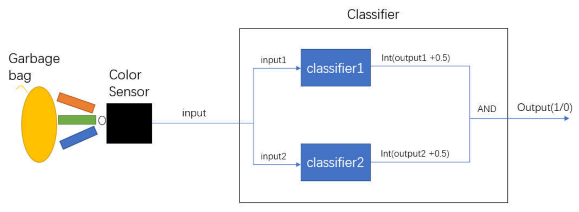
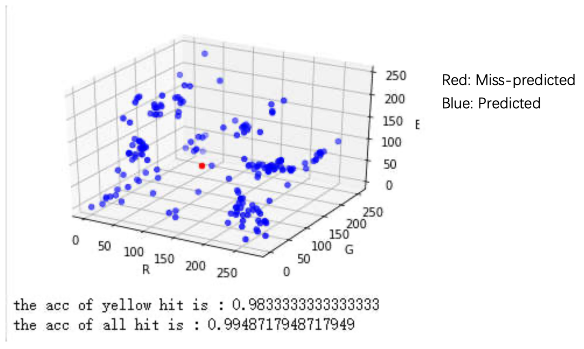


Figure 5.2: Logistic Regression Model

$$\begin{aligned} y &= round(P(haty_1 = +1|x))ANDround(P(haty_2 = +1|x)) \\ &= (round(\frac{1}{1+e^{-\omega_1^T x}}))AND(round(\frac{1}{1+e^{-\omega_2^T x}})) \end{aligned} \tag{5.1}$$

$$\begin{pmatrix} \omega_1 & \omega_2 \end{pmatrix} = \begin{pmatrix} 0.01808125 & -0.03512899 \\ 0.0070016 & 0.09511748 \\ -0.04289697 & -0.04180179 \end{pmatrix}$$

Then we evaluate the model and compute the accuracy of the model we finally get:



```
the acc of yellow hit is : 0.9833333333333333
the acc of all hit is : 0.9948717948717949
```

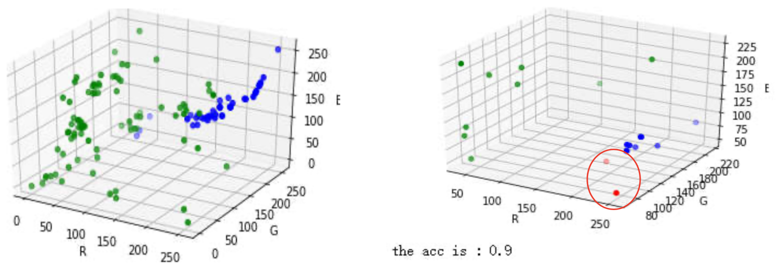
Figure 5.3: Evaluation of the Logistic Regression Model

Finally through the accuracy we can get the conclusion that the model is good enough for the machine to identify the color of the garbage bag is yellow or not in most cases. The model1 is not sensitive enough to separate the colors

Red: Not-Yellow Blue: Yellow,

Red: Miss-predicted Blue: Predicted yellow and red, but it is good enough to separate the other colors and yellow, the model2 is sensitive to distinguish red and yellow, so combining them, we get a nice classifier.

Figure 5.4 and Figure 5.5 are the two classifiers' train sets and test sets:



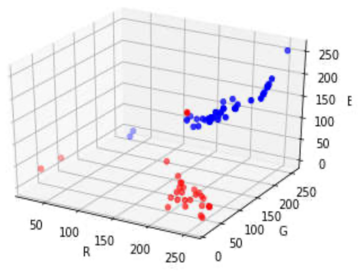
```
the acc is : 0.9
```

(a) Training Set (b) Test Set

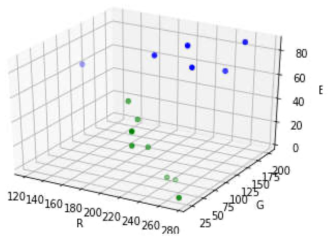
Figure 5.4: Classifier1's Training and Test Sets

5.1.2 Support Machine Model

We use the second method, support vector machine to build another model with the linear kernel, then get the math model:



(a) Training Set



```
the acc is : 1.0
```

(b) Test Set

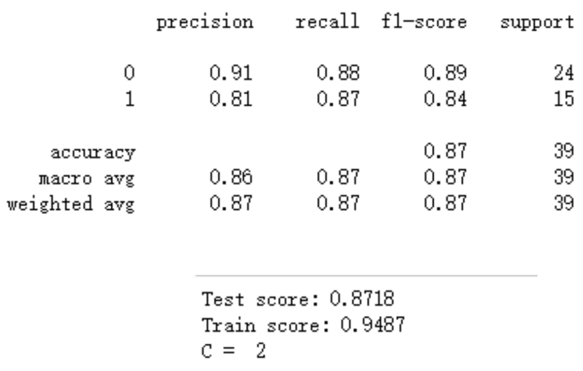
Figure 5.5: Classifier2's Training and Test Sets

$$y = sign(\omega^T x - b) \tag{5.2}$$

$$\omega = \begin{pmatrix} 0.02667914 & 0.04600077 & -0.01446091 \end{pmatrix}$$

$$b = -9.99444942$$

Finally, evaluate the model and get the result: the surface that separates the yellow points and not-yellow points with accuracy 87%.



```
              precision    recall  f1-score   support

           0       0.91      0.88      0.89        24
           1       0.81      0.87      0.84        15

    accuracy                           0.87        39
   macro avg       0.86      0.87      0.87        39
weighted avg       0.87      0.87      0.87        39
```

```
Test score: 0.8718
Train score: 0.9487
C =  2
```

Figure 5.6: SVM Model Evaluation 1

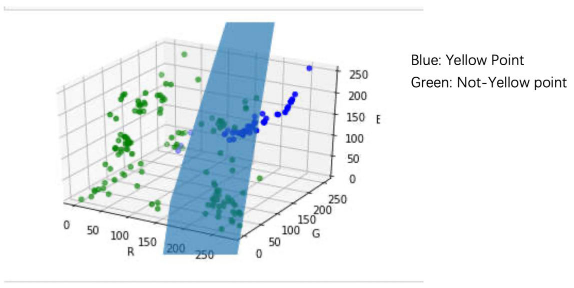


Figure 5.7: SVM Model Evaluation 2

5.1.3 Results of The Models

we use the presented methods to predict the color of the RGB value which output by the TCS3200 color sensor taken samples from the object. As we explained before, the direct detection is not a good method because it's fuzzy and not precise enough: methods with machine learning technologies are better. After comparing the accuracy of machine learning models with different algorithms, also considering the difficulty of implementing the model in the machine, we finally choose the logistic regression model with bagging algorithm as the final model used by the robot.

