Hindawi
Journal of Robotics
Volume 2024, Article ID 5928227, 18 pages
https://doi.org/10.1155/2024/5928227

Research Article

The Navigation of Home Service Robot Based on Deep Learning and Machine Learning

Yupei Yan,[1] **Weimin Ma**,[1] **Sengfat Wong**,[2] **Xuemei Yin**,[1] **Qiang Pan**,[1] **Zhiwen Liao**,[1] **and Xiaoxin Lin**[1]

[1]*Zhuhai City Polytechnic College, Zhuhai 519090, China*
[2]*Faculty of Science and Technology, University of Macau, Macau 999078, China*

Correspondence should be addressed to Yupei Yan; 195179265@qq.com

Received 23 August 2022; Revised 7 October 2023; Accepted 20 January 2024; Published 12 February 2024

Academic Editor: Changsheng Li

This paper discusses how to improve the accuracy of navigation for home service robot based on the deep learning and machine learning. First, the crawling programing is applied to collect enough images of fridge and washing machine on the web; a deep learning framework is proposed that can distinguish fridge and washing machine more accurately. Following, the data come from the robot operating system topics are collected and cleaned, the linear regression, decision tree, and linear SVR algorithms are applied and compared to predict the power consumption of the robot, and a conclusion is obtained that liner movement will consume more power, which provides a reference for the path planning of the robot. Lastly, the conclusions are proposed that a novel methodology is applied to distinguish different home appliances, which is useful for the accurate navigation of the robot; the liner movement will consume more power compared to turning left or right, which supplies a reference for the optimized path planning for the robot.

1. Introduction

Home service robot has been researched for many years, but there are still some problems for the application of them. One problem is the accurate location of the home service robot in the house, where is an indoor dynamic environment; because of the limited geomagnetic field signal in the indoor environment, the data from inertial measurement unit (IMU) sensors are drifting dramatically in the indoor environment, there is accumulated error of odometer data, which will be more obvious when robot moves for long time.

The map can be built to solve the location problem, but there are still two problems. The first problem is that it is a dynamic environment inside the house; people and animals are moving in the house, and when the robot needs to build the map, people need to stay in the house to control the movement of the robot, which becomes the interference factor for the map building. The second problem is that the position of home appliances and furniture will be changed by people sometimes, or some new home appliances and furniture are added, and old home appliances and furniture are thrown away, which needs the robot to build the map again frequently and brings the trouble for people.

Object recognition is a competitive option for the navigation of the robot in the house [1]; people just tell robot to move to the fridge to take some food or tell robot to take dirty clothes to the washing machine, and the robot can move to the position of the fridge or washing machine by recognizing the appearance of them in the house, which likes the human action in the real world.

(i) One of the contributions of this paper is that it proposes a novel deep learning framework to distinguish the appearance of fridges and washing machines more accurately when they look similar or when they are partly shielded in the dark environment, which is useful for the robot to judge whether it moves to the correct and accurate position to finish some tasks, compared with other navigation algorithms in the

indoor and dynamic environment, this methodology improves the accuracy for the navigation of the robot.

(ii) Besides the accurate navigation of the home service robot in the indoor environment, how to save the battery power is also important; the power will be different when robot moves forward and backward or turns left and right. The data of movement are relative with the odometer data in the robot operating system (ROS), which can be read in the ROS topics; the original data contain many noise data values, abnormal values, missing values, and duplicate values, which need to be cleaned from the original data, after data cleaning, the data need to be normalized and uniformed to eliminate the unevenness of the numerical value in the data table. The classified data and predicted data should be assigned, and then the various machine learning algorithms are running to analyze which kind of movement will consume more power; it guides the robot to try to take less such movement and try other less power consumption movements in the navigation [2].

(iii) The another contribution of this paper is that a conclusion is obtained that the liner movement will consume more power compared with the rotation, which comes from the result of the machine learning algorithms; it supplies a novel methodology to analyze power consumption; also, it supplies a reference that the robot should try to move less linearly when planning the least power path in the house where is an indoor and dynamic environment.

(iv) Compared with the mathematical and analytical methodologies of literature, the methodologies in this paper are proposed based on the data analysis algorithms, such as deep learning and machine learning, which provide another novel idea to solve the problem of accurate navigation of the home service robot.

(v) This paper proposes two items; one is accurate navigation of the home service robot in the indoor environment based on visual recognition of the home appliance through the novel-designed deep learning framework. The other is that the power consumption of the home service robot is predicted based on three machine learning algorithms, the corresponding data are read from the topics of ROS, and the data are cleaned and uniformed before running the machine learning algorithms, which provides a reference for the optimized path planning.

2. Location by Deep Learning

The home service robot usually works in a house where an indoor dynamic environment, with the home appliances and furniture removed, added, and changed position. The accurate navigation of the robot by laser scan sensor seems difficult; if robot can recognize the position of home appliances and furniture by camera, the service function will be intensified.

Tensorflow Object Detection library can be loaded in Ubuntu and transformed as the message format in ROS, which can recognize the home appliances and furniture in the house in ROS, but the confidence becomes lower when the light is dimmed or sheltered by people, animals, plant or other objects, or taking video from the top view of the object by the camera on the robot arm, or taking the video of a part of the object by camera, as shown in Figure 1.

Another problem is that it seems difficult to distinguish two similar objects when they are placed together in a dark environment, like fridge and washing machine are placed together in the house, as shown in Figure 2.

Usually, the other home appliances are easy to be recognized and distinguished by the Tensorflow Object Detection library or other models and libraries; the appearance of fridges and washing machines are rectangular columns, it is difficult to be recognized and distinguished in some special status, so the deep learning should be applied to help robot recognize the fridge and washing machine more accurately.

2.1. Web Crawler. In order to obtain enough pictures of fridges and washing machines to train the deep learning network, the web crawler programing is applied to obtain them on the internet. By input the keyword "fridge" or "washing machine" on the images item of the search engine to get the URL of them.

The request and re library are imported, the request.get () function requests the content of the URL by adding the header information, because the requested content is in image format, the content.decode () function is called to transform the format. The re.findall("objURL":"(.*?)",', htlm_1) function calls regular expressions to search for images that match the conditions, "objURL" is the dict format data with the key-value result, the ".*?" can traverse the entire web page code with less strings, the following is judging if the current result has been traversed already, if it is true, the current result will be dropped, the whole process will be circulated until the required quantity of images is obtained.

In this paper, the images of fridge and washing machine are crawled, as shown in Figures 3 and 4. A script is written to rename the crawled images as "object.number.jpg" format which is prepared for the following deep learning. The quantity of crawled images is 2,600, which can be divided into train set and test set. For washing machine, there is a total of 1,300 images, in which the train set is 1,000 images and for the test set is 300 images. For fridge, the train set is 1,000 images, and for the test set, it is 300 images as well, which can make sure the same recognition ability for fridge and washing machine.

2.2. Deep Learning Framework. There are lots of popular deep learning networks, such as LeNet5, Alex Net, Vgg16, etc. Because the purpose of the deep learning network in this paper is to distinguish the fridge and washing machine more accurately, which are two classifications and the respective characteristics are obvious, the deep learning network can be revised from LeNet5 and improved to be more suitable for the distinction task in this paper, which can be shown in Figure 5.

Figure 1: The image of home appliances that are difficult to be recognized by robot.

Figure 2: The image of fridge and washing machine when they are placed together.

The difference from LeNet5 is following: the first is that it is an RGB color image but not a gray image, so it is three-channel image which is red, green, and blue; the second is that it is two classifications but not 10 classifications, the relationship of features is not so complex, so there is two dense layers but not three dense layers; the third is that the detailed information in training image is abundant, there needs more pixels to show the abundant information, so the input is normalized as 150×150, but not 28×28. After the improvement of the LeNet5 network, the proposed deep learning network will be more suitable for the distinction task in this paper.

As shown in Figure 6, in order to process the following data, the size of all images has been normalized as 150×150, the size of the first Conv2D is 3×3, the learning stride is 1, so the output is 148×148, the quantity of convolution kernel is 16, this layer scans the whole image for the next step processing.

