Portfolio

Jiajun Long

- **Experimental Characterization and Comparison of Three Typical Omnidirectional Mobile Robots**
  Undergraduate Research Assistant
  Fall 2022 – Spring 2023
  Dr. Zhengzhong Jia's Lab

  Abstract:
  Omnidirectional vehicles (ODVs) or mobile robots are widely used in various industries due to their ability to move and rotate in narrow spaces without the need for multiple adjustments of moving direction or stopping. There are multiple types of ODVs, including those with standard wheels and those with special wheels such as Swedish wheels. Comprehensive characterization of the ODV's overall performance, which is very important when selecting specific configurations in the robot design process, is still missing in the literature. This paper qualitatively summarizes the comprehensive characteristics of three typical omnidirectional mobile robots in indoor scenarios through theoretical analysis and physical experiments. We compare the robot's performance through six representative evaluation metrics (maneuverability, efficiency, etc.) and express the results in a radar chart. The results of this study can assist designers in designing omnidirectional robots more effectively for specific application scenarios.

- **Smart Cleaning Robots Team Project**
  Course: Collaborative Robot Learning
  Spring 2023
  Instructor: Dr. Chaoyang Song

  Abstract:
  This project mainly focuses on the task requirements of mobile robots identifying different kinds of garbage through object recognition in household environment and can throw the garbage into the corresponding trash can. It respectively carries out in-depth exploration on the mapping navigation and object recognition of the robot, and combines the two in the later stage, and proves the feasibility of the task through three experiments: In object recognition experiment, the correct rate of object recognition in simulation environment can reach more than 90%. The similarity between the map generated in the navigation experiment with building map and the ground truth can reach more than 70%, and the effect is good in the integrated experiment. Finally, some solutions to the task limitation and prospects for the future are put forward.

- **Analysis of Multi-agent Reinforcement Learning Team Project**
  Course: Machine Learning for Engineering
  Fall 2022
  Instructor: Dr. Kemi Ding
  Abstract:
  Reinforcement learning of multiple intelligences is very important and effective for cluster-based robot develop- ment. Based on OpenAi's multiagent-particle-envs environment, we implement the multi-intelligent body reinforcement learning MADDPG algorithm. By studying the algorithm of the project and implementing it using pytorch according to our understanding. We also modified the critic-actor neural network. The algorithmic part of the MADDPG that we have built is able to interface with the experimental scenarios that we have built ourselves, enabling the learning of scenarios such as cooperation/competition/communication between robots. Finally, we analyze and summarize the learning effects of clustered robots performing different tasks.

- **Visual Control Robotic Arm Team Project**
  Course: Robot Modeling and Control
  Spring 2022
  Instructor: Prof. Zhenzhong Jia

  Abstract:
  This project introduces the invention of the 5-DOFs robot arm's architecture, visual control by Python, calculation of forward and inverse kinematics and work, procedure of trajectory planning and MATLAB simulation, operation of pressure sensor and the future solution for the visual control robot arm. Although our team didn't work on this topic before, we decided to study and explore more contents which will be covered in this project.

Experimental Characterization and Comparison of Three Typical Omnidirectional Mobile Robots

Zhuolun Li, Jiajun Long, Ximan Zhang, Chengyuan Gao, Shixing Jiang, Zheng Zhu, Zhenzhong Jia

***Abstract*— Omnidirectional vehicles (ODVs) or mobile robots are widely used in various industries due to their ability to move and rotate in narrow spaces without the need for multiple adjustments of moving direction or stopping. There are multiple types of ODVs, including those with standard wheels and those with special wheels such as Swedish wheels. Comprehensive characterization of the ODV's overall performance, which is very important when selecting specific configurations in the robot design process, is still missing in the literature. This paper qualitatively summarizes the comprehensive characteristics of three typical omnidirectional mobile robots in indoor scenarios through theoretical analysis and physical experiments. We compare the robot's performance through six representative evaluation metrics (maneuverability, efficiency, etc.) and express the results in a radar chart. The results of this study can assist designers in designing omnidirectional robots more effectively for specific application scenarios.**

I. Introduction

Mobile robots have become an indispensable part in many modern society applications, such as logistics [1], agriculture [2], military [3], medical [4], and rescue. In the research side, new designs of omnidirectional robots are constantly emerging [5]–[7], becoming one of the research hot topics in ground mobile robots. Compared to traditional non-omnidirectional robots, omnidirectional robots are usually more suitable for operations in small spaces because they can move more flexibly without the need of multiple turns. However, in the previous literature review [8], [9], the analysis of advantages and disadvantages of different omnidirectional robots was mainly based on the summary of the literature and the analysis of kinematic models, lacking direct comparisons of different types of omnidirectional robots. Selecting a suitable omnidirectional robot configuration is an important factor when designing and manufacturing omnidirectional robots. If different structures can be compared in multiple dimensions to highlight their characteristics in practical applications, it will bring more convenience to the design work.

The category of omnidirectional robots is diverse and can be divided into special wheeled and standard wheeled types based on the type of wheels used. The main types of special wheels are Mecanum [10], omniwheels [11], and ball wheels [12], among which Mecanum wheels are the most commonly used. Similar to traditional omni-directional wheels, Mecanum wheels consist of a series of freely rotating rollers connected to the outer edge of the wheel, but the rollers are oriented at a 45° angle to the wheel's central axis. A typical Mecanum robot often uses four Mecanum wheels, which are arranged in either an O-shaped or X-shaped configuration. However, some design features of Mecanum wheels can lead to negative effects such as vibration at high speeds and reduced control efficiency [13]. Standard wheeled robots mainly rely on caster wheels or active steering to achieve omnidirectional movements. Robots composed of independently steerable and independently driven wheel groups can achieve nonholonomic omnidirectional movement [6]. At least two wheel groups are required to achieve omnidirectional movement. Another commonly used method to achieve omnidirectional movement with traditional wheels is to construct wheels with offset steering axes. Robots that use the ASOC method belong to the holonomic omnidirectional robot category that uses traditional wheels.

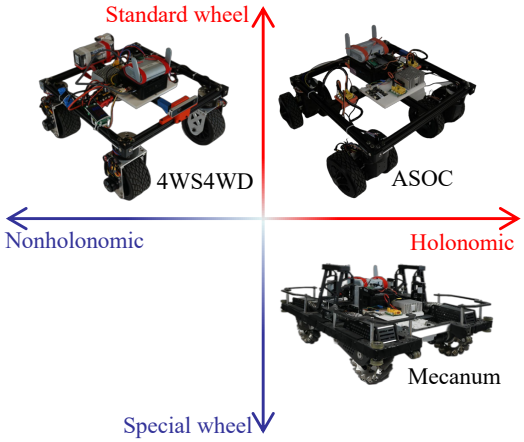


Fig. 1. The three omnidirectional robot platforms used in this paper are nonholonomic 4-wheel steering and 4-wheel drive (4WS4WD) robots that use traditional wheels, holonomic ASOC robots that use traditional wheels, and holonomic Mecanum robots that use special wheels.

Strictly speaking, holonomy is an essential attribute of omnidirectional robots. The motion of a holonomic mobile robot does not have kinematic constraints. However, nonholonomic pseudo-omnidirectional robots can also exhibit good omnidirectional motion performance in practical applications. Nonholonomic 4WS4WD mobile robots [14] need to rotate the steerable wheels to the corresponding positions before motion, and these types of omnidirectional robots are also called pseudo-omnidirectional robots. Usually, in scenarios with a main motion direction, pseudo-omnidirectional robots

The authors are with Shenzhen Key Laboratory of Biomimetic Robotics and Intelligent Systems; they are also with Guangdong Provincial Key Laboratory of Human-Augmentation and Rehabilitation Robotics in Universities, Department of Mechanical and Energy Engineering, Southern University of Science and Technology (SUSTech), Shenzhen, 518055, China.
Corresponding author: jiazz@sustech.edu.cn.
Video link: https://www.youtube.com/watch?v=re6EBwktieI

do not need to change the orientation of the wheels very quickly. There is no relevant research on the comparison of omnidirectional mobility performance between pseudo-omnidirectional robots and omnidirectional robots.

In theory, the rotational speed and translational speed of a robot are completely decoupled and independent. However, due to the motor speed limitations, there is an upper limit on the angular velocity of traction wheels. This leads to kinematic differences between different omnidirectional mobile robots [15]. Meanwhile, there are differences in traction and smoothness among the wheels of different configurations [16]. When performing agile maneuvers, these limitations significantly affect the actual motion performance of omnidirectional robots, making them difficult to meet the omnidirectional movement requirements. It is difficult to determine the robot's characteristics solely through the analysis of kinematics and dynamics without considering the characteristics of the tires.

This paper aims to verify the advantages and limitations of several typical omnidirectional mobile platforms (see Fig. 1) through actual experiments and to discuss their further improvements. These results can provide important references for the control and design of omnidirectional mobile robots. This article will first introduce the kinematic models of three typical omnidirectional mobile systems and the corresponding robots. Then, it will describe in detail the mobility tests and analyses of these three robots.

II. Platforms' Kinematics

It is a common practice to study omnidirectional moving systems using kinematic models. In this paper, three representative omnidirectional mobile systems are selected, namely: the holonomic omnidirectional mobile system with non-circular wheels, the holonomic omnidirectional mobile system with conventional wheels, and the nonholonomic pseudo-omnidirectional mobile system with conventional wheels. Furthermore, in the experimental section, the motion control of the three mobile platforms will be based on their corresponding kinematic models. The omnidirectional capability of these three types of mobile platforms has been proven by previous researchers, therefore the number of wheels for the omnidirectional mobile systems introduced in this section will be greater than the minimum required for achieving omnidirectional motion.

A. Mecanum Wheeled Robot Modeling

A single Mecanum wheel needs to be analyzed. The Mecanum wheel consists of a hub and a roller. Under the assumption of pure rolling motion, the total velocity can be decomposed into two components: one along the drive direction and the other along the slide direction. Motion direction is the non-sliding rolling direction of wheel i, which is in the same direction as the y_i axis of the reference coordinate system $\{B\}$, and it produces the driving speed. The sliding direction of the roller is the sliding direction. Its angle to the lateral are $\alpha_i = 45°, -45°, 45°, -45°$ for $i = 1, 2, 3, 4$ respectively, which generate sliding speed.

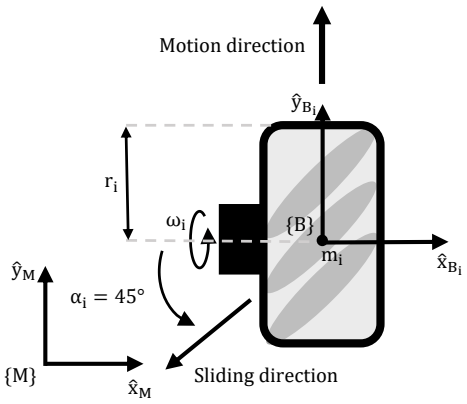


Fig. 2. Illustration of a single Mecanum wheel.

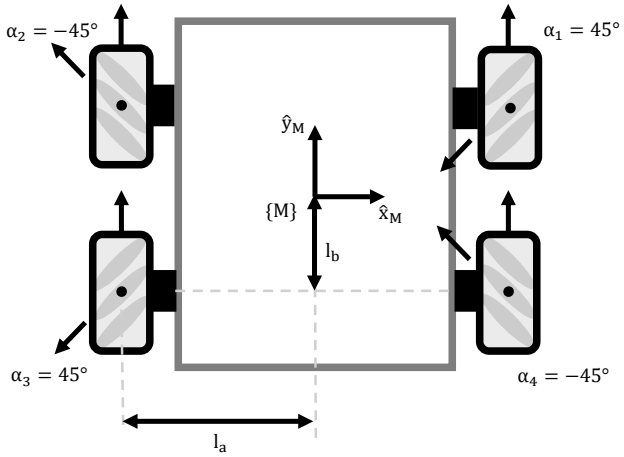


Fig. 3. Illustration of the whole Mecanum robot model.