5.2 Hardware Mounting and Dynamic Control Result

According to the hardware pins and software definitions, the wires connections are shown in Figure 5.8

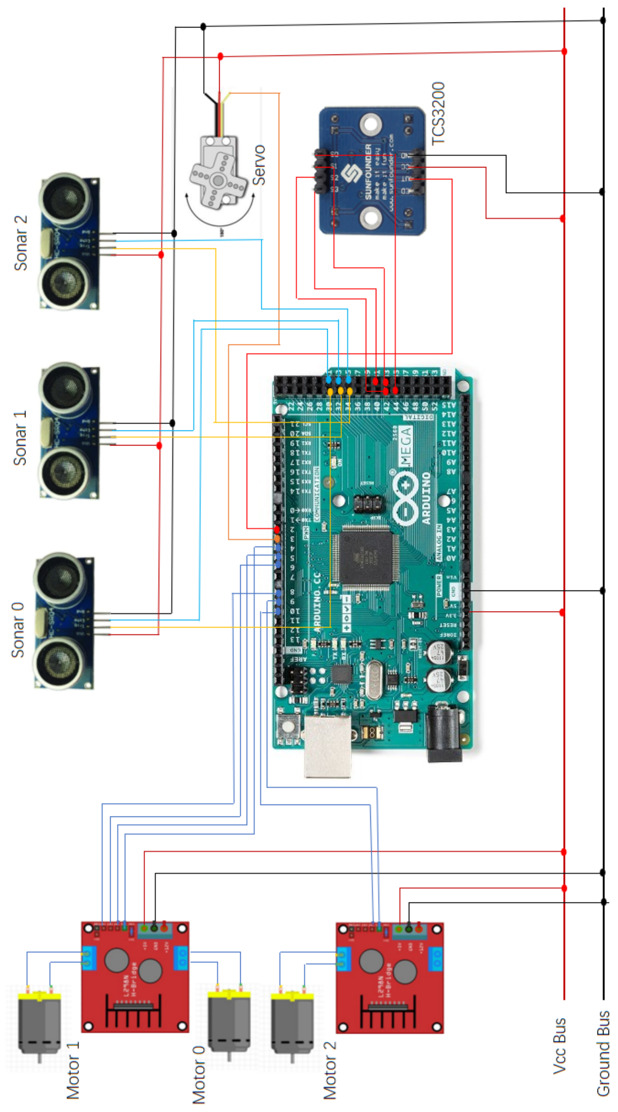


Figure 5.8: The Wires Connection of the Robot

In reality, it took us a lot effort to mount all the components on the bin. Since this project is a rubbish bin, we had to save some space in the bin for the rubbish, therefore we had to put most of the modules beneath the bin base. Because of this, the fixing of the modules has to be very stable, or the modules will risk to hit the ground. On the other hand, use screw method might make some chip back touch the metal bin base, risk in potential short circuit. We apply some thin plastic sponge film to insulate the chips and the metal bin base, then stick the modular to the plastic sponge. The wires also have to be fasten carefully in order to not touch the ground. Considering that the color sensor should be used in a dark environment, the sensor part should not face up, otherwise the cylindrical cover will filter well the ambient light. We drilled some holes in order to screw the color sensor on the bin surface, so that the photosensitive part is covered by the cylindrical part.

After having fixed all the parts, the overall robot bin is shown in Figure 5.9.

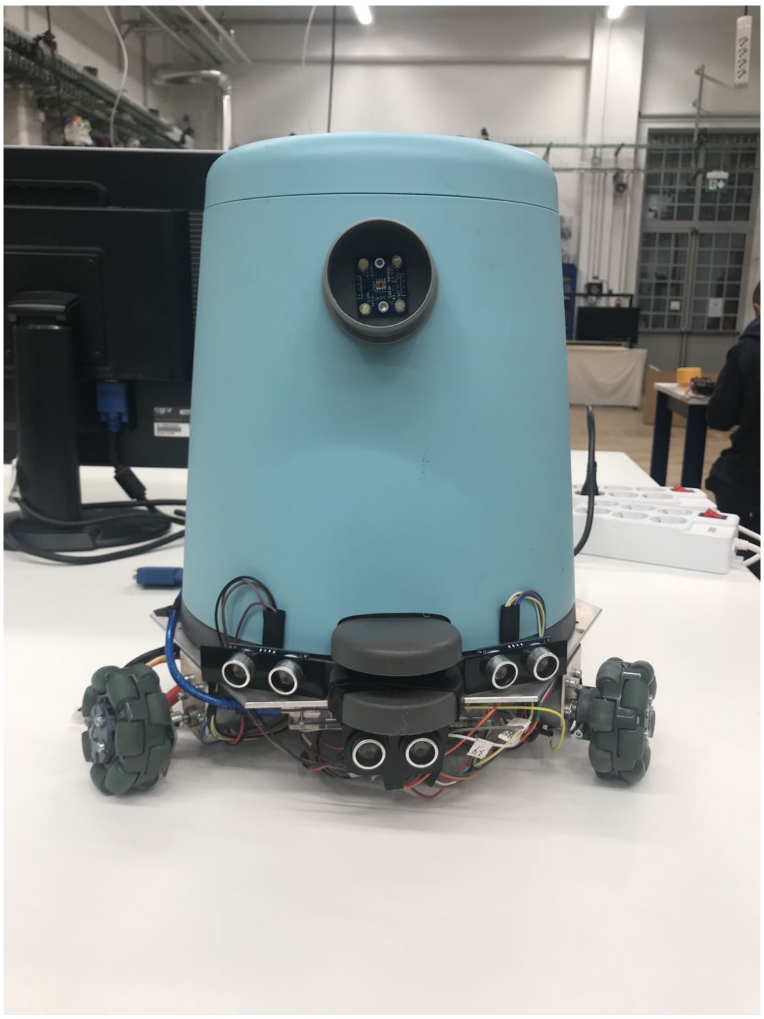


Figure 5.9: The Realization of the Robot.

5.2.1 Dynamic Control Result

The result for the dynamic control can be divided into the following parts: sonar sensor sampling rate adjustment, PID control on the rotation time, servomotor reaction with respect to color sensor feedback.

Basically, we tested the bin functions according to a complete procedure.

Initially, if nothing is in the detecting range of the sonar sensors, the bin will move randomly. During our test we find that, shuffles and move backward may cause some collision, because the bin is lack of left, right and back detection. Therefore, in our case, we exclude the shuffles and back ward movement in random walk function.