Because it is a valid convolution, the output shape is $150 - 2 = 148$, and there are total 16 convolution kernels whose size is 3×3; they are three-channel images; if adding the bias value, the quantity of parameters is $(3 \times 3 \times 3 + 1) \times 16 = 448$.

The kernel of the next MaxPooling2D layer is 2×2, and the size is 74×74; it extracts the max value of the 2×2 square as the characteristic in this area. There is no parameter counted in the pooling layer.

The above procedure has been circulated three times to take the max value as the obvious characteristic in the image; the parameters can be counted as above paragraphs, the flatten layer is the 1D vector which can be applied to the next fully connected layer; finally, the output result should be true or false to judge if it is fridge or not, so the output of dense_1 layer is 1. The total parameters are the summation of parameters in each layer.

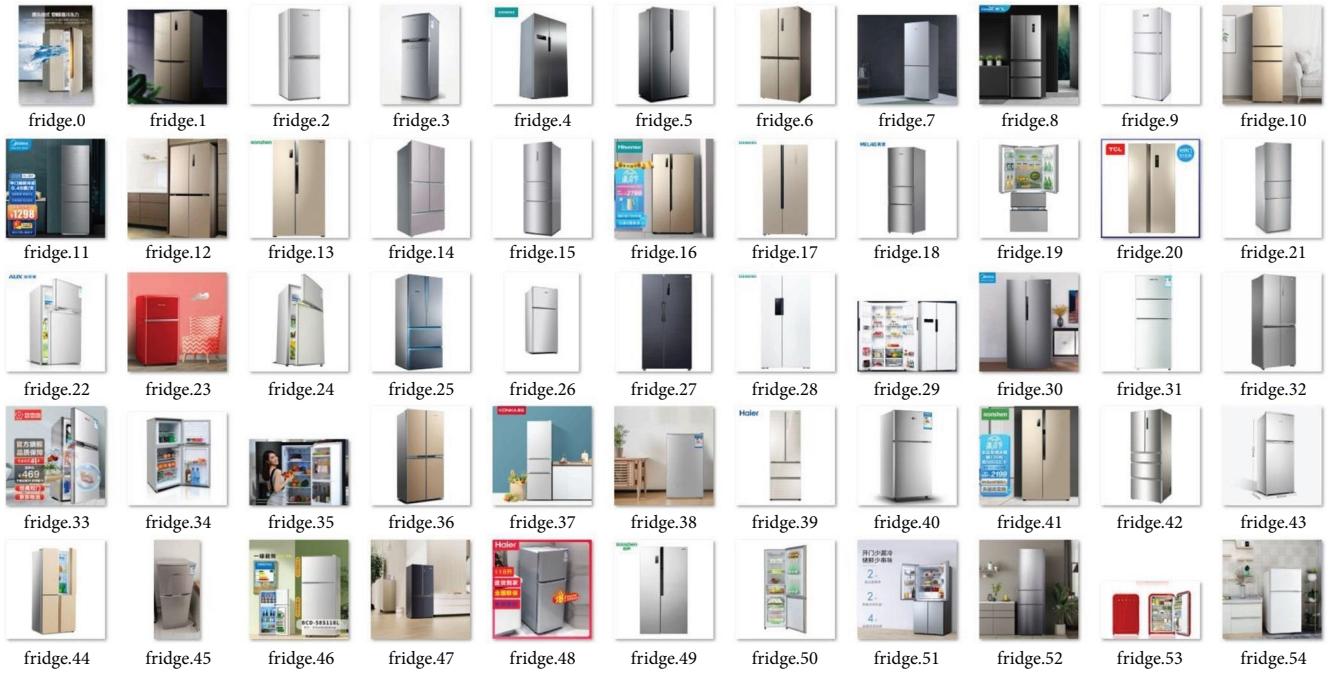

Figure 3: The crawled images of the fridge.

Figure 4: The crawled images of the washing machine.

2.3. Training and Test

2.3.1. RMSprop. In order to solve the problem of too large swing amplitude in the update process in the loss function, as well as accelerating the convergence speed of the function, RMSProp algorithm uses the differential square weighted average for the gradient of weight W and bias B, in Formula (1)–(4), W represents the horizontal axis, b represents the vertical axis, dw represents the change in the direction of the horizontal axis, db represents the change in the direction of the vertical axis, S_{dw} is the result of the gradient squared weighted average when there is changes in the direction of the horizontal axis when learning, S_{db} is the corresponding gradient squared weighted average for vertical axis, α is the learning rate, β is similar to the attenuation factor in the momentum gradient descent method, representing the influence of the past gradient on the current gradient, and the

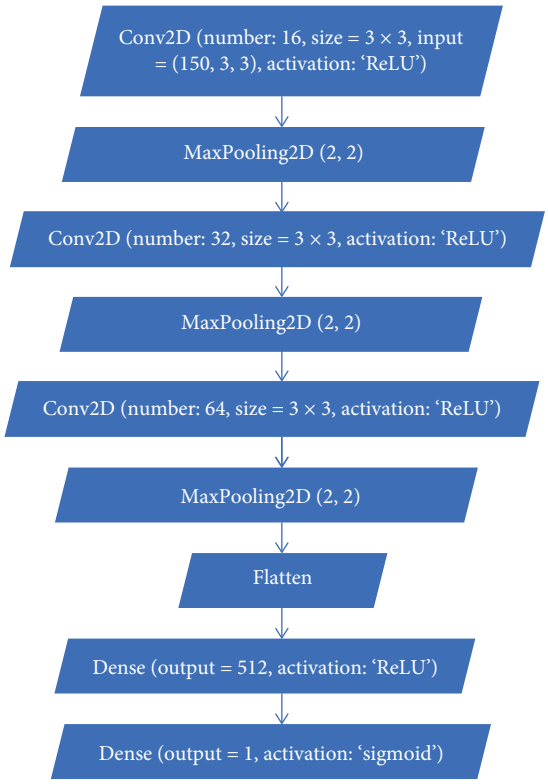

Figure 5: The framework of deep learning.

Model: "sequential"

Layer (type)	Output shape	Param #
conv2d (Conv2D)	(none, 148, 148, 16)	448
max_pooling2d (MaxPooling2D)	(none, 74, 74, 16)	0
conv2d_1 (Conv2D)	(none, 72, 72, 32)	4,640
max_pooling2d_1 (MaxPooling2)	(none, 36, 36, 32)	0
conv2d_2 (Conv2D)	(none, 34, 34, 64)	18,496
max_pooling2d_2 (MaxPooling2)	(none, 17, 17, 64)	0
Flatten (flatten)	(none, 18,496)	0
Dense (dense)	(none, 512)	9,470,464
Dense_1 (dense)	(none, 1)	513

Total params: 9,494,561
Trainable params: 9,494,561
Non-trainable params: 0

Figure 6: The structure of deep learning.

general value is 0.9. In case the denominator is zero, a very small value ε is added on the denominator; usually, it is 10^{-8}. S_{dw} and S_{db} are the gradient momentum that is accumulated by the loss function during the first $t-1$ rounds of iteration.

$$S_{dw} = \beta S_{dw} + (1-\beta)(dW)^2, \tag{1}$$

$$S_{db} = \beta S_{db} + (1-\beta)(db)^2, \tag{2}$$

$$W = W - \alpha \frac{dW}{\sqrt{S_{dw}} + \xi}, \tag{3}$$

$$b = b - \alpha \frac{db}{\sqrt{S_{db}} + \xi}. \tag{4}$$

The RMSProp algorithm calculates the differential square weighted average of the gradients, which is beneficial to eliminate the direction of large swing amplitude, correcting the swing amplitude, and making the swing amplitude of each dimension is small. On the other hand, it also makes the network function converge faster.