After detailed force analysis, we can know that the linear speed of wheel i is the sum of motion speed and sliding speed. As shown in Fig.2, point m_i is the center of wheel i, and the linear velocity of point m_i can be obtained through body twist V_B which can be expressed as $V_B = [v_x \ v_y \ \omega]^T$ under the coordinate system $\{B\}$, it can be expressed as:

$$ {}^{B}\dot{m}_i = \begin{bmatrix} 1 & 0 & -y_i \\ 0 & 1 & x_i \end{bmatrix} V_B \tag{1} $$

Through this variable, the linear velocity ${}^{d_i}\dot{m}_i$ in the driving direction can be calculated, this variable can be expressed as:

$$ {}^{d_i}\dot{m}_i = \begin{bmatrix} 1 & t_{\alpha_i} \end{bmatrix} {}^{B}\dot{m}_i \tag{2} $$

The driving speed ω_i can be derived from the above two formulas:

$$ \omega_i = \frac{1}{r_i} {}^{d_i}\dot{m}_i \tag{3} $$

So far, the velocity relationship between Mecanum wheels and the whole model has been obtained, as shown in Fig.3, and the kinematic model analysis has been completed.

B. 4WS4WD Mobile Robot Modeling

Then we will analyze the kinematic model of the 4WS4WD robot. Under this model, four wheels drive and

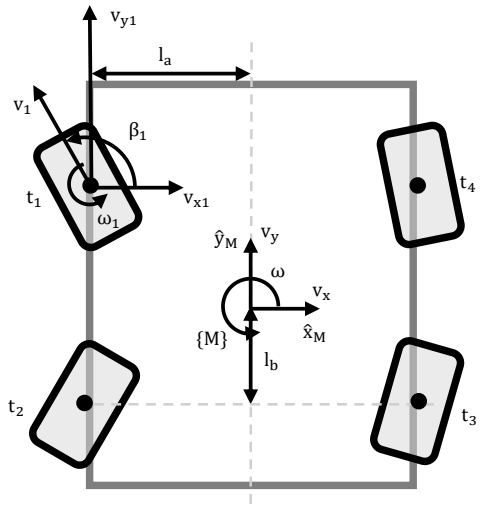


Fig. 4. Illustration of the whole 4WS4WD robot model.

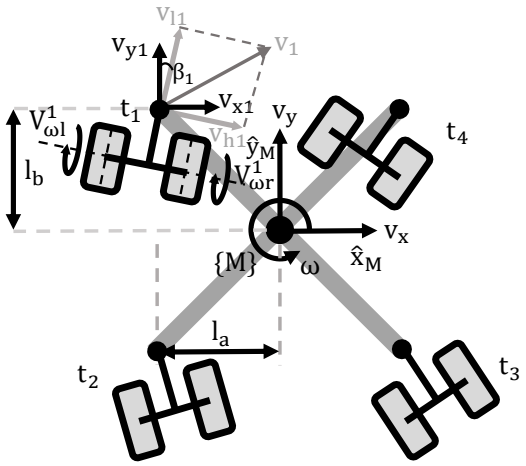


Fig. 5. Illustration of the whole ASOC robot model.

rotate independently, so each wheel needs to be analyzed independently. Here, we select one single wheel shown in the figure for analysis. First, through the calibration of the main coordinate system and the wheel coordinate system, the center of mass of the four wheels can be obtained, they are $(x_1, y_1) = (-l_a, l_b)$; $(x_2, y_2) = (-l_a, -l_b)$; $(x_3, y_3) = (l_a, -l_b)$; $(x_4, y_4) = (l_a, l_b)$. We need to define some characters in Fig.4. The linear speed of the entire model is v, and for each wheel i, its speed is v_i, which can be expressed as:

$$v = \sqrt{v_x^2 + v_y^2}; v_i = \sqrt{v_{x_i}^2 + v_{y_i}^2} \tag{4}$$

And ω is the angular velocity of the entire model. The equation below is under the assumption of no slipping situation, the speed relationship between the 4WS4WD model and each wheel can be obtained. So derive the wheel t_i, which can be expressed as:

$$\begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -y_i \\ 0 & 1 & x_i \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{5}$$

and $\beta_i = atan(v_{y_i}/v_{x_i})$.

So far, the velocity relationship between 4WS4WD wheels and the whole model has been obtained, and the kinematic model analysis has been completed.

C. ASOC-based Mobile Robot Modeling

The following is the kinematic analysis of the ASOC wheel model. In this model, eight wheels rotate independently, controlled by eight motors, and each motor in $t_i$prevents excessive wheel rotation. $\{M\}$ is the model coordinate system of ASOC wheel model, and v_x and v_y are the velocities in the direction of model x and y respectively. The resultant velocity is $v = \sqrt{v_x^2 + v_y^2}$, where ω is the total angular velocity of the ASOC wheel model. t_i is selected for analysis, as shown in Fig.5, v_i is the resultant speed of left and right wheels in the coordinate system t_i, which can be expressed as:

$$v_i = \sqrt{v_{l_i}^2 + v_{h_i}^2} = \sqrt{v_{x_i}^2 + v_{y_i}^2} \tag{6}$$

where v_{x_i} and v_{y_i} are the velocities of the pivot point of the ASOC module in the direction of x and y, respectively. And v_{l_i} and v_{h-I} are the velocities in the direction of x and y in the coordinate system of t_i, respectively. Next, take a look at the other parameters in the figure below, β_i is the angle between the coordinate system of $\{M\}$ and t_i. $V_{\omega l}^i$ and $V_{\omega r}^i$ are the velocities of the left wheel and the right wheel in the ASOC module of t_i, respectively. x_i and y_i are the x and y wheelbase departures from the pivot point of each ASOC module to the model coordinate system, respectively. At any moment of model movement, its subject can be regarded as a rigid body. After the parameters are basically defined, according to the structure of ASOC wheel model, the left wheel and right wheel speed of t_i module can be calculated, which can be expressed as:

$$\begin{bmatrix} V_{\omega_l}^i \\ V_{\omega_r}^i \end{bmatrix} = C_{\omega i} R_i C_i \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{7}$$

where

$$C_i = \begin{bmatrix} 1 & 0 & -y_i \\ 0 & 1 & x_i \end{bmatrix} \text{is the speed transition,} \tag{8}$$

$$R_i = \begin{bmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{bmatrix} \text{is the rotation matrix,} \tag{9}$$

$$C_{\omega i} = \frac{1}{R} \begin{bmatrix} \frac{L_s}{2L_o} & 1 \\ -\frac{L_s}{2L_o} & 1 \end{bmatrix} \text{is the Jacobian matrix.} \tag{10}$$

In the Jacobian matrix, R is the radius of the wheel, L_s is the offset distance of the wheel, and L_o is the distance between the hub shaft and the wheel's center of mass. So far, the velocity relationship between ASOC wheels and the whole model has been obtained, and the kinematic model analysis has been completed.

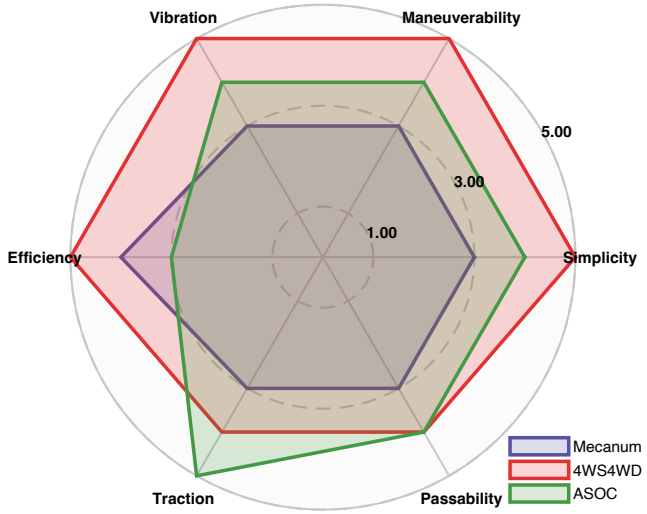


Fig. 6. A radar chart is obtained by comparing the relative performance of different entities. The robot with the best performance in a certain dimension is given a score of 5, while the worst performer is given a score of 3.

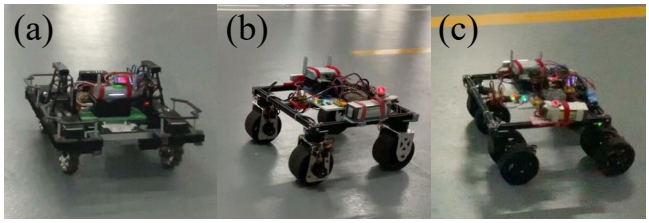


Fig. 7. The three robot platforms that participated in experiments are shown in the figures. Figure (a) shows the Mecanum robot, Figure (b) shows the 4WS4WD robot and Figure (c) shows the ASOC robot. The testing environment is an indoor parking lot with a wear-resistant epoxy material on the ground.

III. Evaluation

In order to evaluate the three types of omnidirectional robots mentioned in this paper from multiple dimensions, we adopted the method of practical experiments and comparison. The three robots are the four-wheeled Mecanum robot, the 4WS4WD robot, and the ASOC robot (see Fig. 7). According to the previous literature summary, this section evaluates omnidirectional robots from structural complexity, maneuverability, and efficiency [2], [15], [16]. The vibration [17], traction capacity [15], and passability [5] of these three types of robots have been discussed in previous studies, and therefore will not be reiterated in this section.

In Fig. 6, the numerical values of various performance indicators are obtained through comparison. For a particular characteristic, the robot with the weakest performance scored 3 points, and the robot with the best performance scored 5 points. If the performance is close or equal, it is scored the same. The specific evaluation content of the relevant characteristics will be discussed in detail in Sec. III-A and III-B. The results of several dimensions of evaluation are presented below. In terms of traction ability, the ASOC robot has more driving motors (8 motors) and uses traditional wheels, while the Mecanum robot uses only 4 motors, and the rotational efficiency of Mecanum wheels is lower than that of traditional wheels [16]. Therefore, the ASOC robot has the most potent traction ability, while the Mecanum wheel has the weakest traction ability. In terms of vibration, the multi-roller structure used by the Mecanum wheel will produce discontinuous contact during movement [17], causing significant vibration. The ASOC robot uses 8 motors, resulting in more vibration than the 4WS4WD robot. In terms of passability, the Mecanum wheel has weak passability during lateral movement. The ASOC robot and the 4WS4WD robot, both using traditional wheels, have similar passability. In terms of structural complexity, the Mecanum wheel has the most number of parts, mainly due to the complex structure of the roller-combined wheel. The ASOC wheel group has two motors, which increases the complexity of the structure, while the 4WS4WD robot is the most concise among the three robots. In terms of maneuverability, based on the experiments in Sec. III-A and III-B, it can be seen that the 4WS4WD robot performs the best in terms of maneuverability, while the overall performance of the Mecanum robot is the worst. In terms of power consumption and efficiency, based on the experimental results, the ASOC robot has more driving motors, resulting in the highest energy consumption when completing the same working conditions, and the lowest driving efficiency. The mecanum robot has a lower driving efficiency due to significant slippage at high speeds. In comparison, the 4WS4WD robot has the lowest energy consumption.

The experimental results will have performance deviations caused by vehicle size and part selection. In order to reflect the impact of robot configuration on practical applications, the wheel sets of these three omnidirectional robot platforms are directly rigidly connected to the body. And uniformly adopt the four-wheel set platform, because the four-wheel set is the most widely used in various fields. Table I is the parameter table of the three robots. To compare the motion performance between platforms, we standardized the platform frame, motor selection, and tire selection (each wheel group has the same tire width). The weight difference between the platforms is mainly due to the different number of actuators used, which is caused by the choice of robot type. In order to compare the assembly complexity of each platform, the table also counts the number of parts (including fasteners) for each robot.

We use the Intel T265 visual odometry as the trajectory recording module, which can ensure a certain positioning accuracy (1cm) indoors. The path tracking module adopts a forward-looking PI controller. Under different target speed

TABLE I
THE PARAMETER TABLE OF THE THREE ROBOTS.

	Mecanum	**4WS4WD**	**ASOC**
Length (m)	0.370	0.385	0.385
Width (m)	0.315	0.385	0.385
Actuators	4 * M3508	4*M3508 4*GM6020	8*M3508
Weight (kg)	9.2	8.0	10.9
Parts (pcs)	708	228	236

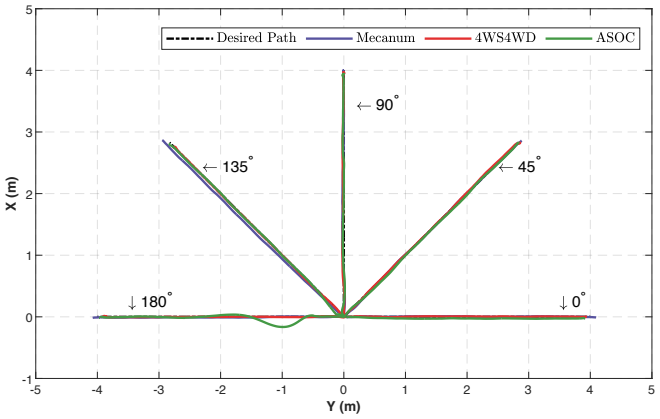


Fig. 8. In isotropic experiments, the motion trajectory diagrams of three robots are presented. The tracking errors of Mecanum and 4WS4WD robots are similar, while the ASOC robot exhibits significant oscillations during the initial motion in large turning scenarios. However, its tracking error converges within 3 meters.

conditions, only the target speed of the control module is changed, and the parameters in the PI controller are not modified.