Once the user appears in the detecting range, the bin will adjust it's trail and move towards the user. We adjust the sampling rate of the sonar sensors in order to obtain the best behavior. The sonar sensors should not refresh too fast. If we apply the sampling rate too fast, firstly it would be not very meaningful since the obstacle may not change its position within a very short period of time, say, 30ms. Moreover, if the sonar sensors become very sensitive, even a slight perturbation will cause some oscillation of the bin, make the bin act "shaking" behavior frequently.

The PID values applied to the rotation also works well in our result, in most of cases, after at most 3 rotations, the bin can face the user directly.

The brake function to avoid collision is also well performed, maintaining a 25cm comfort zone.

Last and the most important function, the color recognition and lid open function. If we show the color sensor a blue bag, which is used to pack food residues in Italy, the bin lid rest closed, denies the rubbish bag. In another trial, we show the color sensor a yellow bag, which is used to pack plastic and tins in Italy, the bin lid is open, enable the use to put the bag in. This result is shown in Figures 5.10 and 5.11, the final robot movement steps are shown in Figure 5.12.

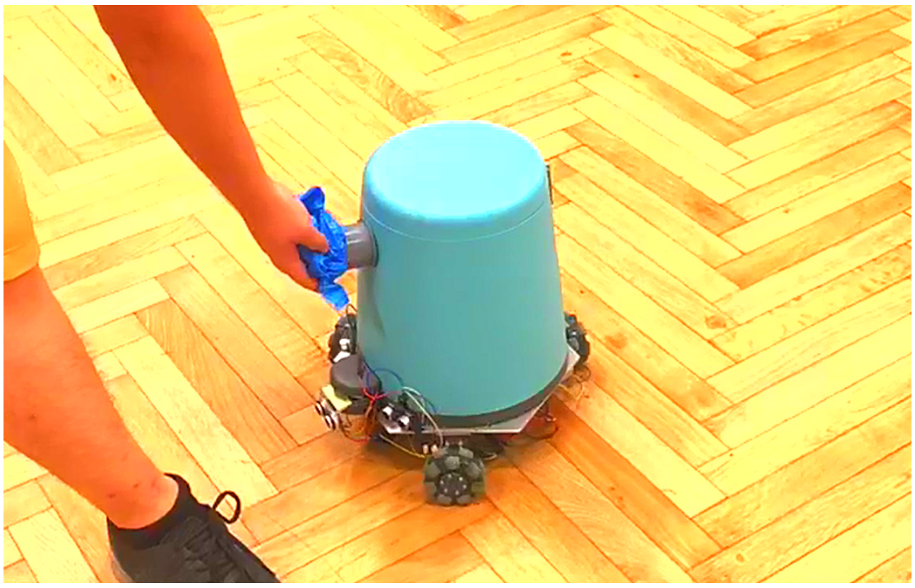


Figure 5.10: Showing a Blue Bag, Lid Close

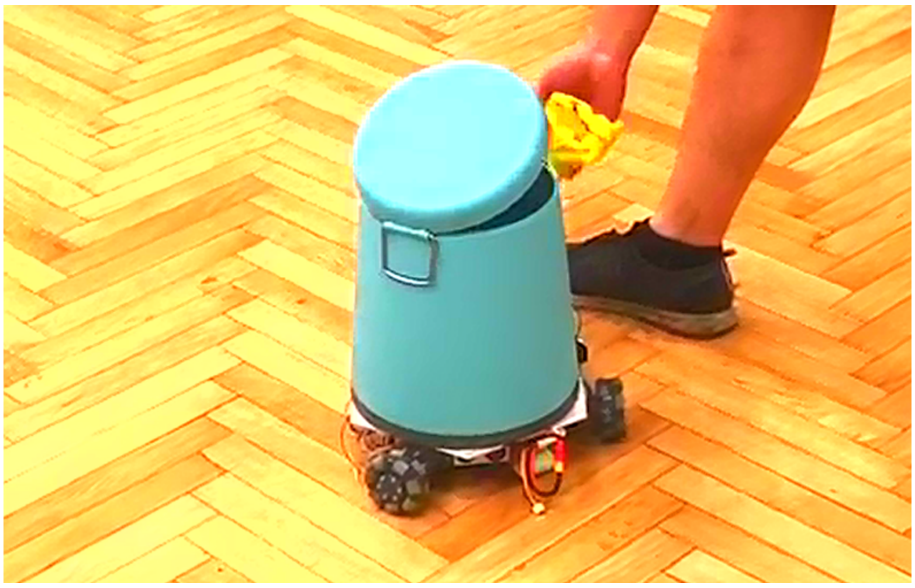


Figure 5.11: Showing a Yellow Bag, Lid Open

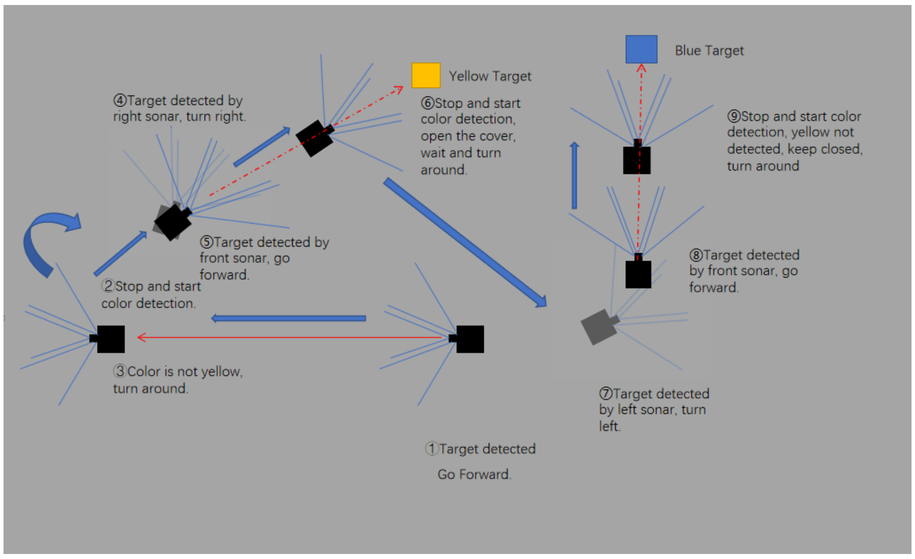

Yellow Target
Blue Target
④Target detected by right sonar, turn right.
⑥Stop and start color detection, open the cover, wait and turn around.
⑨Stop and start color detection, yellow not detected, keep closed, turn around
⑤Target detected by front sonar, go forward.
②Stop and start color detection.
⑧Target detected by front sonar, go forward.
③Color is not yellow, turn around.
⑦Target detected by left sonar, turn left.
①Target detected
Go Forward.