2.3.2. Binary Cross Entropy. Binary cross entropy is a common loss function in binary classification problems.

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\left(p(y_i)\right) + (1-y_i)\log(1-p(y_i)). \tag{5}$$

The y_i is the binary label 0 or 1, which means if the result is fridge and washing machine in this paper, $p\ (y)$ is the probability, and the output belongs to the label y. As a loss function, binary cross-entropy is used to judge the quality of the prediction result of the model. Generally, for the case where the label y is 1 and the predicted value $p\ (y)$ approaches 1, then the value of the loss function should approach 0; otherwise, if the predicted value $p\ (y)$ approaches 0, the loss function will be very large [3].

2.3.3. Training Process. For three-channel image, the scope of integer value for RGB is 0–255, which is too large to affect the predicted result, ImageDataGenerator() function will transform the value to the float whose scope is 0–1 before the training process starts, which makes the image resistant to geometric transformation, for example, it can reduce interference caused by uneven light in images, etc.

The training process will take many GPU resources; the value of "allow_growth" should be "True" to control the GPU resource gradually.

The batch size is 20, the epoch is 15, and the target size is 150×150, because binary cross entropy loss is applied, the class_mode is "binary," there are 100 steps in each epoch, the return times of validation generator are 50, the log is recorded for each epoch. After the above parameters are setup, the training process starts.

From Figure 7, it is obvious that the loss becomes lower and accuracy becomes higher in each epoch; the accuracy reaches 99.3% in the 15th epoch, which means that the model has a perfect distinction function for the training images, but the accuracy can not reach so high for the validating images.

```
Epoch 1/15
2022-07-24 11:45:24.281225: I tensorflow/stream_executor/platform/default/dso_loader.cc:44]
Successfully opened dynamic library cudnn64_7.dll
2022-07-24 11:45:25.302134: W tensorflow/stream_executor/cuda/redzone_allocator.cc:312]
Internal: Invoking ptxas not supported on Windows
Relying on driver to perform ptx compilation. This message will be only logged once.
2022-07-24 11:45:25.509873: I tensorflow/stream_executor/platform/default/dso_loader.cc:44]
Successfully opened dynamic library cublas64_100.dll
100/100 - 41s - loss: 0.6335 - acc: 0.7165 - val_loss: 0.6184 - val_acc: 0.7190
Epoch 2/15
100/100 - 36s - loss: 0.3778 - acc: 0.8475 - val_loss: 1.7057 - val_acc: 0.5450
Epoch 3/15
100/100 - 36s - loss: 0.2742 - acc: 0.8960 - val_loss: 0.5220 - val_acc: 0.7980
Epoch 4/15
100/100 - 36s - loss: 0.2032 - acc: 0.9180 - val_loss: 0.5131 - val_acc: 0.8040
Epoch 5/15
100/100 - 36s - loss: 0.1324 - acc: 0.9485 - val_loss: 0.8492 - val_acc: 0.7600
Epoch 6/15
100/100 - 36s - loss: 0.0917 - acc: 0.9695 - val_loss: 0.6824 - val_acc: 0.8000
Epoch 7/15
100/100 - 36s - loss: 0.0641 - acc: 0.9740 - val_loss: 0.9078 - val_acc: 0.8280
Epoch 8/15
100/100 - 36s - loss: 0.0668 - acc: 0.9835 - val_loss: 1.0911 - val_acc: 0.8180
Epoch 9/15
100/100 - 35s - loss: 0.0435 - acc: 0.9835 - val_loss: 1.5149 - val_acc: 0.7830
Epoch 10/15
100/100 - 36s - loss: 0.0404 - acc: 0.9890 - val_loss: 1.3739 - val_acc: 0.8000
Epoch 11/15
100/100 - 36s - loss: 0.0374 - acc: 0.9915 - val_loss: 1.3672 - val_acc: 0.8020
```

Figure 7: The training process.

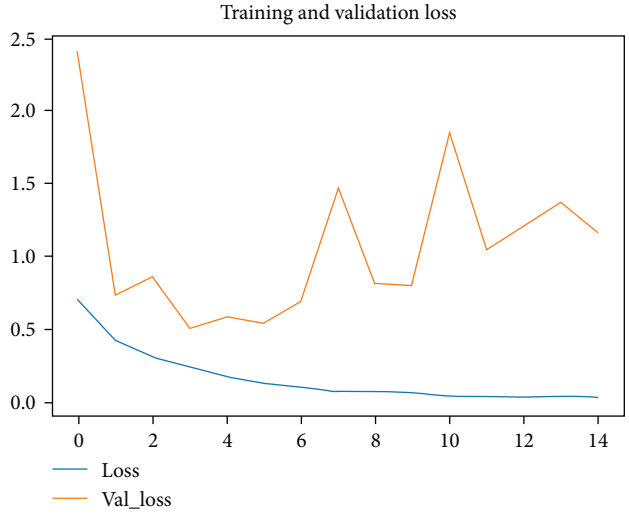

Figure 8: The training and validation loss.

From Figure 8, the loss near zero in the 15th epoch that shows the fitting is so good for the training images, but the validation loss is drifting in 15 epochs, it shows that the model is not so perfect for the test images, there are lots of reasons for this problem; usually, it is caused by the overfitting, while it also means the fitting is not suitable for the totally new test images.

The final result is the probability of fridge, which is called validation accuracy; if it is below 50%, it means the image is a washing machine; otherwise, it means the image is a fridge. In epoch 1, the validation accuracy is 50.7%; for the two classifications, it is nearly half of the full probability, which seems reasonable because it is the 1st epoch. In epoch 15, the validation accuracy is 83.5%, which means that the classification function has been improved a lot for the new images in the test set. Because there are some special images in the test set, as shown in Figure 1, the model can help robot to distinguish the fridge and washing machine when they are shielded or only part of them are observed.

3. Path Planning by Machine Learning

Usually, quintic polynomials are applied in the path planning for arm, mobile robot, and UAV; for the robot that is powered by battery, good power consumption management is not only for the cost, heat control, and charging time but also it can save the life of the battery and reduce the warranty times.

Usually, power consumption is the multiplied term of the voltage and current, which relates with the motion of wheels for the home service robot, and the motion of wheels relates with the movement distance and rotation angle, which are read from the odometer, so if the relationship between odometer data and power value can be analyzed, the optimized path planning will be clear.

Figure 9: The nanorobot in the test.

In order to test the power consumption more accurately, only base_control topic is launched in ROS; other topics like robot_navigation, robot_vision are not launched, so devices like laser scan sensor and camera do not work in this situation.

3.1. Processing and Analysis of Power Consumption Data. In ROS, odometer data can be outputted from/odom topic, voltage, and currents value can be outputted from/battery topic; both of them can be recorded by inputting command "rostopic echo /odom > odom.txt," "rostopic echo /battery > battery.txt" in Ubuntu.

In the test, in order to collect the odometer data and power value evenly in all directions, the tested mobile robot "nanorobot," which is controlled by application software in the mobile phone to move randomly while avoiding collisions in each direction in the home, as shown in Figure 9, "nanorobot" is produced by WeiEmbedded company; the main components of "nanorobot" are laser scan sensor, binocular depth camera, the model number of NVIDIA GPU board is Jetson Xavier NX, the operating system (OS) is Ubuntu 16.04 LTS, the version of ROS is Kinetic.

From Figure 10, the existing key "secs" can be recorded as "second," which is recorded since 0 : 00, January 1, 1970 in Linux system; "position: *x*, *y*, *z*" can be simplified as "*px*, *py*, *pz*"; "orientation: *x*, *y*, *z*, *w*" can be simplified as "*ox*, *oy*, *oz*, *ow*"; "linear: *x*, *y*, *z*" can be simplified as "*lx*, *ly*, *lz*"; "angular: *x*, *y*, *z*" can be simplified as "*ax*, *ay*, *az*," as shown in Table 1. The /odom topic is published many times in one second, so the value of many rows belongs to one second; for example, the value with an index from 0 to 9 belongs to one second 1,691,289,812 in Table 1.

So, the values of each key in one second should be averaged, and taking "second" as the primary key, which can be expressed in Table 2.