A. Characterization of the System's Kinematic Isotropy

From the kinematic analysis of omnidirectional robots, they should have the same ability to move in all directions, but the dynamics of the robot, the dynamics of the wheel-set, and the slip of the tires will cause deviations in actual motion. In order to study the influence of configuration on motion performance in all-directional motion, we designed isotropic experiments (see Fig. 8). The platform is moved in various directions ($0°$, $45°$, $90°$, $135°$, $180°$) at a target speed of $1m/s$ to evaluate the tracking error and speed maintenance ability of the robot.

As shown in Fig. 9(a-c), Mecanum and 4WS4WD robots are capable of maintaining very low tracking errors in all directions (maximum tracking error $< 0.07m$). ASOC robot exhibits a significant increase in maximum tracking error ($> 0.15m$) in motions greater than $90°$. The Mecanum robot has different resistances in omnidirectional motion, which makes it difficult to maintain target speed in all directions (see Fig. 9(a)). The average power consumption of the Mecanum robot (see Fig. 9(e)) in all directions can also demonstrate its issues with maintaining speed. The average driving power consumption of the robot is greater than $15W$ at $45°$, $90°$, and $135°$ (higher than at $0°$ and $180°$), and the robot cannot reach the predetermined speed, resulting in steady-state speed tracking errors. Literature also indicates that omnidirectional robots with non-circular wheels are more susceptible to performance losses due to installation errors [16].

The 4WS4WD robot has a wheel steering time. As the motion angle increases, the influence of the steering time on the actual motion speed also increases (see Fig. 9(b)). It is worth noting that the torque of the steering motor needs to overcome the scrubbing torque of the tires when turning in place. In practical applications, when the chassis is under high load, there will be a significant increase in the response time of the steering motor to reach the target position.

The ASOC robot experiences significant speed fluctuations in motions greater than 90 degrees due to the slow speed of the wheel set transition. In the $180°$ scenario, this transition process takes nearly 1 second, during which the ASOC robot experiences significant tracking errors (see Fig. 9(c)).

The power consumption of the 4WS4WD and ASOC robots is relatively similar in omnidirectional motion. The ASOC robot exhibits the highest power consumption. Due to the use of fewer actuators, the Mecanum robot has the lowest average power consumption (see Fig. 9(e)). Overall, the 4WS4WD robot exhibits the best performance in kinematic isotropy, while the Mecanum robot has good path tracking ability but has issues with maintaining speed. The ASOC robot performs poorly in scenarios involving large turning angles.

B. Path Following

The reference path used for evaluation is a continuous but non-differentiable path. The robot needs to go through four 90-degree corners in succession (see Fig.10(a-i) and (b-i)). This path not only demonstrates the omnidirectional movement characteristics of the robot, but also evaluates the limitations of the three types of robots. The experiments were performed on a flat indoor terrain. The target velocities are 0.5 m/s, 1.0 m/s, 1.5 m/s, 2.0 m/s, and 2.5 m/s.

From the average error and standard deviation graphs at various speeds (Fig. 10(c)), it can be observed that increasing the target speed exponentially increases the average and standard deviation of tracking errors. Tagliavini et al. [15] proposed using kinematics-based velocity space under wheel speed constraints as an indicator for evaluating omnidirectional robots. However, in practical experiments, due to the limited adhesion provided by the tires, there are significant differences in the motion capabilities of each robot before approaching the wheel speed limit. Next, two scenarios with relatively low speed of 1.0m/s and relatively high speed of 2.0m/s are selected for analysis. At a speed of 1.0m/s (see Fig.10(a)), the tracking performance of all three platforms is good, with the 4WS4WD robot exhibiting the smallest oscillation at the turning position. During the rapid change of direction of the ASOC robot's wheel group, the orientation of the ASOC robot changed significantly. The speed tracking of the three platforms was not significantly affected by the turning angle. At a speed of 2.0m/s (see Fig.10(b)), it can be seen from the speed change graph of the three robots during tracking that even when running at high speeds, the omnidirectional robot's overall speed does not completely stop when tracking non-differentiable trajectories. From the trajectory, Mecanum has the largest deviation at corners and the longest recovery time. The 4WS4WD robot experiences significant speed loss after turning (see Fig. 10(b-iii)). Under low-speed conditions (0.5m/s and 1.0m/s), the Mecanum robot have the lowest energy consumption. However, at higher speeds, the Mecanum robot doesn't have an efficiency advantage. Compared to the Mecanum and ASOC robot,

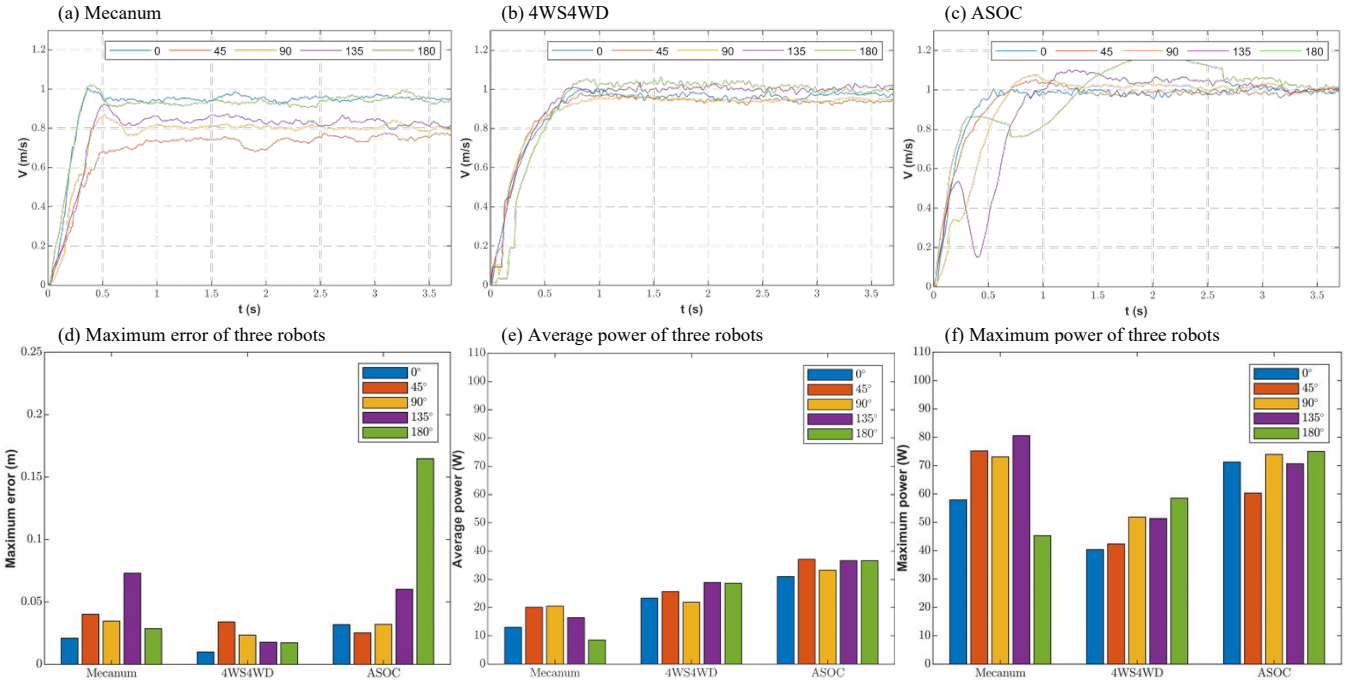


Fig. 9. Figure (a-c) show the variation of motion speed over time for three robots in isotropic experiments. The tracking errors of the Mecanum and 4WS4WD robots are close, while the ASOC robot exhibits significant oscillations during initial motion in large turning scenarios, but eventually converges within 3 meters of tracking error. Figure (d) displays the maximum error of the three platforms. Figure (e) displays the average power consumption of the three platforms. Figure (f) displays the maximum power consumption of the three platforms.

the 4WS4WD robot performs the best under high-speed conditions, with the lowest power consumption and tracking error (see Fig. 10(e)) . Since the wheels of the ASOC are driven independently by 8 motors, the ASOC robot has a higher power output limit at high speeds (see Fig. 10(f)) and can maintain a higher speed when turning.

IV. Conclusion

This paper employs experimental and comparative methods to study the comprehensive characteristics of different configurations of omnidirectional robots. Based on two dimensions, namely the configuration of the wheels and the platform's motion characteristics, we classify existing omnidirectional robots into four categories. We then build three representative platforms. In order to improve the reliability of the test results, multiple variables are standardized in the design of these robots, and forward kinematics is used for joint velocity control. In the isotropic experiments, we found that holonomic omnidirectional robots exhibit anisotropic motion effects due to their configuration characteristics and installation accuracy. The 4WS4WD omnidirectional robot exhibits excellent isotropic motion effects when it can turn quickly. In the path tracking experiments, all three types of omnidirectional robots can maintain low tracking errors under low-speed conditions. However, due to the limited adhesion provided by the wheels, the robots cannot maintain low tracking errors when changing path at high speeds. As the target speed increases, the tracking error increases exponentially. The adhesion coefficient of special wheels is usually lower than that of traditional wheels, resulting in poorer path tracking capabilities. This paper summarizes the performance differences of different types of omnidirectional robots in different dimensions, which can provide convenience for the selection of omnidirectional robot designs. The main argument of this paper is based on qualitative analysis, while not taking into account the effects of unified variables (load, size, etc.) on different types of robots. In the future, there are plans to use quantitative analysis to evaluate the comprehensive performance of omnidirectional robots.

Acknowledgment

This work was supported in part by the Science, Technology and Innovation Commission of Shenzhen Municipality under Grant No.ZDSYS20200811143601004.

References

[1] C. Sprunk, B. Lau, P. Pfaff, and W. Burgard, "An accurate and efficient navigation system for omnidirectional robots in industrial environments," *Autonomous Robots*, vol. 41, pp. 473–493, 2017.

[2] X. Gao, J. Li, L. Fan, Q. Zhou, K. Yin, J. Wang, C. Song, L. Huang, and Z. Wang, "Review of wheeled mobile robots' navigation problems and application prospects in agriculture," *IEEE Access*, vol. 6, pp. 49 248–49 268, 2018.

[3] G. Witus, "Mobility potential of a robotic six-wheeled omnidirectional drive vehicle (odv) with z-axis and tire inflation control," in *Unmanned Ground Vehicle Technology II*, vol. 4024. SPIE, 2000, pp. 106–114.

[4] G.-Z. Yang, B. J. Nelson, R. R. Murphy, H. Choset, H. Christensen, S. H. Collins, P. Dario, K. Goldberg, K. Ikuta, N. Jacobstein *et al.*, "Combating covid-19—the role of robotics in managing public health and infectious diseases," p. eabb5589, 2020.

[5] S. Jiang, Z. Li, S. Lin, W. Shi, Z. Zhu, H. Che, S. Yin, C. Zhang, and Z. Jia, "Design, control and experiments of an agile omnidirectional mobile robot with active suspension," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2022, pp. 913–918.