Figure 5.12: The Robot's Movement in Reality

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In this thesis, based on the situation we presented in the introduction section, we designed and implemented an automatic robot which can identify the types of garbage through the color of the garbage bag (only yellow garbage accepted) and has an automatic cruise function in a room (including obstacle detection, approaching and avoid).

The robot can pick the target's color value (RGB value) through TCS3200 color sensor, we use the color sensor to collect hundreds of data as the data set to train a machine learning math model. We use different methods and compare the accuracy of different models and pick the most efficient one as our final model; the final accuracy is about 98 percent, obtained by logistic regression and bagging algorithm. With the help of the model, combining the RGB values the robot read from the target, the Arduino controls the servomotors, open the lid if the garbage bag is yellow, otherwise keeps the lid closed and moves away.

There are three sonars on the robot facing different directions; each one can detect the distance between the obstacles and the bin. If there's no obstacle detected in the detecting range, it means there is a wide open area in front of the bin, the robot walks randomly forward and rotating. Once an obstacle appears in the detection range, the robot will approach the potential user. The robot goes forward if the front distance detected by all the sonars is the minimum, otherwise the robot will turn to the direction which has the smallest distance in order to face the obstacle. When the front distance to the obstacle is smaller or equal to the threshold distance (we set this as 25cm), the robot will stop and start the color detection. After the color detection finished, no matter the bin accept or deny the rubbish, the bin will turn around and start another automatic cruise function, random walk or approach another obstacle.

6.2 Future Works

6.2.1 Multi-sensor Distance Detection

Different detectors have different detecting range and different resolutions. Using multiple sensors will compensate the drawbacks of the other sensors.

The idea of using sonar sensors to measure the obstacle distance and position can be improved by using another type of sensor. As it is clear from the detection principle, the echo must be received before the next trigger signal is sent out. Therefore, a delay must be present, the distance measuring is digitized since the refresh rate is low. This is not a big problem if the robot moves at a slow speed. However, once the robot speed is high, the distance difference of two neighbouring measurement will be quite large and this makes the measured distance less meaningful. An alternative way to measure distance is to use an infrared range sensor.

An IR sensor [15] measures the distance using another principle. Unlike the sonar detector using the time of flight, IR sensor calculate the ratio between the luminosity amplitudes sent out and received. Figure 6.1 gives an intuitive explanation.

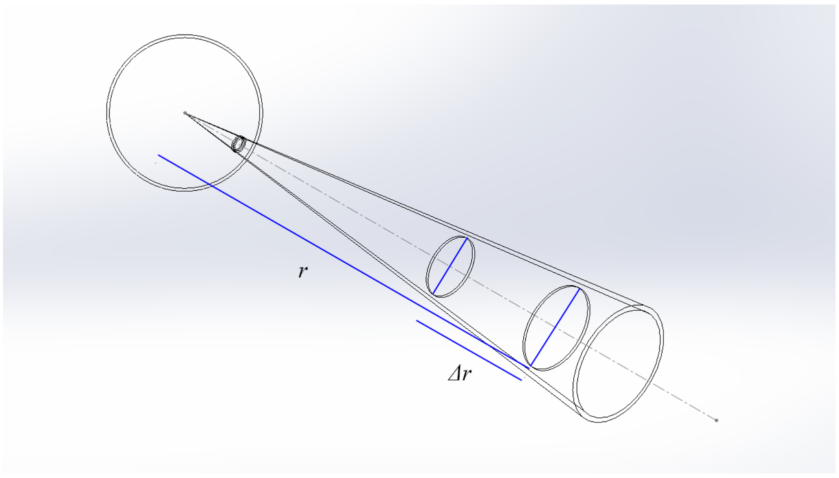


Figure 6.1: IR Sensor Signal Intensity With Respect to Distance

As a result, the receiver signal is proportional to the inverse of the distance.

In a particular case, we can get some sensor parameters from the sensor table.

6.2.2 Turning Smart Phone into Robots [16] [17]

What is even better than apply multiple different sensors? Using smart phones. In fact, we noticed that recently the Intel labs use the smart phone as the modular of their robot. The camera can be used to sense the image signals. Smart phone also create possibilities of remote control via Bluetooth or WLAN.

The Figure 6.2 shows the conceptual Intel OpenBot design.

The Intel lab use this OpenBot to train a driving situation. Both user controlling via wireless communication, and the automatic driving by its inner algorithm. The communication and the driving training are shown in Figure 6.3 and Figure 6.4.

6.2.3 Garbage Type Detection Technologies

In this project, the robot can only identifies and accepts the yellow garbage bag. To be further improved, we can use the the multi-categories method and multi-garbage-bin model to identify more types of garbage. Different colors of the garbage bags is corresponding to different garbage bins.

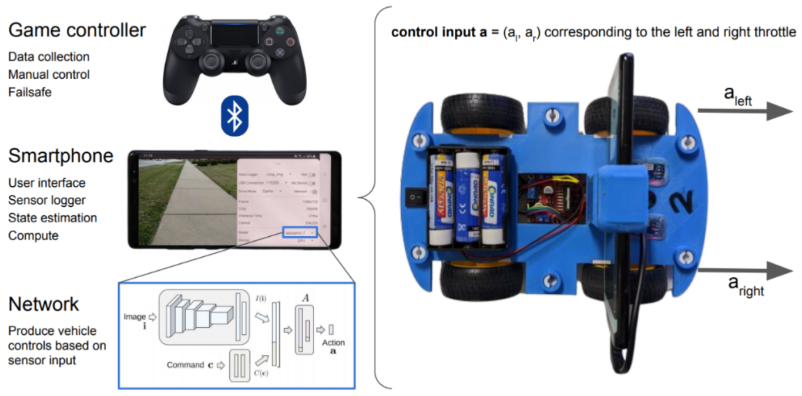


Figure 6.2: Intel OpenBot Design Idea.

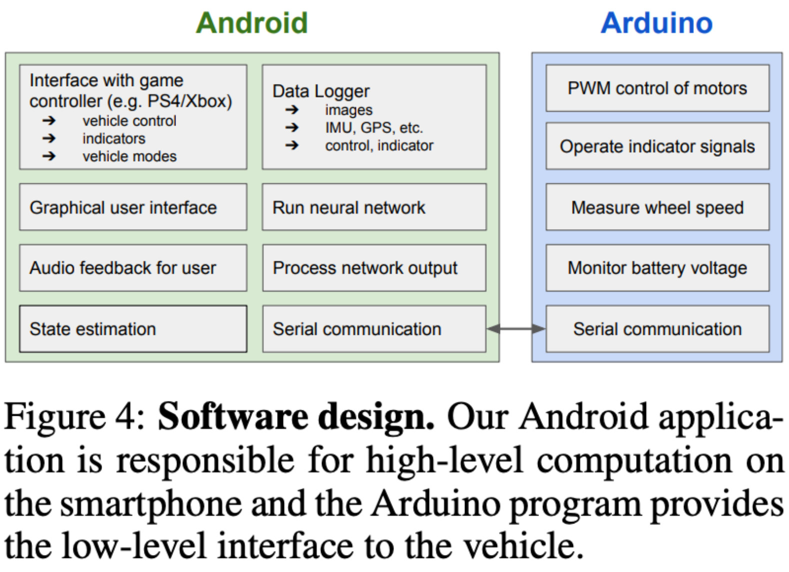


Figure 4: **Software design.** Our Android application is responsible for high-level computation on the smartphone and the Arduino program provides the low-level interface to the vehicle.