From Figure 11, for the output data from/battery topic, the existing key "secs," which can be recorded as "second"; other keys that need to be extracted are "voltage" and "currents," as shown in Table 3. The battery topic is published only one time in each second; there is no need for the values to be averaged in one second and take the "second" as the primary key; the table can be expressed as Table 4.

(i) In Table 4, the power is equal to the voltage multiplied by currents, so the new key "power" can be added into Table 4. Comparing Tables 2 and 4, the key "second" is the primary key in both tables, so both tables can be merged as a whole table by the key "second," which can be shown in Table 5.

(ii) In Table 1, the key "*px*," "*py*," "*pz*" means the position of the robot in three dimensions coordinate axes, "*ox*," "*oy*," "*oz*," "*ow*" means the quaternion value which represents the orientation of the robot in three dimensions coordinate axes, they represent the gesture of the robot in one moment, but they have no relationship with the power consumption of the robot, so they can be deleted from Table 1. the key "*lx*,""*ly*," "*lz*" means the linear velocity of the robot in three dimensions coordinate axes, the key "*ax*," "*ay*," "*az*" means the angular velocity of the robot in three dimensions coordinate axes, in which *x*, *y*, *z* is the rotation axis, because the home service robot moves on the horizontal surface, the value of linear *z* and angular *x*, *y* can be deleted from the whole table. Because the key "'power" is in Table 5, the keys "voltage" and "current" can be deleted, so only keys "*lx*," "*ly*," "*az*," and "power" are kept, as shown in Table 6.

(iii) In order to measure the power consumption for the movement of the robot, the values of keys are recorded when robot keeps in both stopped and movement status.

(iv) From Table 7, it is obvious that when robot keeps in stopped status and does not move again, there is still power consumption in each second, which comes from the main board of the robot. So, in order to obtain the power only for the movement of the robot, the value of key "power" when values of other keys are not zero should reduce the value of key "power" when values of other keys are zero, which can be expressed as the following formula:

$$\begin{aligned}\text{Power}_{\text{only movement}} &= \text{Power}_{\text{values of other keys are not zero}} \\ &\quad - \text{Power}_{\text{values of other keys are zero}}\end{aligned} \tag{6}$$

(v) The following is how to define the power value when the values of other keys are zero, and the first is computing the averaged power value when the values of other keys are zero, which represents computing the power value when the robot keeps stopped, then dropping the rows whose values of other keys are zero, which means dropping the values when the robot keeps stopped; finally, all the power values reduce this averaged power value, in this way, the balanced power values can represent the power consumption for the movement.

(vi) The next is to eliminate the noise and abnormal data. The first is clearing the noise data; because of sensor error or systematic statistics error, some power value points will become much higher or lower in some status, so if the values of each key, such as value of "*lx*," or "*ly*," or "*az*," or "power" at this second is three times larger than the corresponding values at the previous one second and

```
header:
  seq: 13783
  stamp:
    secs: 1658817762
    nsecs: 265778064
  frame_id: “odom”
child_frame_id: “base_footprint”
pose:
  pose:
    position:
      x: –1.91661363317
      y: –0.87584730431
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: –0.14780941113
      w: 0.989015863362
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
twist:
  twist:
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
```

Figure 10: The outputted odometer data from/odom topic.

the next one second, or 1/3 times smaller than the corresponding value at the previous one second and the next one second, this value can be considered as the noise value, whose row should be dropped. The second is clearing the abnormal data, and the universal method is the box plot algorithm, which can drop the data that is out of the normal range.

(vii) Usually, when a car starts moving, the initial speed is low, but the fuel consumption is large; when the car keeps a stable, fast speed, the fuel consumption is not so large as it starts movement, because when the car starts moving, the accelerated speed is large, based on the Newton's second law, the force is large, so the power is large, it can be concluded that power is in direct proportion to the accelerated speed, which can be regarded as the value of key "*lx*", "*ly*", "*az*" in this row reduce the corresponding values in the present row.

Because the values in the tables are all float type, it is not convenient for the processing of some algorithms, such as decision tree algorithm, etc.; all values should be transformed into integer format. Meanwhile, in order to eliminate the influence of big data in the tables, all values should be normalized between 0 and 100,000, the formula can be listed as follows:

$$X^* = \frac{X - \text{Min}}{\text{Max} - \text{Min}} \times 100,000. \tag{7}$$

As shown in Figure 12; when green and yellow lines become higher, the red lines become higher, when green and yellow lines become lower, the red lines become lower, but this phenomenon is not so obvious for the blue lines, so the value of linear x and linear y have more influence to power consumption than the value of angular z. Because the robot keeps stopped from the second 1,691,290,153–1,691,290,431 and after the second 1,691,291,512, the values of "*lx*", "*ly*", "*az*" are zero and have been dropped already in the previous steps, so there are only straight lines during those periods in Figure 12.

Table 1: The table of odometer data.

No. of the rows	Second	*px*	*py*	*pz*	*ox*	*oy*	*oz*	*ow*	*lx*	*ly*	*Iz*	*ax*	*ay*	*az*
0	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
5	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
6	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
7	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
8	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
9	1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
11	1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
12	1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
13	1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
14	1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
15	1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
16	1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
17	1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table 2: The odometer data in each second.

Secs	*px*	*py*	*pz*	*ox*	*oy*	*oz*	*ow*	*lx*	*ly*	*Iz*	*ax*	*ay*	*az*
1,691,289,812	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1,691,289,813	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1,691,289,814	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1,691,289,815	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1,691,289,816	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1,691,289,817	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00016	0.00000	0.00000	0.00000	0.00000	0.00008
1,691,289,818	−0.11564	−0.56495	0.00000	0.00000	0.00000	−0.81075	0.58538	0.28578	0.00008	0.00000	0.00000	0.00000	−0.00088
1,691,289,819	−0.15821	−0.68891	0.00000	0.00000	0.00000	−0.80881	0.58807	0.02188	0.00080	0.00000	0.00000	0.00000	0.01096
1,691,289,820	−0.14351	−0.62755	0.00000	0.00000	0.00000	−0.78669	0.61706	−0.16102	−0.00088	0.00000	0.00000	0.00000	0.12042
1,691,289,821	−0.11413	−0.45507	0.00000	0.00000	0.00000	−0.74442	0.66746	−0.19116	−0.00064	0.00000	0.00000	0.00000	0.11964
1,691,289,822	−0.10686	−0.21683	0.00000	0.00000	0.00000	−0.69703	0.71677	−0.27274	0.00024	0.00000	0.00000	0.00000	0.12290
1,691,289,823	−0.12676	0.02188	0.00000	0.00000	0.00000	−0.65164	0.75840	−0.14368	0.00016	0.00000	0.00000	0.00000	0.06582
1,691,289,824	−0.13119	0.05264	0.00000	0.00000	0.00000	−0.64584	0.76343	0.07314	0.00104	0.00000	0.00000	0.00000	−0.05536
1,691,289,825	−0.10136	−0.23459	0.00000	0.00000	0.00000	−0.69146	0.72205	0.29552	−0.00056	0.00000	0.00000	0.00000	−0.12578
1,691,289,826	−0.10522	−0.47044	0.00000	0.00000	0.00000	−0.72481	0.68878	0.45550	0.00304	0.00000	0.00000	0.00000	−0.08218
1,691,289,827	−0.16671	−1.12003	0.00000	0.00000	0.00000	−0.75394	0.65678	0.70800	0.00360	0.00000	0.00000	0.00000	−0.08050
1,691,289,828	−0.29763	−1.81670	0.00000	0.00000	0.00000	−0.78760	0.61600	0.71010	0.00432	0.00000	0.00000	0.00000	−0.07926
1,691,289,829	−0.49728	−2.49606	0.00000	0.00000	0.00000	−0.81536	0.57880	0.70864	0.00360	0.00000	0.00000	0.00000	−0.07548

```
header:
  seq: 207
  stamp:
    secs: 1658817691
    nsecs:     6989955
  frame_id: "base_footprint"
voltage: 12.2460002899
current: 0.245000004768
charge: 0.0
capacity: 0.0
design_capacity: 0.0
percentage: 0.0
power_supply_status: 0
power_supply_health: 0
power_supply_technology: 0
present: False
cell_voltage: []
location: ''
serial_number: ''
---
```

Figure 11: The outputted battery data from/battery topic.