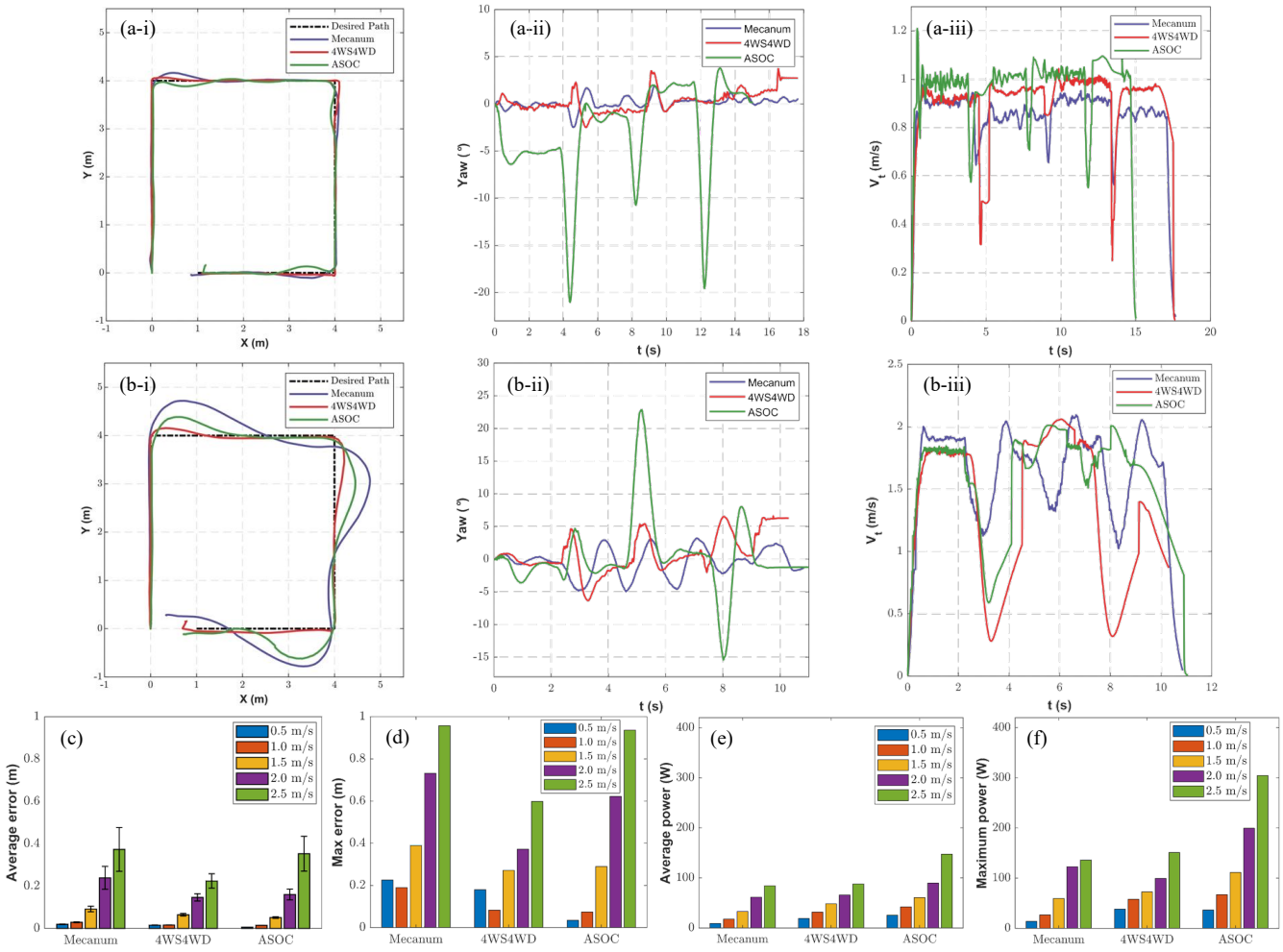


Fig. 10. Figure(a) shows the results of the three robots operating at a speed of 1.0m/s. Figure (a-i) shows the trajectory, Figure (a-ii) shows the yaw angle variation over time, and Figure (a-iii) shows the speed variation over time. Figure(b) shows the results of the three robots operating at a speed of 2.0m/s. Figure (b-i) shows the trajectory, Figure (b-ii) shows the yaw angle variation over time, and Figure (b-iii) shows the speed variation over time. Figure (c) shows the average error and standard deviation of the three robots at various speeds. Figure (d) shows the maximum error of the three robots at various speeds. Figure (e) shows the average power consumption of the three robots at various speeds. Figure (f) shows the maximum power consumption of the three robots at various speeds.

[6] F. Ferland, L. Clavien, J. Frémy, D. Létourneau, F. Michaud, and M. Lauria, "Teleoperation of azimut-3, an omnidirectional non-holonomic platform with steerable wheels," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 2515–2516.

[7] G. Runge, G. Borchert, P. Henke, and A. Raatz, "Design and testing of a 2-dof ball drive: An omnidirectional wheel for mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 81, pp. 195–213, 2016.

[8] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, 2019. [Online]. Available: https://doi.org/10.1177/1729881419839596

[9] G. Campion, G. Bastin, and B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE transactions on robotics and automation*, vol. 12, no. 1, pp. 47–62, 1996.

[10] H. Taheri, B. Qiao, and N. Ghaeminezhad, "Kinematic model of a four mecanum wheeled mobile robot," *International journal of computer applications*, vol. 113, no. 3, pp. 6–9, 2015.

[11] C.-W. Wu and C.-K. Hwang, "A novel spherical wheel driven by omni wheels," in *2008 International Conference on Machine Learning and Cybernetics*, vol. 7. IEEE, 2008, pp. 3800–3803.

[12] M. West and H. Asada, "Design of ball wheel mechanisms for omnidirectional vehicles with full mobility and invariant kinematics," 1997.

[13] B. Adamov, "Influence of mecanum wheels construction on accuracy of the omnidirectional platform navigation (on exanple of kuka youbot robot)," in *2018 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*. IEEE, 2018, pp. 1–4.

[14] M.-H. Lee and T.-H. S. Li, "Kinematics, dynamics and control design of 4wis4wid mobile robots," *The Journal of Engineering*, vol. 2015, no. 1, pp. 6–16, 2015.

[15] L. Tagliavini, G. Colucci, A. Botta, P. Cavallone, L. Baglieri, and G. Quaglia, "Wheeled mobile robots: State of the art overview and kinematic comparison among three omnidirectional locomotion strategies," *Journal of Intelligent & Robotic Systems*, vol. 106, no. 3, p. 57, 2022.

[16] M. J. A. Safar, "Holonomic and omnidirectional locomotion systems for wheeled mobile robots: A review," *Jurnal Teknologi*, vol. 77, pp. 91–97, 12 2015.

[17] L. Xie, C. Scheifele, W. Xu, and K. A. Stol, "Heavy-duty omni-directional mecanum-wheeled robot for autonomous navigation: System development and simulation realization," in *2015 IEEE International Conference on Mechatronics (ICM)*, 2015, pp. 256–261.

A Robot for Cleaning and Sorting Garbages in the Home Environment

Jiajun Long, Ximan Zhang, Yikai Zheng, Jie Luo, Sinrithy Vong

***Abstract*—This paper mainly focuses on the task requirements of mobile robots identifying different kinds of garbage through object recognition in household environment, and can throw the garbage into the corresponding trash can. It respectively carries out in-depth exploration on the mapping navigation and object recognition of the robot, and combines the two in the later stage, and proves the feasibility of the task through three experiments: In object recognition experiment, the correct rate of object recognition in simulation environment can reach more than 90%. The similarity between the map generated in the navigation experiment with building map and the ground truth can reach more than 70%, and the effect is good in the integrated experiment. Finally, some solutions to the task limitation and prospects for the future are put forward.**

I. Introduction

A. Detailed Introduction

Today, with the booming service robot industry, the cleaning robot stands out as a very important part of it, and existing robots have already appeared in some people's house. The current cleaning robots mostly use SLAM to sense the environment, identify home environment through cameras to avoid collisions, and clean up the ground.

This project focuses on mobile robots similar to cleaning robots. It is hoped that on the basis of meeting part of the goals of cleaning robots, mobile robots can perceive the environment through SLAM in home environment and identify different kinds of garbage through object recognition algorithm. Secondly, it can be absorbed through the suction cup of the robot. Finally, the process of throwing the corresponding garbage into the corresponding trash can.

The objectives of this project are described in detail below:

In a defined home environment, randomly placed some household items, such as sofa, chair, table, etc. There is one or more trash cans in this household environment. The robot of this project needs to identify furniture and garbage during the identification and cleaning of the overall environment, throw one or more garbage randomly appearing on the ground into the corresponding trash can and then continue the cleaning work.

B. Related Work

1) SLAM and Navigation: According to the mathematical optimization framework used, laser SLAM can be divided into two categories: filter-based and graph-based laser SLAM. Filter-based methods include Hector SLAM, Fast-SLAM, Gmapping, etc. While graph-optimized methods include Karto SLAM, Large-SLAM, etc.

These methods all have their own advantages and disadvantages. In the "An evaluation of 2D SLAM technologies available in robot operating system"[1], the 2D laser SLAM effects under ROS were compared. The article mentioned that hectorSLAM is based on an optimized algorithm (solving the least squares problem), which does not require an odometer and can adapt to uneven air or ground conditions. However, the selection of initial values has a significant impact on the results, so it is required that the radar frame rate be high. The Gmapping algorithm adopts the RBPF method, which can build indoor environment maps in real-time. In small scenes, the computational complexity is low, and the map accuracy is high, requiring low scanning frequency for LiDAR. However, as the environment increases, the memory and computational complexity required for building maps will become huge, so Gmapping is not suitable for large-scale scene composition. The Sparse Pose Adjustment (SPA) adopted by Karto SLAM in the ROS environment is related to scanning matching and closed-loop detection. The more landmarks there are, the greater the memory requirement need, and the greater the advantages of other methods in mapping in the larger environment.

Based on the open-source mapping code in the ROS environment and combined with our proposed goal of building and navigating indoor small environments by a sweeper, Finally, we chose the Gmapping algorithm as our sweeping robot mapping algorithm.

2) Object Recognition: Object recognition technology is the foundation of the field of artificial intelligence. This project understands YOLO and its subsequent advanced versions through literature[2], introduces the development process of YOLO algorithm, and summarizes the methods of object recognition and feature selection.

In another paper[3], a hierarchical programming algorithm is proposed that can efficiently calculate an optimal plan for finding target objects in a large environment. In the house environment of this project, the increase in the number of moving operations and the number of objects leads to a significant increase in spatial complexity. In this paper, a good hierarchical programming solution is provided, effectively reducing a large problem to a small one by breaking it down into a set of low level in-container programming problems and a high level critical location planning problem utilizing low level programming. This project absorbs its experience and tries to apply it to the project.

II. Dataset and Environment

A. Model

Fig. 1. Physical and simulation drawings

In terms of simulation model selection, this project has constructed the simulation model in the gazebo environment by referring to the TurtleBot3 Burger model, as shown in Fig. 1.

This project sets the robot to be $2kg$, with a radius of $0.15m$, and two wheels with a diameter of $0.064m$. The robot imitates the motion control system of TurtleBot3 Burger and uses the differential wheel to achieve omnidirectional motion. In terms of speed control, this project sets the maximum speed of the robot to be $0.5m/s$, the maximum acceleration of the two wheels to be $1m/s^2$, the maximum angular speed to be $1rad/s$, and the maximum angular acceleration to be $1rad/s^2$. We are equipped with IMU sensors to detect and measure acceleration, tilt, impact, vibration rotation and multi degree of freedom (DoF) motion make it easy to solve navigation, orientation, and motion vector control. Use LiDAR to detect the distribution of obstacles around the robot, achieving functions such as mapping and obstacle avoidance. The LiDAR is set to sample 500 particles and the detection radius is set to $0.1m$-$6m$. Equipped with a depth camera and RGB as object recognition tools, the camera's capture range is set to $10m$, and the captured photo pixels are 2560 $\times$ 1440. There is a vacuum scraper installed below the robot's site to absorb the recognized garbage, simulating the cleaning of the home environment by a cleaning robot.

B. Environment

In the simulation environment construction, this project used Gazebo to build a home environment, which can accurately and efficiently simulate the robot's working function in complex indoor and outdoor environments. Considering the robot's cruising time and the general size of the home environment, our environment size is $20m \times 10m$. In this environment, this project designed some obstacles to test the navigation and obstacle avoidance ability of the sweeping robot, as shown in Fig. 2, then divide the environment into three parts: living room, kitchen, and bedroom. In the living room, there are sofas, trash cans, tables, etc. There are dining tables and chairs as obstacles in the kitchen area, and fitness equipment, desks, etc. are set up as obstacles in the bedroom. At the same time, this project will add objects such as bowls, water bottles, water cups, beds, balls, tables and chairs, refrigerators, wardrobes, barbells, etc. to the simulation environment. These objects are used to verify the robot's object recognition ability.

Fig. 2. A simulation of house

C. Dataset

In this project, MSCOCO data set is mainly used, which is a large image data set developed and maintained by Microsoft, mainly used for target detection, target segmentation and image description, and mainly has the following characteristics:

- Object Segmentation
- Superpixel stuff segmentation
- 1.5 million object instances
- 80 object categories

In this project, all categories of coco data set were not used, but six items in the data set were mainly used for identification. The picture features of the data set were clear, which made it easier for training and testing, as shown in Fig. 3.