Figure 6.3: Intel OpenBot Communication Software Design.

Obviously, using color sensor to detect the garbage type is not the best way. The result correctness is highly depends on the correctness people putting the garbage in the appropriate colored garbage bag. In the future work, we hope the robot can achieve the function of garbage classification and processing through image processing [18]:

- 1. The camera on the robot takes the picture of the garbage.
- 2. Extract image edges, obtain the edge characteristics of the garbage to be discarded.
- 3. Use machine learning technologies like convolutional neural network to achieve the purpose of identifying garbage types.
- 4. Finish classification and output the result.

Here is an example for garbage classification with the picture process: OSCAR created by Intuitive AI. An introduction video [19].

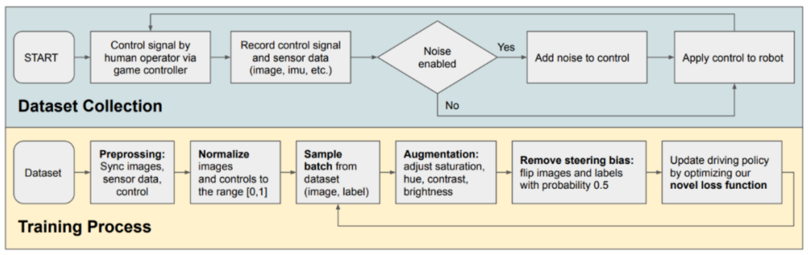

START
Control signal by human operator via game controller
Record control signal and sensor data (image, imu, etc.)
Noise enabled
Yes
No
Add noise to control
Apply control to robot
Dataset Collection
Dataset
Preprossing: Sync images, sensor data, control
Normalize images and controls to the range [0,1]
Sample batch from dataset (image, label)
Augmentation: adjust saturation, hue, contrast, brightness
Remove steering bias: flip images and labels with probability 0.5
Update driving policy by optimizing our novel loss function
Training Process

Figure 6: **Driving policy: Training pipeline.** The flowchart explains the complete process for obtaining our autonomous navigation policy. There are two main components, dataset collection and training the driving policy which is represented by a neural network.

Figure 6.4: Intel OpenBot Driving Training.

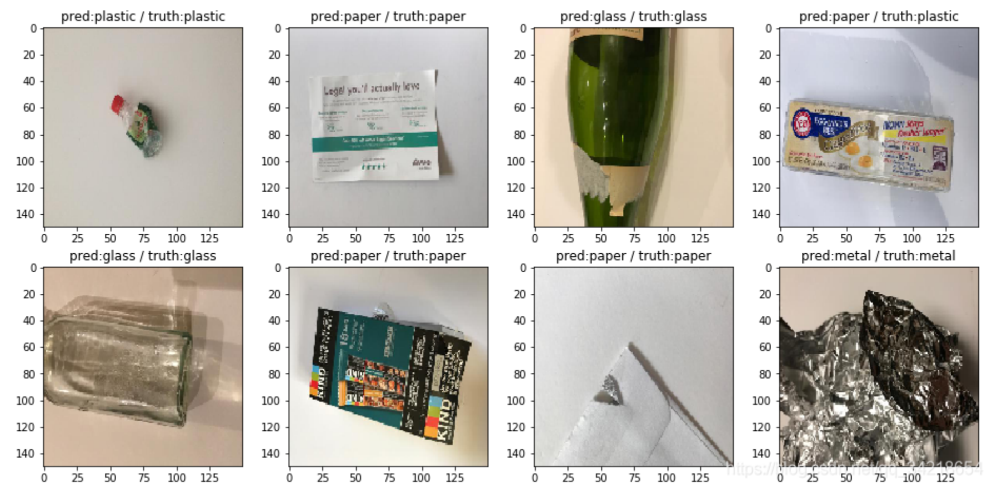


Figure 6.5: Garbage Classification with CNN

More details on Intuitive AI website:https://intuitiveai.ca/

Bibliography

[1] Italy waste sorting [EB/OL]. https://it.wikipedia.org/wiki/Raccolta_differenziata

[2] Shanghai waste sorting (2019) [EB/OL]. http://www.xinhuanet.com/english/2019-07/03/c_138195992.htm

[3] Pyle D. Data preparation for data mining[M]. morgan kaufmann, 1999.

[4] Strobl C, Malley J, Tutz G. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests[J]. Psychological methods, 2009, 14(4): 323.

[5] Ruder S. An overview of gradient descent optimization algorithms[J]. arXiv preprint arXiv:1609.04747, 2016.

[6] Breiman L. Bagging predictors[J]. Machine learning, 1996, 24(2): 123-140.

[7] Suykens J A K, Vandewalle J. Least squares support vector machine classifiers[J]. Neural processing letters, 1999, 9(3): 293-300.

[8] L298N maunal[EB/OL]. https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

[9] TCS3200 manual[EB/OL]. https://www.mouser.com/catalog/specsheets/tcs3200-e11.pdf.

[10] Arduino datasheet[EB/OL].http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

[11] NewPing library[EB/OL]. https://playground.arduino.cc/Code/NewPing/

[12] Random() function[EB/OL]. https://www.arduino.cc/reference/en/language/functions/random-numbers/random/

[13] Servo Library[EB/OL]. https://www.arduino.cc/reference/en/libraries/servo/

[14] TimerOne Library[EB/OL]. https://www.arduino.cc/reference/en/libraries/timerone/.

[15] IR sensor[EB/OL]. https://www.electronicshub.org/ir-sensor/

[16] Intel Müller M, Koltun V. OpenBot: Turning Smartphones into Robots[J]. arXiv preprint arXiv:2008.10631, 2020.

[17] Intel OpenBot video[EB/OL]. https://www.youtube.com/watch?v=qc8hFLyWDOM

[18] Perez-Munuzuri V, Perez-Villar V, Chua L O. Autowaves for image processing on a two-dimensional CNN array of excitable nonlinear circuits: flat and wrinkled labyrinths[J]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 1993, 40(3): 174-181.