Table 3: The table of voltage and current value.

No. of the rows	Second	Voltage	Current
0	1,691,289,815	12.17400	0.31200
1	1,691,289,816	12.17400	0.19500
2	1,691,289,817	12.17400	0.19500
3	1,691,289,818	12.20800	0.20000
4	1,691,289,819	12.19800	0.61200
5	1,691,289,820	12.19800	0.27900
6	1,691,289,821	12.16500	0.41200
7	1,691,289,822	12.13600	0.40000
8	1,691,289,823	12.09400	0.50800
9	1,691,289,824	12.05600	0.15800
10	1,691,289,825	12.11700	0.18700
11	1,691,289,826	12.15500	0.02900
12	1,691,289,827	12.05600	2.27500
13	1,691,289,828	11.99900	0.88700
14	1,691,289,829	11.91800	0.89500
15	1,691,289,830	11.93700	0.88700
16	1,691,289,831	11.92700	0.37000
17	1,691,289,832	12.01800	0.19500

From the Pearson correlation degree matrix in Table 9, the same conclusion can be obtained, that the linear x and linear y have more influence on power consumption; it means that when robot moves linearly, such as move forward or backward, the power will be consumed more, when robot have angular velocity, such as turn left and right, the power will be consumed less.

Besides the rotation and straight line, the moving path can be a curved line; in Figure 13, the green rectangle is the mobile robot, the red ellipse is the obstacle that appears suddenly in front of the robot in the dynamic environment,

Table 4: The voltage and current value in each second.

Secs	Voltage	Current
1,691,289,815	12.17400	0.31200
1,691,289,816	12.17400	0.19500
1,691,289,817	12.17400	0.19500
1,691,289,818	12.20800	0.20000
1,691,289,819	12.19800	0.61200
1,691,289,820	12.19800	0.27900
1,691,289,821	12.16500	0.41200
1,691,289,822	12.13600	0.40000
1,691,289,823	12.09400	0.50800
1,691,289,824	12.05600	0.15800
1,691,289,825	12.11700	0.18700
1,691,289,826	12.15500	0.02900
1,691,289,827	12.05600	2.27500
1,691,289,828	11.99900	0.88700
1,691,289,829	11.91800	0.89500
1,691,289,830	11.93700	0.88700
1,691,289,831	11.92700	0.37000
1,691,289,832	12.01800	0.19500

the destination is the yellow rectangle. In order to avoid the collision with the red obstacle, there are three optional moving paths that can be chosen [4]; the first is the orange color arc line, the second is the blue line, the third is the green line, the blue line can be divided into five stages which are S1–S5, the green line can be divided into two stages which are d1 and d2.

$$L = \lim_{n \to +\infty} \sum_{i=1}^{n} \mathrm{Si}. \tag{8}$$

If the blue line can be divided into lots of stages, the length of orange arc line L can be taken as the infinite serial method of summation for the length of the blue line in each stage, as shown in Formula (8). It is obvious that

$$L > S1 + S2 + S3 + S4 + S5 > d1 + d2. \tag{9}$$

So, for a small part of the orange arc line, it can be taken as that the mobile robot moves along a very short blue straight line while the moving angle is the black arrow line in Figure 13, which points to the tangent line of the orange arc line [5].

For the orange arc line, the robot should rotate the angle $\alpha 1$, which is the angle between the purple color dash line and Y axis first, then rotating along the black arrow line until it reaches the destination [6], the summation of the angle along black arrow line is $\alpha 2$ which is the angle between two orange dash lines, one is the vertical orange dash line which is parallel with Y axis, the other is the nearly horizontal orange dash line which is the extension line of orange arc line.

If the mobile robot moves along the green line, it should rotate the angle $\theta 1$ first, which is the angle between the purple color dash line and gray dash line, in which the purple dash line points to the rotation angle of the mobile robot

Table 5: The table of all values in one second.

Secs	*px*	*py*	*pz*	*ox*	*oy*	*oz*	*ow*	*lx*	*ly*	*lz*	*ax*	*ay*	*az*	Voltage	Current	Power
1,691,289,815	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	12.17400	0.31200	3.79829
1,691,289,816	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	12.17400	0.19500	2.37393
1,691,289,817	−0.06495	−0.40998	0.00000	0.00000	0.00000	−0.80902	0.58779	0.00016	0.00000	0.00000	0.00000	0.00000	0.00008	12.17400	0.19500	2.37393
1,691,289,818	−0.11564	−0.56495	0.00000	0.00000	0.00000	−0.81075	0.58538	0.28578	0.00008	0.00000	0.00000	0.00000	−0.00088	12.20800	0.20000	2.44160
1,691,289,819	−0.15821	−0.68891	0.00000	0.00000	0.00000	−0.80881	0.58807	0.02188	0.00080	0.00000	0.00000	0.00000	0.01096	12.19800	0.61200	7.46518
1,691,289,820	−0.14351	−0.62755	0.00000	0.00000	0.00000	−0.78669	0.61706	−0.16102	−0.00088	0.00000	0.00000	0.00000	0.12042	12.19800	0.27900	3.40324
1,691,289,821	−0.11413	−0.45507	0.00000	0.00000	0.00000	−0.74442	0.66746	−0.19116	−0.00064	0.00000	0.00000	0.00000	0.11964	12.16500	0.41200	5.01198
1,691,289,822	−0.10686	−0.21683	0.00000	0.00000	0.00000	−0.69703	0.71677	−0.27274	0.00024	0.00000	0.00000	0.00000	0.12290	12.13600	0.40000	4.85440
1,691,289,823	−0.12676	0.02188	0.00000	0.00000	0.00000	−0.65164	0.75840	−0.14368	0.00016	0.00000	0.00000	0.00000	0.06582	12.09400	0.50800	6.14375
1,691,289,824	−0.13119	0.05264	0.00000	0.00000	0.00000	−0.64584	0.76343	0.07314	0.00104	0.00000	0.00000	0.00000	−0.05536	12.05600	0.15800	1.90485
1,691,289,825	−0.10136	−0.23459	0.00000	0.00000	0.00000	−0.69146	0.72205	0.29552	−0.00056	0.00000	0.00000	0.00000	−0.12578	12.11700	0.18700	2.26588
1,691,289,826	−0.10522	−0.47044	0.00000	0.00000	0.00000	−0.72481	0.68878	0.45550	0.00304	0.00000	0.00000	0.00000	−0.08218	12.15500	0.02900	0.35249
1,691,289,827	−0.16671	−1.12003	0.00000	0.00000	0.00000	−0.75394	0.65678	0.70800	0.00360	0.00000	0.00000	0.00000	−0.08050	12.05600	2.27500	27.42740
1,691,289,828	−0.29763	−1.81670	0.00000	0.00000	0.00000	−0.78760	0.61600	0.71010	0.00432	0.00000	0.00000	0.00000	−0.07926	11.99900	0.88700	10.64311
1,691,289,829	−0.49728	−2.49606	0.00000	0.00000	0.00000	−0.81536	0.57880	0.70864	0.00360	0.00000	0.00000	0.00000	−0.07548	11.91800	0.89500	10.66661
1,691,289,830	−0.74372	−3.11312	0.00000	0.00000	0.00000	−0.85914	0.50872	0.49500	0.00256	0.00000	0.00000	0.00000	−0.27090	11.93700	0.88700	10.58812
1,691,289,831	−0.82705	−3.26998	0.00000	0.00000	0.00000	−0.90259	0.43051	0.00226	0.00056	0.00000	0.00000	0.00000	−0.00552	11.92700	0.37000	4.41299
1,691,289,832	−0.78062	−3.20159	0.00000	0.00000	0.00000	−0.87108	0.48720	−0.17078	0.00056	0.00000	0.00000	0.00000	0.37514	12.01800	0.19500	2.34351

Table 6: The values of kept keys in one second.