Fig. 3. The picture shows the six types of objects selected by the project for identification in the coco dataset, from left to right, and from top to bottom: giraffe, horse, cat, cup, bottle and bowl. The figure is one of the training pictures of these objects in the coco dataset.

Objects in this project are mainly divided into three categories. The first category is objects that need to be grabbed. In this project, bottles, bowls and cups are mainly grabbed. The second category is the auxiliary identification items of different trash cans. By pasting the images of giraffe, cat and horse on the trash can, different trash cans can be identified. This method is used to complete the auxiliary identification of garbage cans, avoiding the embarrassment of having no corresponding category of garbage cans in the data set. The third type of furniture independent of the above six kinds of garbage can also be identified, including tables, chairs, sofas, etc., but no other operations are made on them.

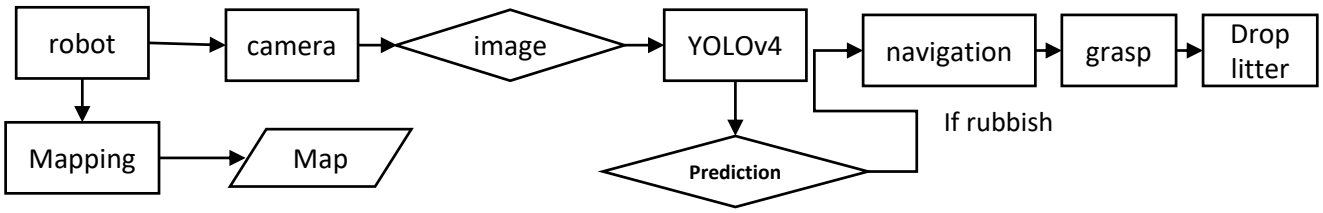


Fig. 4. This picture shows the overall work flow of the project. After the robot is in the home environment, it scans and builds a map for the overall environment, and saves the map. The environment is detected by the RGB camera of the robot, and the images captured by the camera are imported into the object recognition algorithm for recognition. The robot will autonomously navigate to the garbage place, grab the garbage through the suction cup, navigate to the garbage can according to the location of the garbage can, close the suction cup, and continue cleaning after the garbage is put down.

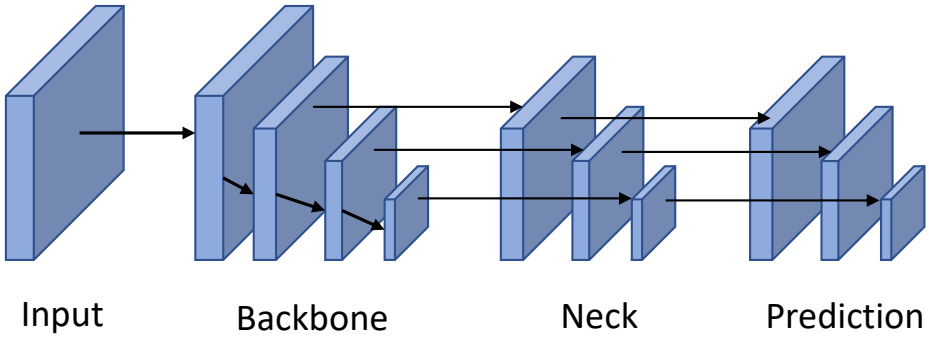


Fig. 5. The figure shows the whole process of YOLOv4 algorithm, which is mainly divided into four steps.

III. Methods

At the beginning, the overall work flow of this project is summarized, as shown in Fig. 4. The process of SLAM mapping, object recognition and navigation is combined. The respective steps are described below.

A. Object Recognition

1) Algorithm Framework: The following is an introduction to the object recognition algorithm framework used in this project.

YOLO (You Only Look Once) is an object detection method. By inputting the whole picture, detection information of multiple objects can be output, and the category and location of objects in the picture can be identified. In our project, we refer to and use YOLOv4 for object recognition[4]. YOLOv4 adds many practical techniques on the basis of YOLOv3, which greatly improves the speed and accuracy. The whole frame of YOLOv4 is shown in Fig. 5. This paper will interpret YOLOv4 from the Input, BackBone, Neck and Head output layers.

The first layer is the input, where 608×608*3 images are input for image preprocessing. Three algorithms are used to enhance this process. Firstly, Mosaic data enhancement algorithm is used to improve the training speed of the model and the accuracy of the network.The second is CmBN, which uses the ”information at the time of the current iteration” for normalization, and implements the operation of enlarging the batch size without additional compensation.The third is SAT (Self Adversarial Training) self adversarial training, which is a new data enhancement method. Using adversarial generation can improve the weak link in the learning decision boundary and improve the robustness of the model.[5]

The second layer is the BackBone, that is, the backbone network, uses the CSPDarknet53 network to extract features and simultaneously uses the Mish activation function, Dropblock regularization, and CSP cross-stage partial connection method. CSPDarknet53 is based on Darknet53 and using CSPNet[6] for reference, which make the model not only guarantees the reasoning speed and accuracy, but also reduces the size of the model, with strong accuracy in the field of target detection. Second, Mish function and ReLU, Swish and other activation function is very similar, but the different data sets can be in a lot of depth in the network have better performance, better accuracy and generalization of reference.[7] Here's Mish's function:

$$y = x \times \tanh((\ln(1 + \exp_x)) \tag{1}$$

The third is Dropblock regularization algorithm[8], which is used to solve the problem of model overfitting, mainly through

the deactivation of the local region.

The third layer is Neck, in YOLOv4 mainly added SPP module and FPN + PAN structure. The SPP module adopts the maximum pooling mode of 1x1, 5x5, 9x9 and 13x13 to carry out multi-scale feature fusion. Then, by using the method of FPN (Feature Pyramid Networks), the high resolution of low-level features and high semantic information of high-level features can be used to achieve the prediction effect through the fusion of features of different layers. Finally, Path Aggregation Network (PAN) was used to fuse the feature information of different size map to obtain accurate information.

The Head of YOLOv4 is the same as that of YOLOv3. The scale of prior frame is extracted by clustering and the position of prediction frame is constrained. Compared with YOLOv3, the loss function $CIOU_{Loss}$ during training and $DIOU_{nms}$ screened by prediction frame are mainly improved.

2) Executive Objective: This paper aims at the cleaning robot in the home environment, so the object recognition mainly includes the objects that may appear in the home environment, such as sofas, tables, chairs, etc., but also includes the garbage that we need to identify, such as cups, bottles and bowls. The primary goal is to ensure that the algorithm, after training through images in the real world, can accurately recognize objects in the simulation environment with high accuracy. Secondly, it is necessary to judge the position and direction of the object in the image representation scene.

B. SLAM and Navigation

1) SLAM: The general process of the Gmapping algorithm is to use the map and motion model from the previous time to predict the current pose, then calculate weights based on sensor observations, resample, update the particle map, and so on. It is based on particle filtering and can be roughly divided into the following four parts to complete. DrawFromMotion, ScanMatch, UpdateTreeWeights, Resample[9].

DrawFromMotion: The state of the particle at the current moment is first updated by the motion model, and Gaussian sampling noise is added to the initial value to perform a rough state estimation. Shown as Fig. 6

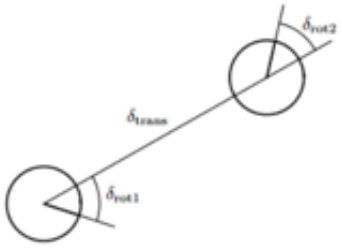


Fig. 6. Odometry model: The robot motion in the time interval [t-1,t] is approximated by a rotation rot1, followed by a translation trans and a second rotation rot2. The turns and translation are noisy.

Use the following algorithm to sample motion, shown as Fig .7

Algorithm sample_motion_model_odometry(u_t, x_{t-1}):

$\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$

$\delta_{trans} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$

$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$

$\hat{\delta}_{rot1} = \delta_{rot1} - \textbf{sample}(\alpha_1 \delta_{rot1} + \alpha_2 \delta_{trans})$

$\hat{\delta}_{trans} = \delta_{trans} - \textbf{sample}(\alpha_3 \delta_{trans} + \alpha_4 (\delta_{rot1} + \delta_{rot2}))$

$\hat{\delta}_{rot2} = \delta_{rot2} - \textbf{sample}(\alpha_1 \delta_{rot2} + \alpha_2 \delta_{trans})$

$x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$

$y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$

$\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

return $x_t = (x', y', \theta')^T$

Fig. 7. First, obtain odometer readings to obtain motion control information,then add Error term to the variation of motion control, add this error to thepose at time t-1, and calculate the pose at time t

ScanMatch: Based on the predicted posture of the motion model, move the predicted posture towards six states: negative x, positive x, negative y, positive y, left rotation, and right rotation. Calculate the matching score for each state, and select the posture corresponding to the highest score as the optimal posture.

Algorithm likelihood_field_range_finder_model(z_t, x_t, m):

$q = 1$

for all k *do*

if $z_t^k \neq z_{\max}$

$x_{z_t^k} = x + x_{k,\text{sens}} \cos\theta - y_{k,\text{sens}} \sin\theta + z_t^k \cos(\theta + \theta_{k,\text{sens}})$

$y_{z_t^k} = y + y_{k,\text{sens}} \cos\theta + x_{k,\text{sens}} \sin\theta + z_t^k \sin(\theta + \theta_{k,\text{sens}})$

$dist = \min_{x',y'} \left\{ \sqrt{(x_{z_t^k} - x')^2 + (y_{z_t^k} - y')^2} \,\middle|\, \langle x', y' \rangle \text{ occupied in } m \right\}$

$q = q \cdot \left(z_{\text{hit}} \cdot \textbf{prob}(dist, \sigma_{\text{hit}}) + \frac{z_{\text{random}}}{z_{\max}} \right)$

return q

After obtaining the optimal particle pose, the particle sampling range can be changed from the flat and wide area to the peak area L represented by the LiDAR observation model, and the new particle distribution can be closer to the real distribution. Shown as Fig .8

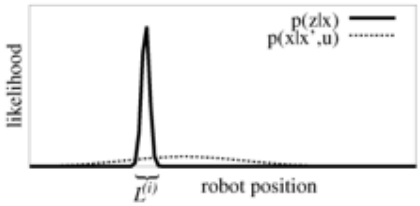


Fig. 8. The two components of the motion model. Within the interval L, the product of both functions is dominated by the observation likelihood in case an accurate sensor is used

By Scann and Match, the peak area represented by L was

found. After finding the peak area represented by L, it is necessary to determine the mean and variance of the Gaussian distribution represented by the peak area. By randomly sampling K points in L, calculate the mean and variance based on the odometer and observation model of these K points, as shown in the following equation.

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} x_j \cdot p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1})$$

$$\Sigma_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1}) \cdot (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T$$

with the normalization factor

$$\eta^{(i)} = \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1}).$$

UpdateTreeWeights: For each particle, we need to calculate its weight for use in subsequent resampling steps.The weight describes the difference between the target distribution and the proposed distribution. The formula for calculating the weight is

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) \\ &= w_{t-1}^{(i)} \cdot \int p(z_t \mid m_{t-1}^{(i)}, x') \cdot p(x' \mid x_{t-1}^{(i)}, u_{t-1})\, dx \\ &\simeq w_{t-1}^{(i)} \cdot \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1}) \\ &= w_{t-1}^{(i)} \cdot \eta^{(i)}. \end{aligned}$$

ReSample: Before performing resampling, the weight of each particle is calculated. Sometimes, due to high environmental similarity or measurement noise, the weight of particles in the near correct state may be smaller, while the weight of particles in the wrong state may be larger.

In the Gmapping algorithm, the author uses the measure of weight deviation to determine resampling.

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}^{(i)}\right)^2},$$

The larger the Neff, the smaller the difference in particle weight. When Neff drops below a certain threshold, it indicates a significant difference between the distribution of particles and the true distribution. At the particle level, it appears that some particles are very close to the true value, while many particles are far from the true value. At this point, resampling happens.