[19] Introduction video of OSCAR[EB/OL]. https://www.youtube.com/watch?v=Sc2hdmwGgnU.

Appendix A

Codes of The Color Sensor with Machine Learning

A.1 Codes of Logistic Regression with Bagging Algorithm

```
import pandas as pd
import statsmodels.api as sm
import pylab as pl
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')

train_df.head()

list=train_df.values.tolist()

from mpl_toolkits.mplot3d import axes3d
from matplotlib import style

fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')


x_0,y_0,z_0,x_1,y_1,z_1 = [],[],[],[],[],[]
for val in list:
    if int(val[-1]) == 0:
        x_0.append(val[0])
        y_0.append(val[1])
        z_0.append(val[2])
    else:
        x_1.append(val[0])
        y_1.append(val[1])
        z_1.append(val[1])
ax1.scatter(x_0, y_0, z_0, c='g', marker='o',label='Not_yellow')
ax1.scatter(x_1, y_1, z_1, c='b', marker='o',label='Yellow')
ax1.set_xlabel('R')
ax1.set_ylabel('G')
ax1.set_zlabel('B')

plt.show()

train_df.hist()

pl.show()

train_cols = train_df.columns[:3]
logit = sm.Logit(train_df['Type'], train_df[train_cols])
result=logit.fit()
result_para = logit.fit().params.values
print (result_para)

fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')
test_df['predict'] = result.predict(test_df[train_cols])
count = 0
hit = 0
x_hit_0,y_hit_0,z_hit_0,x,y,z,x_hit_1,y_hit_1,z_hit_1 = [],[],[],[],[],[],[],[


for value in test_df.values:
    count += 1
    res = value[-1]
    if res >= 0.5:
        value[-1] = 1
    if res < 0.5:
        value[-1] = 0
    if value[-1] == int(value[3]):
        hit += 1
        if value[-1] == 1:
            x_hit_1.append(int(value[0]))
            y_hit_1.append(int(value[1]))
            z_hit_1.append(int(value[2]))
        else:
            x_hit_0.append(int(value[0]))
            y_hit_0.append(int(value[1]))
            z_hit_0.append(int(value[2]))
    if value[-1] != int(value[3]):
        x.append(int(value[0]))
        y.append(int(value[1]))
        z.append(int(value[2]))

ax1.scatter(x_hit_0, y_hit_0, z_hit_0, c='g', marker='o',label='Not_yellow_rig
ax1.scatter(x_hit_1, y_hit_1, z_hit_1, c='b', marker='o',label='Yellow_right')
ax1.scatter(x, y, z, c ='r', marker='o',label='misprediction')

ax1.set_xlabel('R')
ax1.set_ylabel('G')
ax1.set_zlabel('B')

plt.show()
print ("the acc is :",hit/count)

train_df2 = pd.read_csv('train_p2.csv')
test_df2 = pd.read_csv('test_p2.csv')

list2=train_df2.values.tolist()

from mpl_toolkits.mplot3d import axes3d
from matplotlib import style

fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')


x_0,y_0,z_0,x_1,y_1,z_1 = [],[],[],[],[],[]
for val in list2:
    if int(val[-1]) == 0:
        x_0.append(val[0])
        y_0.append(val[1])
        z_0.append(val[2])
    else:
        x_1.append(val[0])
        y_1.append(val[1])
        z_1.append(val[1])
ax1.scatter(x_0, y_0, z_0, c='r', marker='o',label='Not_yellow')
ax1.scatter(x_1, y_1, z_1, c='b', marker='o',label='Yellow')
ax1.set_xlabel('R')
ax1.set_ylabel('G')
ax1.set_zlabel('B')
```

```
plt.show()

train_df2.hist()

pl.show()

train_cols2 = train_df2.columns[:3]
logit2 = sm.Logit(train_df2['Type'], train_df2[train_cols2])
result2=logit2.fit()
result_para2 = logit2.fit().params.values
print (result_para2)

fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')
test_df2['predict '] = result2.predict(test_df2[train_cols2])
count = 0
hit = 0
x_hit_0,y_hit_0,z_hit_0,x,y,z,x_hit_1,y_hit_1,z_hit_1 = [],[],[],[],[],[],[],[


for value in test_df2.values:
    count += 1
    res = value[-1]
    if res >= 0.5:
        value[-1] = 1
    if res < 0.5:
        value[-1] = 0
    if value[-1] == int(value[3]):
        hit += 1
        if value[-1] == 1:
            x_hit_1.append(int(value[0]))
            y_hit_1.append(int(value[1]))
            z_hit_1.append(int(value[2]))
        else:
            x_hit_0.append(int(value[0]))
            y_hit_0.append(int(value[1]))
            z_hit_0.append(int(value[2]))
    if value[-1] != int(value[3]):
        x.append(int(value[0]))
        y.append(int(value[1]))
        z.append(int(value[2]))

ax1.scatter(x_hit_0, y_hit_0, z_hit_0, c='g', marker='o',label='Not_yellow_rig
ax1.scatter(x_hit_1, y_hit_1, z_hit_1, c='b', marker='o',label='Yellow_right')
ax1.scatter(x, y, z, c ='r', marker='o',label='misprediction')
```

```
ax1.set_xlabel('R')
ax1.set_ylabel('G')
ax1.set_zlabel('B')

plt.show()
print ("the acc is :",hit/count)

data = pd.read_csv('data(p12).csv')

listall=data.values.tolist()
fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')


x_0,y_0,z_0,x_1,y_1,z_1 = [],[],[],[],[],[]
for val in listall:
    if int(val[-1]) == 0:
        x_0.append(val[0])
        y_0.append(val[1])
        z_0.append(val[2])
    else:
        x_1.append(val[0])
        y_1.append(val[1])
        z_1.append(val[1])
ax1.scatter(x_0, y_0, z_0, c='r', marker='o',label='Not_yellow')
ax1.scatter(x_1, y_1, z_1, c='b', marker='o',label='Yellow')
ax1.set_xlabel('R')
ax1.set_ylabel('G')
ax1.set_zlabel('B')

plt.show()

data.hist()
pl.show()

data['predict1'] = result.predict(data[train_cols2])
data['predict2'] = result2.predict(data[train_cols2])
count = 0
count_yellow = 0
hit = 0
hit_yellow = 0

listall=data.values.tolist()
fig = plt.figure()
```

```
ax1 = fig.add_subplot(111, projection='3d')

x_1,y_1,z_1,x_0,y_0,z_0 = [],[],[],[],[],[]

for value in data.values:
    count += 1
    if int(value[3]) == 1:
        count_yellow += 1
        if value[-1] >= 0.5 and value[-2] >= 0.5:
            hit += 1
            hit_yellow += 1

            x_1.append(int(value[0]))
            y_1.append(int(value[1]))
            z_1.append(int(value[2]))
        else:
            x_0.append(int(value[0]))
            y_0.append(int(value[1]))
            z_0.append(int(value[2]))
    else:
        hit += 1
        x_1.append(int(value[0]))
        y_1.append(int(value[1]))
        z_1.append(int(value[2]))

ax1.scatter(x_0, y_0, z_0, c='r', marker='o',label='Right_pred')
ax1.scatter(x_1, y_1, z_1, c='b', marker='o',label='Miss_pred')
ax1.set_xlabel('R')
ax1.set_ylabel('G')
ax1.set_zlabel('B')

plt.show()

print ("the acc of yellow hit is :",hit_yellow/count_yellow)
print ("the acc of all hit is :",hit/count)
```