Secs	*lx*	*ly*	*az*	Power
1,691,289,815	0.00000	0.00000	0.00000	3.79829
1,691,289,816	0.00000	0.00000	0.00000	2.37393
1,691,289,817	0.00016	0.00000	0.00008	2.37393
1,691,289,818	0.28578	0.00008	−0.00088	2.44160
1,691,289,819	0.02188	0.00080	0.01096	7.46518
1,691,289,820	−0.16102	−0.00088	0.12042	3.40324
1,691,289,821	−0.19116	−0.00064	0.11964	5.01198
1,691,289,822	−0.27274	0.00024	0.12290	4.85440
1,691,289,823	−0.14368	0.00016	0.06582	6.14375
1,691,289,824	0.07314	0.00104	−0.05536	1.90485
1,691,289,825	0.29552	−0.00056	−0.12578	2.26588
1,691,289,826	0.45550	0.00304	−0.08218	0.35249
1,691,289,827	0.70800	0.00360	−0.08050	27.42740
1,691,289,828	0.71010	0.00432	−0.07926	10.64311
1,691,289,829	0.70864	0.00360	−0.07548	10.66661
1,691,289,830	0.49500	0.00256	−0.27090	10.58812
1,691,289,831	0.00226	0.00056	−0.00552	4.41299
1,691,289,832	−0.17078	0.00056	0.37514	2.34351

Table 7: The table of power consumption when robot keeps in stopped status.

Secs	*lx*	*ly*	*az*	Power
1,691,291,514	0.00000	0.00000	0.00000	2.66716
1,691,291,515	0.00000	0.00000	0.00000	2.51683
1,691,291,516	0.00000	0.00000	0.00000	2.61855
1,691,291,517	0.00000	0.00000	0.00000	2.62170
1,691,291,518	0.00000	0.00000	0.00000	2.56542
1,691,291,519	0.00000	0.00000	0.00000	2.51986
1,691,291,520	0.00000	0.00000	0.00000	2.56454
1,691,291,521	0.00000	0.00000	0.00000	2.47128
1,691,291,522	0.00000	0.00000	0.00000	2.71608
1,691,291,523	0.00000	0.00000	0.00000	2.47319
1,691,291,524	0.00000	0.00000	0.00000	2.56344
1,691,291,525	0.00000	0.00000	0.00000	2.67151
1,691,291,526	0.00000	0.00000	0.00000	2.37884
1,691,291,527	0.00000	0.00000	0.00000	2.47213
1,691,291,528	0.00000	0.00000	0.00000	2.67037
1,691,291,529	0.00000	0.00000	0.00000	2.67151
1,691,291,530	0.00000	0.00000	0.00000	2.71818
1,691,291,531	0.00000	0.00000	0.00000	2.56762

when it is in the initial position [7], the gray dash line points to the rotation angle when it is in the second position after the rotation. Then, the mobile robot moves along the green line d1, following rotation, the angle $\theta 2$, which is the angle between the gray color dash line and green line d2, which points to the destination [8].

If the mobile robot moves along the orange arc line, the rotation angle is $\alpha 1 + \alpha 2$; if it moves along with the green line, the rotation angle is $\theta 1 + \theta 2$. From Figure 13, $\alpha 1 > \theta 1$, $\alpha 2 > \theta 2$, so $\alpha 1 + \alpha 2 > \theta 1 + \theta 2$, which means that the mobile robot will rotate at a larger angle if it moves along the orange arc line than it moves along the green line [9].

Combined with the moving distance and moving angle, it can be concluded that moving a green straight line will save more power than moving an orange arc line, so for the least power consumption path planning, the mobile robot should try to take rotation and move a straight line.

The following is the comparison of the blue line and the green line, assuming that the mobile robot only moves straight line but not moves arc line in the whole process; from the blue line, it is obvious that more rotation will lead to more distance from the summation of short straight lines.

Although rotation will consume less power than move straight line, but more rotations will lead to a longer path; also it will lead to a larger rotation angle because of the total summation of the rotation angle in each stage.

From the above analysis, the path of the orange arc line, more stages of the blue line, and the green straight line are considered comprehensively; in order to save the power of the battery, the path should be designed as the combination of least rotation angle and straight line movement as green line shows.

3.2. Machine Learning. If the power consumption is considered an influential factor for the path planning of the robot, the machine learning algorithm can be taken into consideration to predict the power consumption issue in this paper.

Before running the machine learning algorithm, the train data and test data should be split from the values of Table 8 with train_test_split() function [10]; there are the total 1,146 rows in the power consumption table, in which the train data set contains 1,046 rows data, the test data set contains 100 rows data.

Because the distribution of the predicted power value is during 1–1,00,000, it is difficult that the predicted power value is totally equal with the true power value. So, the error-tolerant rate is proposed to see if the predicted power value is in the range of proportion of true value. For example, predicted value $\leq$ true value $\times$ 1.3 while

predicted value $\geq$ true value $\times$ 0.7.

So it can be summarized as follows:

predicted value $\leq$ true value $\times$ "error-tolerant rate" while predicted value $\geq$ true value $\times$ (2 − "error-tolerant rate").

The value of error-tolerant rate can be taken from 1.1 to 1.6.

3.2.1. Linear Support Vector Regression (Linear SVR) Algorithm. The support vector algorithm is applied to obtain a partition hyperplane that can split the two types of data more separately; the linear SVR algorithm is evolved from the support vector algorithm, which is applied to calculate the loss value when the predicted value and true value have some bias.

In Figure 14, the green line means the error-tolerant rate is 0.7, that the line for the predicted value equals with 0.7 $\times$ true value, the yellow line means the error-tolerant rate is 1.3, that the line for the predicted value equals with 1.3 $\times$ true value, the red line means the predicted value is totally equal to true value, the data which are distributed between green line and yellow line are the values that can be accepted, which

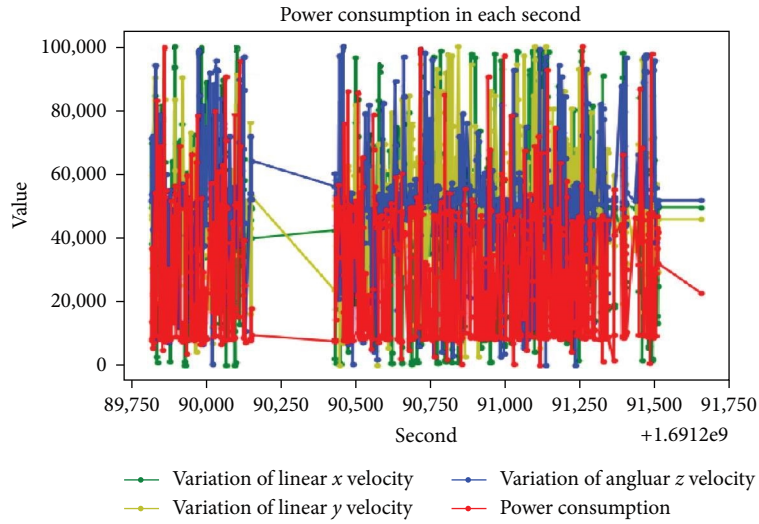

Figure 12: The variation of all values in Table 8.

Table 8: The power consumption table.