2) Navigation: For the navigation package configuration of ROS, navigation is a 2D navigation package set that outputs target position and safe speed for mobile robots by receiving odometer data, tf coordinate transformation tree, and sensor data. According to the official structural framework diagram provided by ROS, the navigation package is divided into three parts: AMCL and Move-Base and Map-sever.(see Fig. 9)

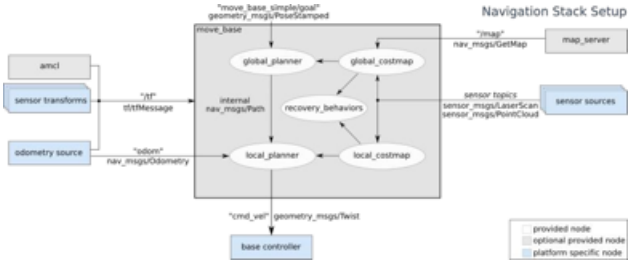


Fig. 9. Navigation framework diagram

Move-Base receives information such as odometer messages, robot posture and position, map data, and performs global and local planning within the node. Local planning is aimed at changing one's path according to changes in the environment during the navigation process, achieving automatic obstacle avoidance. AMCL achieves probability self-determination based on existing maps and LiDAR. It is used to confirm and output the robot's position in the current map. The principle of this positioning method is as follows: each sample stores position and direction data representing the robot's posture. Particles are randomly sampled, and when the robot moves, particles memorize the robot's actions based on their state, using recursive bayesian estimation for resampling. Map-Sever is used to load static global maps constructed by the Gmapping algorithm into a map server.

IV. Experiments

A. Object Recognition

In the first small experiment, the project focused on the mAP (Mean Average Precision) of the test set, i.e. the exact results. mAP is a commonly used evaluation index in target detection models. The first things to know are Precision and Recall. By precision and recall, AP can be known step by step. AP is for a certain category of all pictures. However, in the data set of this project, we are not only concerned with the identification of one category, so we need to use mAP to measure the quality of the model in all categories.

In terms of specific implementation, the original test code of YOLOv4 was modified first to obtain more parameters and results required by this project. The learned model was observed by observing mAP50, mAP75 and mAP95.

The experimental results are shown in the Fig.10 below.

Fig. 10. The figure shows the final value obtained after multiple evaluations using YOLOv4 to validate the test set of the COCO dataset.

It can be found that in the experiment, mAP50 remained at 0.566 and mAP75 at 0.401, basically consistent with the results of YOLOv4 paper, indicating high accuracy of model training.

In the second small experiment, the probability that the object recognition is correct is judged by recognizing the object in the simulation environment. Since the training mainly uses images in the real world, this experiment is used to explore the accuracy rate of object recognition in the simulation environment, laying a foundation for the smooth progress of the later experiments. The main process is to put the three pieces of garbage selected in this project into the simulation environment, and directly let the camera on the robot acquire images for object recognition through YOLOv4, and judge the recognition accuracy. The following is part of the experimental results, as shown in the Fig. 11

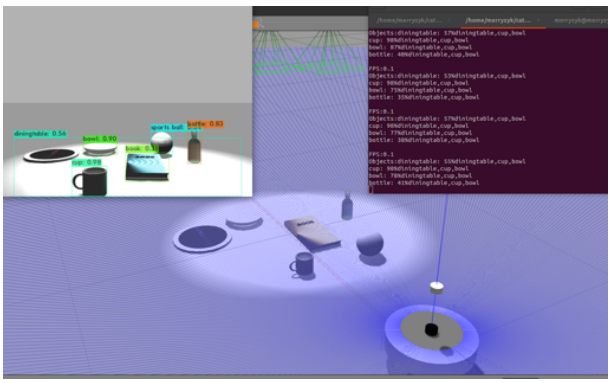


Fig. 11. The main purpose of this figure is to show the recognition accuracy of the object recognition model generated by training with real images on virtual items in the simulation environment. In addition to this figure, several sets of tests were performed, which are not listed here.

As can be seen from the pictures shown, it can be seen that the recognition of multiple items is still relatively accurate, among which the recognition accuracy of bowl, cup and bottle in this picture is high, reaching 80% or more, and the recognition rate of cup with the most obvious features even reaches 98%. Relatively speaking, book is difficult to identify accurately due to its shape. The recognition accuracy of the sports ball in the upper right corner is poor due to the light. Another object not recognized in the image is the frisbee in the upper left corner, which is difficult to recognize in gazebo images due to its shape and color. At the same time, there are some mistakes in recognition. There is no so-called dining table in the picture, but the white circle appears in the picture because the light source is too bright. But the light source is too dark and can't clearly identify other objects, which is one of the biggest problems encountered in the simulation environment.

In the third small experiment, this project hopes to compare with the object recognition algorithm popular at the same time in YOLOv4, and prospect the later YOLO algorithm. Control variable method is expected to be adopted in the experiment to observe the performance of each algorithm under the same weight, the same data set and the same training times, focusing on the size of FPS and mAP.

Due to the limitation of equipment and time in this project, the results could not be well presented. However, in the paper published by the author of YOLOv4, there are experiments comparing various algorithms, as shown in Fig. 12. Here, we further analyze the results.

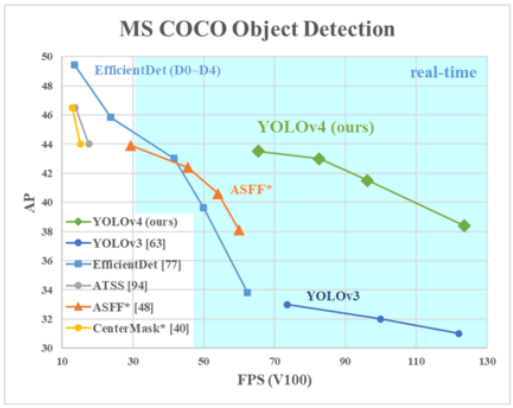


Fig. 12. The compared result

As can be seen from this image, YOLOv4 EfficientDet runs three times faster than other state-of-the-art object recognition algorithms at the time, with comparable performance, and improves AP and FPS by 10% and 12%, respectively, compared to YOLOv3.

Compared with the YOLOv4 we used, the current YOLO algorithm has become more and more mature and convenient. Now there have been YOLOv8 versions, each version has a small improvement, which is the result of continuous innovation by predecessors. Object recognition, as one of the very important components in the field of robotics and computer, will become more and more mature in the future, and the recognition accuracy will be higher and higher, and the use of human society will be greater. If you want to learn more about more advanced algorithms, please check out the updated papers in the YOLO series. And the YOLO collection doesn't stop there, with more innovations to come.

B. SLAM and Navigation

1) Task: Use simulation software gazebo to build a home environment, as shown in the figure. The cleaning robot will use the mapping algorithm to create a map based on the environment, and then use the navigation function package carried by the ROS to navigate, allowing the robot to reach the designated location.

The performance of the mapping algorithm will be evaluated by calculating the overlap between the mapping results and the predesigned map.

2) Setup: By using the open-source Gmapping algorithm and navigation package of ROS, relevant parameters are configured to achieve robot mapping and navigation in maps.

When configuring the gmapping algorithm, referring to the algorithm principles mentioned earlier, we modify and configure the following parameters, while selecting default values for the remaining parameters. The parameter 'particles

(int, default: 30)' determines the number of particles in the gmapping algorithm. Gmapping uses a particle filter algorithm, and the particles are constantly iteratively updated. Therefore, selecting an appropriate number of particles can ensure the algorithm has high speed while ensuring accuracy.The parameter 'MinimumScore (float, default: 0.0)' is the minimum matching score, which is an important parameter that determines your confidence in the laser. A higher value indicates a higher requirement for the laser matching algorithm, and the laser matching is more likely to fail and switch to using odometer data. Setting it too low can cause a lot of noise in the map.

When configuring navigation, it is mainly necessary to configure the cost map in the Move Base section. As the SLAM built map is a static map, static maps cannot be directly applied to navigation. Based on this, some auxiliary information maps need to be added to obtain auxiliary information by configuring the cost map to achieve real-time navigation obstacle avoidance function. There are two cost maps: global-Costmap and local-Costmap, the former is used for global path planning, and the latter is used for local path planning. The cost map generally has the following levels:

Static Map Layer: A static map constructed by SLAM.

Obstacle Map Layer: The obstacle information perceived by sensors in navigation.

Expansion Layer: Expand (outward) on the above two layers of the map to avoid the robot's shell colliding with obstacles.

3) Results: To evaluate the quality of the maps obtained, an analysis of the error between the generated map and the ground truth was conducted. To that end, the best fit alignment between the ground truth and the map obtained is computed, using intensity-based image registration tools. The process works as follows: Normalize two images using the image process toolbox in MATLAB, convert the size of the images to a consistent size, and obtain a grayscale image of the image. Then, compare the SSIM values (Structural Similarity Index) and RMSE values (Root Mean Square Error) of the two images to obtain the accuracy of the image. The SSIM range is [0,1], and the larger the value, the better the image quality. When two images are identical, SSIM=1. The principle of RMSE calculation is to square the difference between the real value and the predicted value, then sum and average it, and finally root it out. The smaller the RMSE value, the more similar the image is. Fig. 13 shows the results of one of the component images. Fig. 14 shows the comparison results before and after the construction of the SLAM.

C. Integrated Experiment

Finally, the integration experiment is carried out to combine object recognition with robot mapping and navigation, completing the most important step of this project. The main purpose of the experiment is to verify that the robot can remove garbage in the household environment and put it in the trash can process.

The experiment is mainly divided into two parts. The first part is to verify that the integration effect of various parts of the robot can be realized in a household environment with

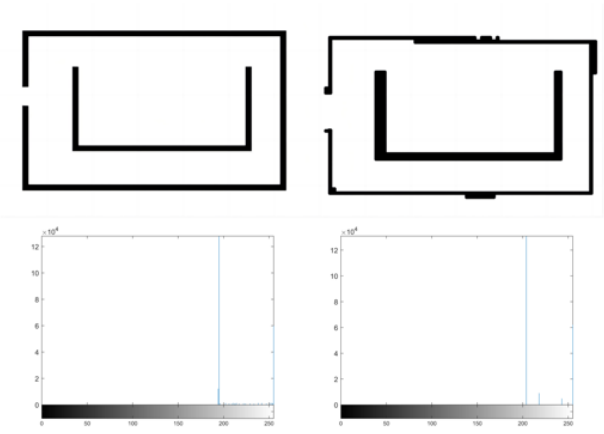
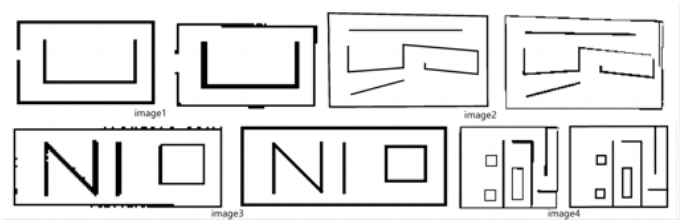


Fig. 13. The results of one of the experiments are shown here. Above the image is a comparison between the map constructed by the SLAM algorithm and the original image, while below the image are grayscale images of the two images



Simulations Experiments				
	image1	image2	image3	image4
RMSE	20.1239	91.4085	92.3797	94.5178
SSIM	0.8528	0.7035	0.6576	0.7171

Fig. 14. The upper part of the figure shows the visual comparison between the four groups of experimental Centamap and the map built by SLAM, and the lower part shows the SSIM value and RMSE value mentioned above

only one garbage and one garbage can. The specific implementation process is as follows: determine the coordinates of the garbage picked up when building the world environment, write the corresponding round-trip program through the fixed-point navigation, and realize the pick-and-place of three different garbage, identify the garbage through the object, judge the distance between the robot and the garbage through the depth camera, and then execute the corresponding round-trip program. Use object recognition function to identify garbage, then obtain the location information of the object in the scene, and then execute the corresponding pick program.

The purpose of the second part of the experiment is to verify that this project can complete the process of identifying garbage in the household environment, picking up garbage, and throwing it into the trash can. Three different kinds of garbage and three different trash cans are used in this part of the experiment, aiming to carry out more research on the basis of the first part of the experiment.

The test results are shown in the Fig. 15 and Fig. 16. During

Fig. 15. This image shows the garbage we need for object recognition

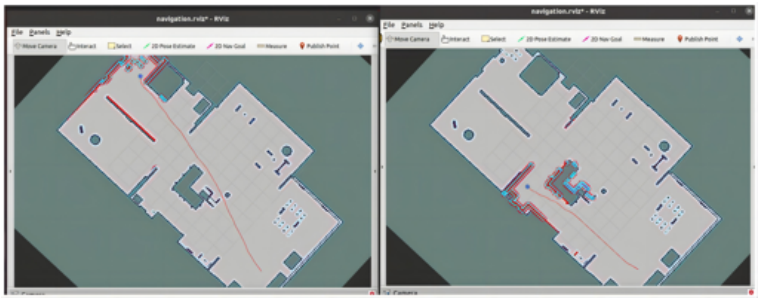


Fig. 16. the navigation path planning diagram for returning to the designated garbage storage area after picking up the garbage we have selected

the experiment, due to issues with object recognition training, the bowl garbage we designed was identified as a baseball, and the task of picking and sending it back to our designated area cannot be completed.