A.2 Codes of Support Vector Machine

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

data = pd.read_csv('data(p12).csv')
data.head()

x = data.drop('Type', axis=1)
y = data['Type']

x.head()

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20)

from sklearn.svm import SVC
svclassifier = SVC(kernel='linear',C = 2)
svclassifier.fit(x_train, y_train)

print('Test score: %.4f' % svclassifier.score(x_test, y_test))
print('Train score: %.4f' % svclassifier.score(x_train, y_train))
print ('C = ',2)

y_pred = svclassifier.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))

n_Support_vector = svclassifier.n_support_
sv_idx = svclassifier.support_
w = svclassifier.coef_
b = svclassifier.intercept_

from mpl_toolkits.mplot3d import axes3d
from matplotlib import style

list=data.values.tolist()
x_0,y_0,z_0,x_1,y_1,z_1 = [],[],[],[],[],[]
for val in list:
    if int(val[-1]) == 0:
        x_0.append(val[0])
        y_0.append(val[1])
        z_0.append(val[2])
    else:
        x_1.append(val[0])
        y_1.append(val[1])
        z_1.append(val[1])

ax = plt.subplot(111, projection='3d')
x = np.arange(0,255,1)
y = np.arange(0,255,1)
x, y = np.meshgrid(x, y)
z = (w[0,0]*x + w[0,1]*y + b) / (-w[0,2])
surf = ax.plot_surface(x, y, z, rstride=1, cstride=1)
ax.set_zlim(0,255)

ax.scatter(x_0, y_0, z_0, c='g', marker='o',label='Not_yellow')
ax.scatter(x_1, y_1, z_1, c='b', marker='o',label='Yellow')
ax.set_xlabel('R')
ax.set_ylabel('G')
ax.set_zlabel('B')

plt.show()

print ('the para w is:',w)
print ('the para b is:',b)
```

Appendix B

Code of Arduino

```
/*library mandatory for color sensor*/
#include <TimerOne.h>

/*library mandatory for random walk*/
#include <math.h>

/*library mandatory for servomotor*/
#include <Servo.h>
Servo myServo;

/*libaray mandatory for sonar sensors*/
#include <NewPing.h>

/*pin definition and variable initialization*/

#define S0    44
#define S1    43
#define S2    42
#define S3    41
#define OUT    2

float g_SF[3];
int   g_count = 0;
int   g_array[3];
int   g_flag = 0;

int sonar_trig0 = 30;
// define left ultrasonic signal receiver pin ECHO to 30.
int sonar_echo0 = 31;
// define left ultrasonic signal transmitter pin TRIG to 31.
int sonar_trig1 = 32;
int sonar_echo1 = 33;
int sonar_trig2 = 34;
int sonar_echo2 = 35;

long int sonar_distance[3] = {150,150,150};
int detection_range = 150;

NewPing sonar[3] = {   // Sensor object array.
  NewPing(sonar_trig0, sonar_echo0, detection_range),
  // Each sensor's trigger pin, echo pin, and max distance to ping.
  NewPing(sonar_trig1, sonar_echo1, detection_range),
  NewPing(sonar_trig2, sonar_echo2, detection_range)
};

int motor0p = 4;
int motor0n = 5;
int motor1p = 6;
int motor1n = 8;
int motor2p = 9;
int motor2n = 10;

int sermotor = 3;

float Kp = 3, Ki = 0.05, Kd = 0.01;
float P = 0, I = 0, D = 0;
float PID_value = 0;
float error = 0;
float previous_error = 0;

void setup(){
  Serial.begin(115200);
  pinMode(sonar_echo0, INPUT);
  pinMode(sonar_trig0, OUTPUT);
  pinMode(sonar_echo1, INPUT);
  pinMode(sonar_trig1, OUTPUT);
  pinMode(sonar_echo2, INPUT);
  pinMode(sonar_trig2, OUTPUT);

  pinMode(motor0p,OUTPUT);
  pinMode(motor0n,OUTPUT);
  pinMode(motor1p,OUTPUT);
  pinMode(motor1n,OUTPUT);
  pinMode(motor2p,OUTPUT);
  pinMode(motor2n,OUTPUT);

  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(OUT, INPUT);

  myServo.attach(sermotor);

  TSC_Init();
  Timer1.initialize();    // defaulte is 1s
  Timer1.attachInterrupt(TSC_Callback);

  attachInterrupt(0, TSC_Count, RISING);

  delay(2000);
}

void loop(){
  while(1){
    read_sonar_values();
    calculate_pid();
    motor_control();
  }
}

void read_sonar_values(){
  delay(60);
  // Wait 60ms between pings (about 15 samples/sec).
  Serial.println("Sonar values");
  sonar_distance[0] = sonar[0].ping_cm();
  if (sonar_distance[0] == 0){
    sonar_distance[0] = detection_range;
  }
  sonar_distance[1] = sonar[1].ping_cm();
  if (sonar_distance[1] == 0){
    sonar_distance[1] = detection_range;
  }
  sonar_distance[2] = sonar[2].ping_cm();
  if (sonar_distance[2] == 0){
    sonar_distance[2] = detection_range;
  }
  /* Send ping, get distance in cm and print result*/
  Serial.println(sonar_distance[0]);
  Serial.println(sonar_distance[1]);
  Serial.println(sonar_distance[2]);
  Serial.println("=============");
```