Secs	Variation of linear x velocity	Variation of linear y velocity	Variation of angular z velocity	Power consumption
1,691,289,817	49,671	45,831	51,782	7,980
1,691,289,818	69,861	47,224	51,629	8,363
1,691,289,819	51,207	59,719	53,548	36,706
1,691,289,820	38,277	30,556	71,307	13,787
1,691,289,821	36,147	34,724	71,181	22,865
1,691,289,822	30,380	50,000	71,711	21,975
1,691,289,823	39,503	48,612	62,448	29,249
1,691,289,824	54,829	63,887	42,791	5,334
1,691,289,825	70,549	36,112	31,363	7,370
1,691,289,830	84,650	90,275	7,821	54,326
1,691,289,831	49,819	55,556	50,874	19,484
1,691,289,833	86,701	69,443	94,257	40,356
1,691,289,835	47,417	43,056	50,350	9,198
1,691,289,836	31,873	20,831	56,232	9,318
1,691,289,837	2,761	47,224	51,816	83,024
1,691,289,838	1,086	40,275	62570	49,426
1,691,289,846	57,292	41,668	78,699	43,096
1,691,289,847	49,243	37,500	51,711	8,142

means that the predicted result is correct when the data are distributed between green line and yellow line.

In Figure 14, the data are distributed more evenly and in a wider range; the variance is larger than in Figure 15, and more data are distributed out of green lines than in Figure 15, the reasons should be analyzed in the following work.

3.2.2. Linear Regression Algorithm. Linear regression algorithm is a universal machine learning algorithm that predicts the data by minimizing the mean square error.

From Figure 15, it is obvious that all the values are distributed nearly parallel with X axis; the reason should be analyzed in the following work.

3.2.3. Decision Tree Algorithm. The decision tree algorithm is a tree-like structure, each node inside the tree represents the test of a feature, the branch of the tree represents each test result of the feature, and each leaf node of the tree represents a category.

In Figure 16, some data are distributed in abnormal scope, whose predicted value is large while the test value is small; the reason should be analyzed in the following work.

3.2.4. Comparison of Three Different Machine Learning Algorithms. In Table 10, the primary key in the table is error-tolerant rate whose value is taken from 1.1 to 1.6, and the three different machine learning algorithms are compared, if the

Table 9: The Pearson correlation degree matrix.

	Variation of linear x velocity	Variation of linear y velocity	Variation of angular z velocity	Power consumption
Variation of linear x velocity	1.00000	−0.03936	0.09135	−0.09774
Variation of linear y velocity	−0.03936	1.00000	−0.11345	0.17816
Variation of angular z velocity	0.09135	−0.11345	1.00000	−0.02638
Power consumption	−0.09774	0.17816	−0.02638	1.00000

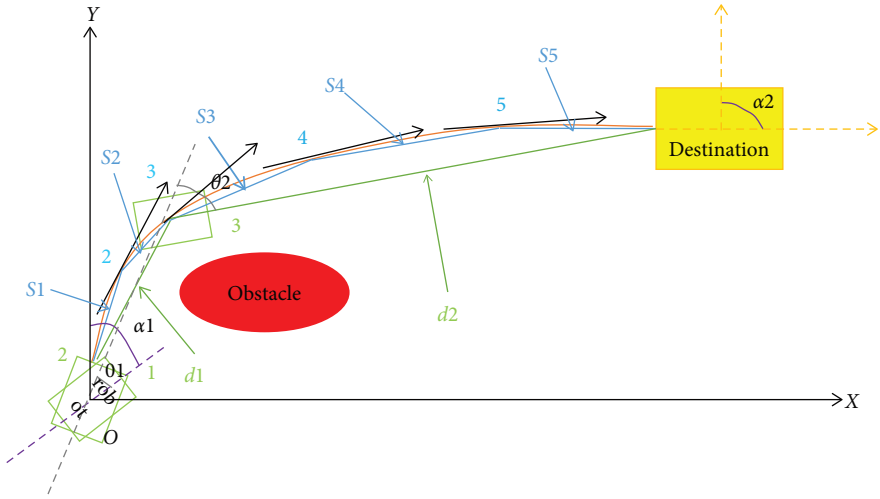

Figure 13: The optional moving path for the mobile robot when there is an obstacle ahead in the dynamic environment.

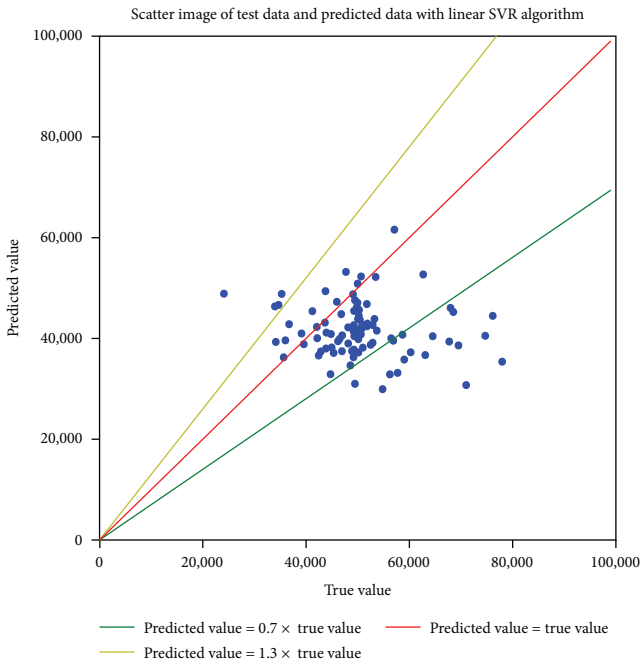

Figure 14: The test result of linear SVR algorithm.

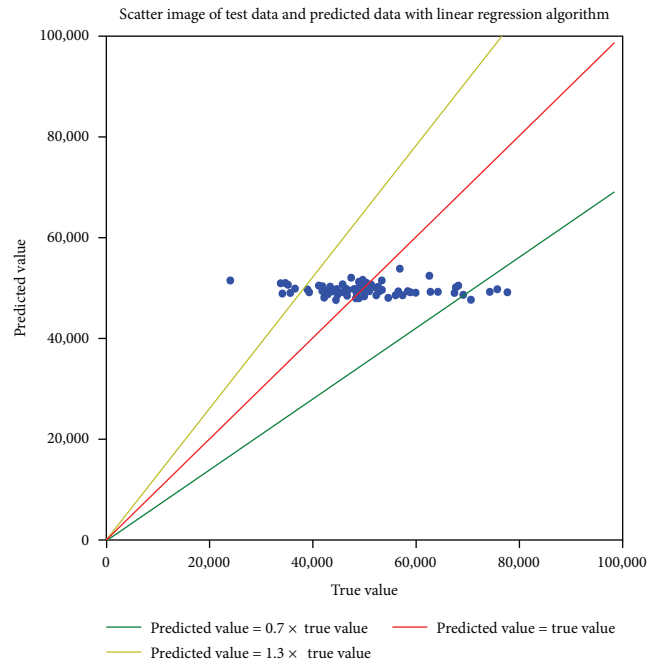

Figure 15: The test result of linear regression algorithm.

error-tolerant rate is 1.3, the accuracy of linear regression is 88%, which is the highest in three machine learning algorithms.

From Figures 14–16, it is obvious that the data are distributed more spread in the decision tree algorithm, the variance is larger, so the mean squared error is the largest in three algorithms. The data are distributed more concentrated in the linear regression algorithm, the variance is smaller, so the mean squared error is the smallest in three algorithms.

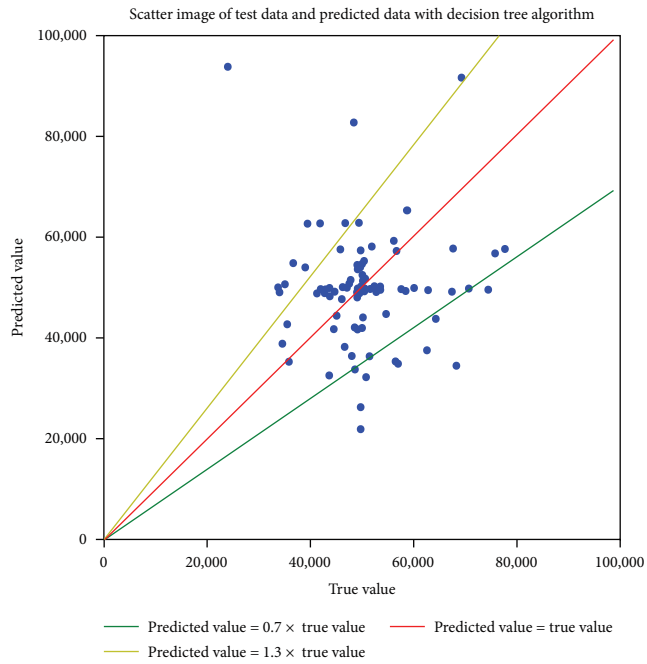

Figure 16: The test result of the decision tree algorithm.