V. Conclusion

This project has carried out theoretical analysis, experimental verification and experimental exploration by studying the mobile robot picking up different kinds of garbage in the household environment and placing it in the corresponding trash can. Although the goal has been achieved in the end, there are still some areas for improvement.

Firstly, in terms of models and environments, models and suckers with clearer and more stable structures can be selected, and more diverse environments can be tried to meet the diversity.

Secondly, in terms of object recognition, YOLO algorithm has been developed to YOLOv8, and more advanced algorithms can be used for model training in the future, which can better improve the accuracy of object recognition. Secondly, data sets with more obvious features can be used, such as ImageNet, which not only has a large number of images, but also has more features, which can also improve the recognition effect.

Finally, in terms of navigation and mapping, more advanced mapping methods can be used to improve the mapping effect. At the same time, a more intelligent autonomous navigation system can be realized to achieve better results in completing tasks.

This project lays a good foundation for the exploration of the following tasks in this field and provides guidance for the following experiments and expansions.

Acknowledgments

Thanks to Mr. Song for the guidance of this project and to every student for their hard work.

References

[1] João Machado Santos, David Portugal, and Rui P. Rocha. An evaluation of 2d slam techniques available in robot operating system. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, 2013.

[2] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022. The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19.

[3] Yoonyoung Cho, Donghoon Shin, and Beomjoon Kim. ω^2: Optimal hierarchical planner for object search in large environments via mobile manipulation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7888–7895, Oct 2022.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[5] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan L. Yuille. Adversarial examples for semantic segmentation and object detection. *CoRR*, abs/1703.08603, 2017.

[6] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. Cspnet: A new backbone that can enhance learning capability of CNN. *CoRR*, abs/1911.11929, 2019.

[7] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *CoRR*, abs/1908.08681, 2019.

[8] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Dropblock: A regularization method for convolutional networks. *CoRR*, abs/1810.12890, 2018.

[9] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

Reinforcement learning of cluster robots based on MADDPG algorithm

Yifei Chen
12010502

Jiajun Long
12011624

Yijun Fang
12011301

***Abstract*—Reinforcement learning of multiple intelligences is very important and effective for cluster-based robot development. Based on OpenAi's multiagent-particle-envs environment, we implement the multi-intelligent body reinforcement learning MADDPG algorithm. By studying the algorithm of the paper and implementing it using pytorch according to our understanding. We also modified the critic-actor neural network. The algorithmic part of the MADDPG that we have built is able to interface with the experimental scenarios that we have built ourselves, enabling the learning of scenarios such as cooperation/competition/communication between robots. Finally, we analyze and summarize the learning effects of clustered robots performing different tasks.**

***Index Terms*—MADDPG, critic-actor network, cluster robots, multi-intelligent**

I. Introduction

Clustered robots extract engineering principles from the study of these natural systems to build multi-robot systems with comparable capabilities. In this way, cluster robots aim to build systems that are more robust, more fault-tolerant, and more flexible than individual robots, and that can better adapt their behavior to environmental changes. Implementing swarm behavior in robots requires more than applying swarm intelligence algorithms to existing robotic platforms. In fact, researchers need to completely rethink traditional robot functions such as perception, control, localization, and the design of the robot platform itself. Over the past two decades, researchers in cluster robotics have made significant progress, providing proof-of-concept for the potential of cluster robots and enabling researchers to better understand how complex behaviors emerge in nature. Nonetheless, translating this research into practice remains fraught with challenges that need to be properly addressed by researchers. Indeed, to date, only a few experiments have successfully demonstrated a large number of autonomous self-organizing robots, and there is still a gap in practical applications of clustered robots.

Research on complex behaviors of clustered robots using reinforcement deep learning is also popular nowadays. Different terrain obstacle conditions are encountered during the training process, and the optimal convergence solution is eventually found. At the same time, the grappler is trained simultaneously in this process as a dynamic factor in the system,which makes the training environment of the grappler more complex and observes its training effect. It is obtained from the way of evolutionary selection in nature.

The traditional reinforcement learning algorithms that we are most often exposed to are single-intelligent reinforcement learning algorithms, but there are also many important application scenarios that involve the interaction between multiple intelligences, such as the control of multiple robots, language communication, multi-player games, etc. [2], and the clustered robots mentioned in this paper. Open-AI's MADDPG (Multi-Agent Policy Gradient) algorithm is able to train and test multiple intelligences in a stable and efficient way [3].

II. Expiation of the algorithm

A. Reinforcement Learning Introduction

The basic concept of reinforcement learning, we believe, can be divided into 2 layers.

Layer 1: Reinforcement learning consists of two parts: the intelligences (AGENTS) and the environment (ENV). During the reinforcement learning process, the intelligent body and the environment are always interacting.

Layer 2: After an intelligence acquires a state in the environment, it uses that state to output an action, which is also called a decision. This action is executed in the environment, and the environment outputs the next state and the reward for the current action based on the action taken by the intelligence. The goal of the intelligence is to obtain as many rewards as possible from the environment.

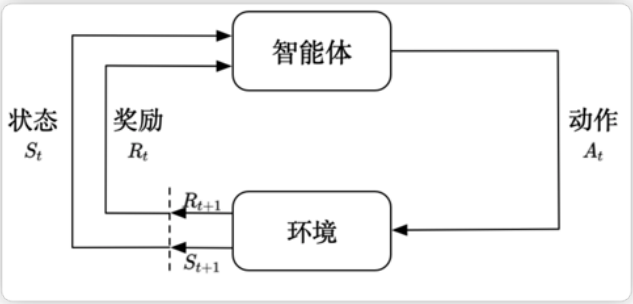


Fig. 1. The basic structure of reinforcement learning

We believe that the fastest way to cognitively reinforce learning is to use the analogy with supervised learning that we have learned. Suppose we train a classifier, such as a neural network. In order to distinguish whether the input image is a car or an airplane, the correct label information needs to be passed to the neural network during the training process. When

the neural network makes a wrong prediction, such as the input picture of a car, and it predicts that it is an airplane, we tell it directly that the prediction is wrong and the correct label should be car. Finally we write a loss function based on similar errors and train the neural network by back propagation. First of all the observation obtained by the intelligence is not independently and identically distributed, there is actually a very strong continuity between the previous frame and the next frame. The data we get are correlated time series data, which do not satisfy the independent identical distribution. Second, for each decision action, there is no label to tell the intelligence whether it is good or wrong. Only at the end of a round can the good or bad action taken be known by a REWARD. Finally, the process of acquiring an intelligence's capabilities is actually a trial-and-error exploration process. The intelligent body will be in the decision making time to explore the unknown way or use the existing experience value to make a judgment. We can see from these characteristics that the actions of the intelligent body affect the data it subsequently obtains. In the process of training an intelligent body, many times we also get data by the interaction of the intelligent body that is learning with the environment. So if the intelligences do not remain stable during the training process, it makes the data we collect very bad. We train the intelligence through the data, and if there is a problem with the data, the whole training process will fail. So a very important problem in reinforcement learning is how to keep the intelligent body's action steadily improving all the time. We can also think of stock trading as a reinforcement learning process. We can continuously buy and sell stocks and then learn how to buy and sell based on the feedback given by the market to maximize our reward.

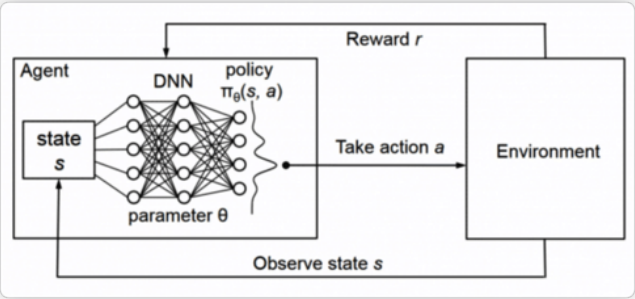


Fig. 2. The procedure of reinforcement learning

B. Influences of RL decision making

- **Policy:**The intelligence will use the strategy to pick the next action.
- **Value function:**We use the value function to evaluate the current state. The value function is used to evaluate how much of an effect the entry of an intelligence into a state can have on the rewards that follow. The larger the value function, the more favorable it is for the intelligence to enter the state.
- **Model:**The model represents the intelligence's understanding of the state of the environment, which determines how the world operates in the environment.

$$\nabla J(\theta) = E_{s\sim p^\pi, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s,a)\right] \quad (1)$$

$$L(\theta) = E_{s,a,r,s'}\left[\left(Q^*(s,a \mid \theta) - y\right)^2\right] \quad \text{with } y = r + \gamma \max_{a'} \bar{Q}^*(s', a') \quad (2)$$

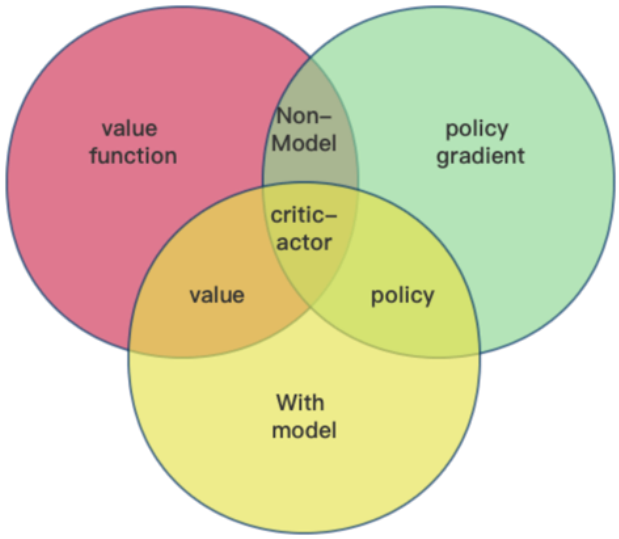


Fig. 3. The relationship between factors that influence decision making

The value-based agent learns the value function explicitly and its strategy implicitly. The policy is derived from its learned value function (the value function that follows the action that maximizes the value of this function). policy-based agent learns the policy directly (gets an optimal action in each state). Value-based reinforcement learning algorithms such as Q-learning, and policy-based reinforcement learning algorithms such as Policy Gradient (PG) algorithm are generally used in processes with large scale and continuous actions, such as robot movement and grasping.

Modeled reinforcement learning refers to building a virtual world based on the experience in the environment and learning in both the real environment and the virtual world; model-free reinforcement learning refers to learning the optimal policy without modeling the environment and interacting with the real environment directly. Currently, most deep reinforcement learning methods use model-free reinforcement learning because: model-free reinforcement learning is simpler, more intuitive, and has rich open source material, such as the AlphaGo series, which uses model-free reinforcement learning; in most of the current reinforcement learning research, the environment is static and describable, and the state of the intelligence is discrete and observable (e.g., Atari game platform), such relatively simple and deterministic problems do not require the evaluation of state transfer functions and

reward functions, and can be directly trained using model-free reinforcement learning with a large number of samples to obtain better results [4].

C. DDPG algorithm(Deep Deterministic Policy Gradient)

The strategy network plays the role of the actor, which is responsible for presenting the output to the outside world and outputting the action. q network is the commentator, which will evaluate the actor's output at each step, scoring it and estimating how much the actor's action will be rewarded in the future, i.e., estimating what the Q value of the actor's output is approximately, i.e., $Q_w(s,a)$ The actor needs to make an action based on the current state of the stage an action. The commentator is the judge, who needs to score the actor's performance based on the current state of the stage and the actor's output $Q_w(s,a)$ to adjust his strategy based on the judge's score, i.e., to update the actor's neural network parameters θ to do better next time. The reviewer, on the other hand, has to adjust his scoring strategy based on the feedback from the audience, i.e., the feedback reward from the environment, which means that he has to update the parameters of the reviewer's neural network. w The ultimate goal of the reviewer is to get the actor's performance to receive as much cheers and applause from the audience as possible, thus maximizing the total future gain.