```
}

void motor_control(){
  if (sonar_distance[1] <= 25){
    brake();
    delay(3000);
    colordet();
    delay(4000);
    right_rotation();
    delay(1550);
  }

  if(sonar_distance[0] > 70 && sonar_distance[1] > 70 &&
     sonar_distance[2] > 70){
    randomwalk(); // random walk if wide open
    error = 0;
  }

  else if((sonar_distance[0])< sonar_distance[1] &&
          sonar_distance[0] < sonar_distance[2] ){
    error = 70 - sonar_distance[0];
    brake();
    delay(25);
    left_rotation();
    delay(10 + PID_value);
    Serial.println("turn left");
  }

  else if(sonar_distance[1] < sonar_distance[0] &&
         sonar_distance[1] < sonar_distance[2] ){
    error = 0;
    forward();
    delay(100);
  }

  else if(sonar_distance[2] < sonar_distance[1] &&
         sonar_distance[2] < sonar_distance[0] ){
    error = sonar_distance[2] - 70;
    brake();
    delay(25);
    right_rotation();
    delay(10 - PID_value);
    Serial.println("turn right");
  }
}
```

```
void randomwalk(){
  int random_operation = random(10);
  if (random_operation == 0){
    forward();
    delay(900);
  }
  if (random_operation == 1){
    backward();
    delay(600);
  }
  if (random_operation == 2){
    left_rotation();
    delay(400);
  }
  if (random_operation == 3){
    right_rotation();
    delay(400);
  }
  if (random_operation == 4){
    left_rotation();
    delay(500);
  }
  if (random_operation == 5){
    right_rotation();
    delay(500);
  }
  if (random_operation > 5){
    forward();
    delay(1000);
  }
  Serial.print("random walk: ");
  Serial.println(random_operation);
}

/*dynamics, motor1 seems more powerful
  than the other two, thus less PWM value*/
void left_rotation(){
  analogWrite(motor0p,100);
  digitalWrite(motor0n,LOW);
  analogWrite(motor1p,70);
  digitalWrite(motor1n,LOW);
  analogWrite(motor2p,75);
  digitalWrite(motor2n,LOW);
}
```

```
void right_rotation(){
  analogWrite(motor0n,100);
  digitalWrite(motor0p,LOW);
  analogWrite(motor1n,70);
  digitalWrite(motor1p,LOW);
  analogWrite(motor2n,75);
  digitalWrite(motor2p,LOW);
}

void forward(){
  analogWrite(motor0p,110);
  digitalWrite(motor0n,LOW);
  analogWrite(motor1n,80);
  digitalWrite(motor1p,LOW);
  digitalWrite(motor2p,LOW);
  digitalWrite(motor2n,LOW);
}

void backward(){
  analogWrite(motor0n,95);
  digitalWrite(motor0p,LOW);
  analogWrite(motor1p,75);
  digitalWrite(motor1n,LOW);
  digitalWrite(motor2p,LOW);
  digitalWrite(motor2n,LOW);
}

void brake(){
  digitalWrite(motor0n,LOW);
  digitalWrite(motor0p,LOW);
  digitalWrite(motor1p,LOW);
  digitalWrite(motor1n,LOW);
  digitalWrite(motor2p,LOW);
  digitalWrite(motor2n,LOW);
}

void left_shuffle(){
  analogWrite(motor0p,70);
  digitalWrite(motor0n,LOW);
  analogWrite(motor1p,80);
  digitalWrite(motor1n,LOW);
  analogWrite(motor2n,70);
  digitalWrite(motor2p,LOW);
}
```

```
void right_shuffle(){
  analogWrite(motor0n,70);
  digitalWrite(motor0p,LOW);
  analogWrite(motor1n,80);
  digitalWrite(motor1p,LOW);
  analogWrite(motor2p,70);
  digitalWrite(motor2n,LOW);
}

void calculate_pid(){
  P = error;                     //proportional
  I = I + error;                 //integration
  D = error - previous_error; //derivative

  PID_value = (Kp * P) + (Ki * I) + (Kd * D);

  previous_error = error;
}

void TSC_Init()
{
  digitalWrite(S0, LOW);
  digitalWrite(S1, HIGH);
}

void TSC_FilterColor(int Level01, int Level02)
{
  if(Level01 != 0)
    Level01 = HIGH;
  if(Level02 != 0)
    Level02 = HIGH;
  digitalWrite(S2, Level01);
  digitalWrite(S3, Level02);
}

void TSC_Count()
{
   g_count ++ ;
}

void TSC_Callback()
{
   switch(g_flag)
   {
    case 0:
        Serial.println("->WB Start");
        TSC_WB(LOW, LOW);
        break;
    case 1:
        Serial.print("->Frequency R=");
        Serial.println(g_count);
        g_array[0] = g_count;
        TSC_WB(HIGH, HIGH);
        break;
    case 2:
        Serial.print("->Frequency G=");
        Serial.println(g_count);
        g_array[1] = g_count;
        TSC_WB(LOW, HIGH);
        break;

    case 3:
        Serial.print("->Frequency B=");
        Serial.println(g_count);
        Serial.println("->WB End");
        g_array[2] = g_count;
        TSC_WB(HIGH, LOW);
        break;
   default:
        g_count = 0;
        break;
   }
}

void TSC_WB(int Level0, int Level1)
{
   g_count = 0;
   g_flag ++;
   TSC_FilterColor(Level0, Level1);
   Timer1.setPeriod(500000);
}

void colordet()
{
   Timer1.resume();
   delay (2000);

   g_SF[0] = 0.20782*2;
   g_SF[1] = 0.24808*2;
```

```
g_SF[2] = 0.21029*2;

/*print RGB*/
Serial.println(g_SF[0],5);
Serial.println(g_SF[1],5);
Serial.println(g_SF[2],5);

g_flag = 0;

for(int i=0; i<3; i++)
   Serial.println(int(g_array[i] * g_SF[i]));

int R = int(g_array[0] * g_SF[0]);
int G = int(g_array[1] * g_SF[1]);
int B = int(g_array[2] * g_SF[2]);
Timer1.stop();
float index_R1 = 0.01808125;
float index_G1 = 0.0070016;
float index_B1 = -0.04289697;
float index_R2 = -0.03512899;
float index_G2 = 0.09511748;
float index_B2 = -0.04180179;

float para1 = R*index_R1 + G*index_G1 + B*index_B1;
float para2 = exp(-para1);
float p1 = 1/(1+para2);

float para3 = R*index_R2 + G*index_G2 + B*index_B2;
float para4 = exp(-para3);
float p2 = 1/(1+para4);

/*determine whether the color is yellow*/
if (p1 >= 0.5 && p2 >= 0.5){

Serial.print("lid open: ");
Serial.println(1);

myServo.attach(sermotor);
myServo.write(180);  // lid  open for 4 seconds
delay (4000);
myServo.write(0);
delay (50);
myServo.detach();
}
```

```
    else{
    Serial.print("lid close: ");
    Serial.println(0);
    }
}
```