Table 10: The accuracy of three machine learning algorithms.

Error-tolerant rate	Linear SVR	Linear regression	Decision tree
1.0	0.0	0.0	0.0
1.1	0.23	0.58	0.42
1.2	0.62	0.79	0.67
1.3	0.78	0.88	0.78
1.4	0.89	0.95	0.88
1.5	0.97	0.98	0.96
1.6	0.99	0.99	0.98

Table 11: The evaluation index of three machine learning algorithms.

Evaluation index	Linear SVR	Linear regression	Decision tree
Mean_squared_error	196514219.3683342	84900066.41350283	151702985.06
r2_score	−1.4069571985590508	−0.03987806413692363	−0.8580975621344848
Precision score	0.0	0.0	0.0
Recall score	0.0	0.0	0.0
f1_score	0.0	0.0	0.0

In Figure 15, the data are integrated into a fixed area that nearly parallel with X axis, the predicted values are trends to equal to the mean value, which is about 50,000, and the error of predicted values is distribution in a limited scope [11], which is more relative to the error of training data, so the r2 score of linear regression algorithm is near zero, but this cannot be explained in Figures 14 and 16, so the r2 score of the linear SVR algorithm and decision tree algorithm is smaller. The reason why all r2 scores of the three algorithms are negative should be analyzed in the following work.

In Formula (7), all values are normalized between 0 and 1, then magnified 100,000 times; it is almost impossible that the test value is equal to the predicted value, which can be reflected in the first row in Table 10, so the precision score, recall score, f1 score are all zero in Table 11.

The motion of wheels will consume power, the other devices on robot will consume power as well. In the test, there is an NVIDIA Jetson nano main board on the nanorobot, which will consume the power a lot and take more percent of power consumption, especially when the velocity of robot is

little. Because the accuracy and deviation of the linear regression algorithm are acceptable compared with linear SVR and decision tree algorithm, the predicted result of power consumption of the linear regression algorithm is better.

4. Conclusion

In this paper, two issues are tried to be solved for home service robot; one is the distinction of fridge and washing machine in some special situations, such as they are shielded by people or animals, or the light is dimmed or just shows a part of them in the camera of robot. The other issue is that how to judge the power consumption, which can provide a reference for how to plan the path to save power for robot.

Deep learning is applied to solve the first issue, the test result shows that the training accuracy is nearly 100%, but validation accuracy only reaches 83%, so the validated result is not so perfect; the reason maybe overfitting, so how to prevent overfitting becomes the next step work, usually the regularization method can be applied to prevent overfitting and improve the generalization ability of the model, or the parameters can be adjusted appropriately so that the w value can be more accurately adjusted during BP back propagation, or the sigmode activation function can be changed into ReLU activation function, or part of the fully connected layer can be changed into partial connected layer, or some pooling layers can also be added, deleted or modified, and the size of convolution kernel can be modified to improve the prediction accuracy of the model, etc.

For the second issue, there is a conclusion obtained that curve line movement will consume more power than straight line movement, and straight-line movement will consume more power than just rotation at one fixed position. The three different machine learning algorithms are applied to predict the power consumption; the test result shows that the predicted accuracy is not good; the reason comes from many factors; the first is the data from IMU sensors is not accurate in the indoor environment, the second is there are lots of gaps in the encoder which brings the error, the third is the friction on the ground will bring the error, the last is that the parameters of the machine learning models should be adjusted accurately to improve the accuracy of the prediction, which can be completed in the following work.

The conclusions obtained in this paper can be applied on the swarm of home service robots that communicate with each other by the distributed network in ROS [12]; for example, when one node which has powerful computation resources finishes the object recognition work with deep learning algorithm, it can send the corresponding results to another node which has less hardware resources with the distributed communication framework, which can realize the load balancing function in the distributed network.

Another application is the home robots for telemedicine/telehealth; for example, when elders need to measure the value of electrocardiosignal, blood pressure, and blood oxygen, the robot, which is equipped with physiological monitoring instruments can plan the path from its position to the position of elders with less power consumption by the proposed machine learning algorithms, then recognizing the fingers, wrist and arms of elders more accurately by the proposed deep learning algorithms.

The conclusions can be applied in other more application scenarios, which bring more development opportunities for the home service robot.

Data Availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by The Research Project of Zhuhai City Polytechnic College in 2021 (KY2021Y01Z), Project of Qualified Teachers in College in 2021 (ZLJS2020120622, ZLJS2020120623, and ZLJS2020120624), Guangdong Province Science and Technology Innovation Strategy Special Funding in 2022 (pdjh2022b0992), Guangdong University Innovation Team Project (Natural Science) (Project No. : 2023KCXTD078), and Science and Technology Innovation Team of Zhuhai City Polytechnic (Project No. : KJCXTD202005).

References

[1] S. Zhu, "Automatic recognition of facial expression based on computer vision," *International Journal on Smart Sensing and Intelligent Systems*, vol. 8, no. 3, pp. 1464–1483, 2015.

[2] H. Miao and X. Huang, "A heuristic field navigation approach for autonomous underwater vehicles," *Intelligent Automation & Soft Computing*, vol. 20, no. 1, pp. 15–32, 2014.

[3] G. Ryou, Y. Sim, S. H. Yeon, and S. Seok, "Applying asynchronous deep classification networks and gaming reinforcement learning-based motion planners to mobile robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6268–6275, Brisbane, QLD, Australia, 2018.

[4] M. Radmanesh, M. Kumar, and P. H. Guentert, "Overview of path-planning and obstacle avoidance algorithms for UAVs: a comparative study," *Unmanned Systems*, vol. 6, no. 12, pp. 95–118, 2018.

[5] G. H. Xu, T. W. Zhang, and Q. Lai, "A new path planning method of mobile robot based on adaptive dynamic firefly algorithm," *Modern Physics Letters B*, vol. 34, no. 29, pp. 159–167, 2020.

[6] S. M. Sombolestan, A. Rasooli, and S. Khodaygan, "Optimal path-planning for mobile robots to find a hidden target in an unknown environment based on machine learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 1841–1850, 2019.

[7] Z. Jian, S. Zhang, S. Chen, Z. Nan, and N. Zheng, "A global–local coupling two-stage path planning method for mobile robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5349–5356, 2021.

[8] M. Alireza, D. Vincent, and W. Tony, "Experimental study of path planning problem using EMCOA for a holonomic mobile robot," *Journal of Systems Engineering and Electronics*, vol. 32, no. 6, pp. 1450–1462, 2021.

[9] R. Fareh, M. Baziyad, T. Rabie, and M. Bettayeb, "Enhancing path quality of real-time path planning algorithms for mobile robots: a sequential linear paths approach," *IEEE Access*, vol. 8, no. 3, pp. 167090–167104, 2020.

[10] T. Li, D. Ho, C. Li, D. Zhu, C. Wang, and M. Q. H. Meng, "Houseexpo: a large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5839–5846, Las Vegas, NV, USA, 2020.

[11] M. Chandrasekaran, M. Muralidhar, C. M. Krishna, and U. S. Dixit, "Application of soft computing techniques in machining performance prediction and optimization: a literature review," *The International Journal of Advanced Manufacturing Technology*, vol. 46, no. 5–8, pp. 445–464, 2010.

[12] C. Tatino, N. Pappas, and D. Yuan, "Multi-robot association-path planning in millimeter-wave industrial scenarios," *IEEE Networking Letters*, vol. 2, no. 4, pp. 190–194, 2020.