At the very beginning of the training, the parameters of these two neural networks are randomized. So the reviewers are initially scored randomly and the actors output random actions. But since the reward of environmental feedback exists, the reviewers' scores become more and more accurate and the performance of the actors they judge becomes better and better. Since the actor is a neural network, a policy network that we want to train well, we need to compute gradients to update and optimize the parameters inside it θ. Simply put, we want to adjust the actor's network parameters so that the reviewer scores as high as possible. Note that the actor here is not concerned with the audience, it is only concerned with the judges, and it only caters to the judges' scores $Q_w(s,a)$

$$\nabla_{\theta_i} J(\mu_i) = E_{x,a\sim D}\left[\nabla_{\theta_i}\mu_i(a_i \mid o_i)\nabla_{a_i} Q_i^{\mu}(x, a_1 \ldots, a_N)_{a_i=\mu_i(o_i)}\right] \tag{3}$$

The optimal strategy for a deep Q network is to learn a good Q network, and after learning this network, we want to choose the action that maximizes the Q. The goal of DDPG is also to solve for the action that maximizes the Q value. The actor is just trying to satisfy the judges' scores, so the gradient of the optimization strategy network is to maximize the Q value, so the loss function is constructed so that Q takes a negative sign. When we write the code to put this loss function into the optimizer, it will automatically minimize the loss, that is, maximize Q.

In addition to the policy network to do the optimization, DDPG has a Q network to optimize as well. The reviewer does not know how to score at first, and it is learning step by step to give accurate scores slowly. The way we optimize

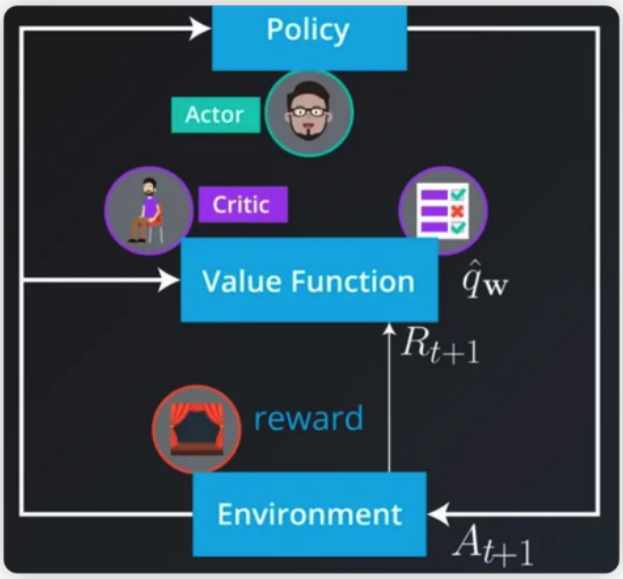


Fig. 4. The actor-critic agent neural network

the Q-network is actually the same as the deep Q-network The way we optimize the Q-network is the same, we fit the future reward Q_{target} with the real reward r and the next Q i.e. Q'. then let the output of the Q-network approximate Q_{target}. so the constructed loss function is the mean squared difference of these two values directly. After constructing the loss function, we put it into the optimizer and let it minimize the loss automatically.

When testing, we only need Actor to do it, and we don't need Critic's feedback at this point. Therefore, during training, we can add some additional information to the Critic stage to get a more accurate Q value, such as the state and actions of other intelligences, etc. This is what is meant by centralized training, i.e., each intelligence evaluates the value of the current action not only based on its own situation, but also based on the behavior of other intelligences.

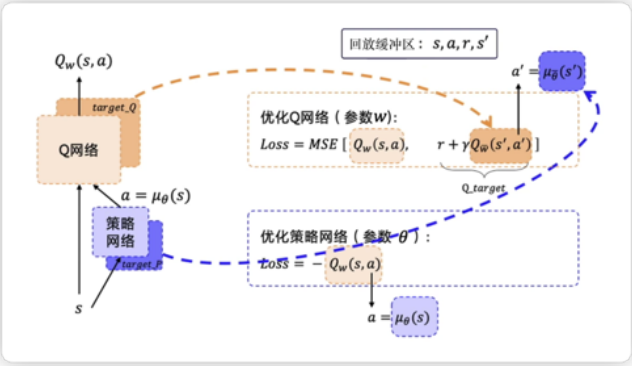


Fig. 5. DDPG with Qlearning and policy gradient

D. Introduction to MADDPG

Through the pseudo-code of MADDPG, we can see that MADDPG is similar to DDPG, but the main difference is

that in the DDPG algorithm, the input of Critic is its own observation-action data, while in MADDPG, the Critic of each agent not only inputs its own In MADDPG, the Critic of each Agent inputs not only his own observation and action information, but also other agents' action and observation (observation status) information, which means that the Critic of MADDPG can take score of the Actor from a more macroscopic perspective.

Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i} \boldsymbol{\mu}_i(o_i^j) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

  end for
end for

Fig. 6. The pseudo-code of MADDPG

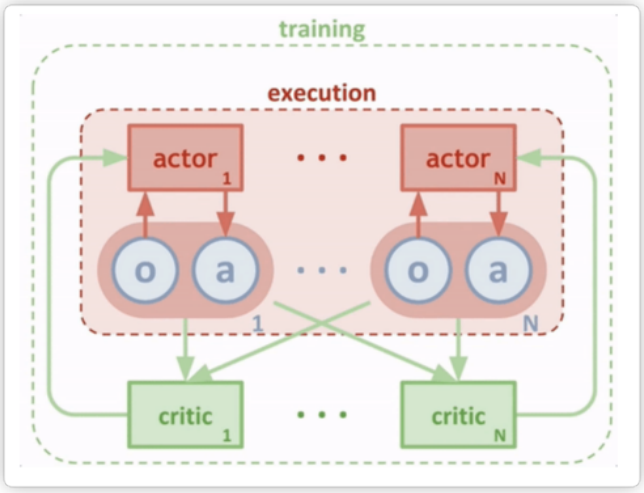


Fig. 7. The structure of MADDPG

III. Code structure

We first analysis the code of MADDPG.
For the training level of the M-set:
the number of neural networks in MADDPG:
If the number of intelligences is N
- The number of Actor networks is 2N

- Each intelligent body corresponds to an actor's training network and target network

- The number of Critic networks is 2N

- where each intelligentsia uses a separate critic training network and target network

1) The input X is the observation space of all our intelligences.
2) For each step of agent traversal is performed:
   for each agent, a policy function is used to derive our action, here we are using the neural network of the actor and a noise is added to it. This series is stored in the experience pool buffer.
3) for each agent:
   - Extract their information from the experience pool.
   - Update the critic's loss function function.
   - Update the actor loss function by policy gradient descent according to the feedback value of the critic.
4) Update the parameter values of the target neural network for optimization.

According to the analysis above, we construct our code as showed in fig.8. Its functions and roles in the entire projects is showed in the fig.9. We use a lot time to construct the buffer.py to store its learning information, in order to provide its best performance.

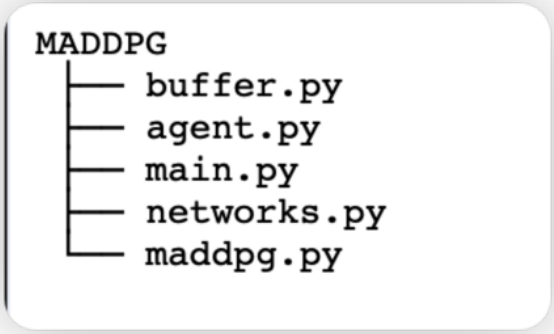


Fig. 8. The code structure of this project

IV. Training scenario Design

Multi-intelligent Particle environment is the OpenAi open source multi-intelligence learning environment built for the operation of the MADDPG algorithm proposed by OpenAi. For the convenience and ease of training, in this environment we can use the $make_env$ function to build the competition/collaboration/communication scenario we need including the number of agents, selecting the tasks they want to accomplish and the relationships (competition/collaboration/communication), such as cooperating to capture each other, maintaining distance, etc.. We can add different objects to them and set their different functions (forest, wall). State information can also be defined, mainly the coordinates/direction/speed of the smart body. The movement space of these balls can be continuous or discrete, and the discrete mode means that the movement of the smart body is discrete into several directions. In addition, in this environment, the collision between objects are able to simulate the actual collision of rigid bodies, by calculating the momentum,

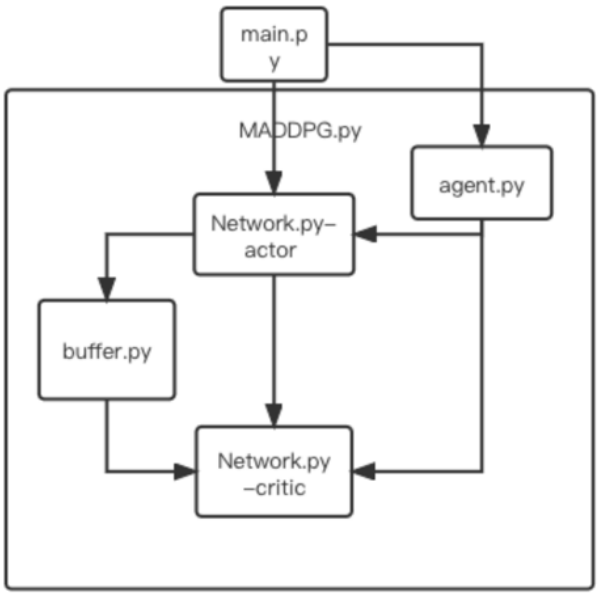


Fig. 9. The code function of this project

force, etc. to calculate the speed and displacement. To a certain extent, it is able to simulate the actual scene of our reality [6].

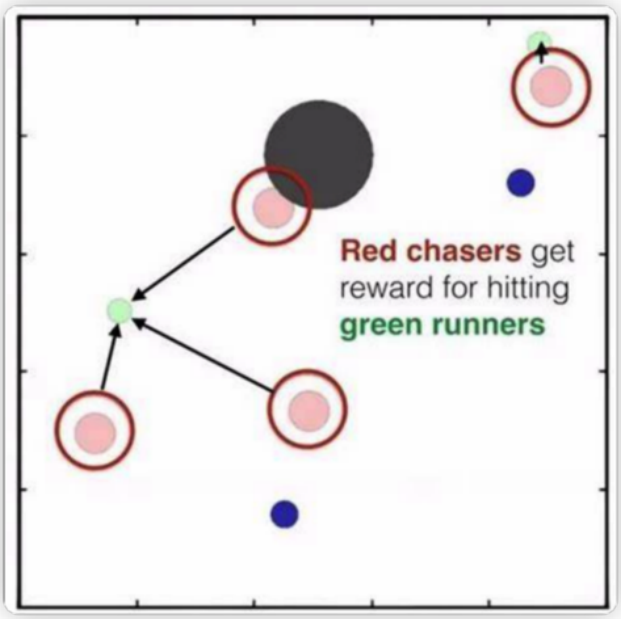


Fig. 10. The multi-particles environment from OpenAi

In order to simulate the motion and growth of clustered robots, we design two kinds of problems with different scenarios: Chasing and controlling. In the chasing problem, the fugitives are faster and wants to avoid being hit by the crowd of machines. The drones are set to a slower speed and chase down the intelligent escapees. There will be obstacles in them that will block their route. And there are also other features:

- There is food, and escapees are rewarded for being nearby.
- The bunker, when both sides are inside the bunker, is not seen by the outside world.
- The drone cluster has a ”leader” in the back of the training that can see where the drones are at all times and can communicate with other drones to coordinate the chase.

And different scenarios were designed to show different relationship between the cluster robots.

- A drone searches for a fixed point.
- A drone chases a fugitive.
- Multiple drones chasing a fugitive. Equal cooperation between each other.
- Multiple drones chasing multiple escapees. (equal relationship between each other, rewards for catching the most escapees).
- Multiple drones chasing multiple escapees. One of them is the leader drones and is able to combine the drones' location information for the chase (combined information).
- Two teams of drones chase the escapees. Each has a LEADER.

As for the controlling problem, The scenario consists of L landmarks, including a target landmark, N cooperative intelligences that know the target landmark and are rewarded according to their distance from the target, and M drones that must prevent the former from reaching the target. The drones do this by pushing each other away from the landmark and occupying it. Although the drones are also rewarded based on their distance from the target landmark, they do not know the correct target; this must be inferred from the actions of the opposing intelligences. Same as the chasing problem, we have different scenario for it.

- A drones tracking and control of a target. Distance control, keep away.
- Control of multiple drones against a target. Distance control, carry out a control and protection that can keep its relationship.
- Control of multiple drones on one target. There is a leader for the unification of information.

The specific motion performance can be seen in the video demos in the additional material.

V. Analysis of Training Performance

We have trained and simulated for the mentioned scenario. We found in our attempts that for the action space of the intelligence, the strategy he takes does not allow him to perform better convergence after a certain number of steps. This defined value we set to 25 steps. Here we look at the training performance for the two scenarios we set, where we set the number of training sessions at 10,000 for each scenario. The overall observation is that there will be better convergence properties around 5000-8000 steps. In fig.11 and fig.13, the relationship between them are equally, which is pretty simple, so their performance is relatively good.

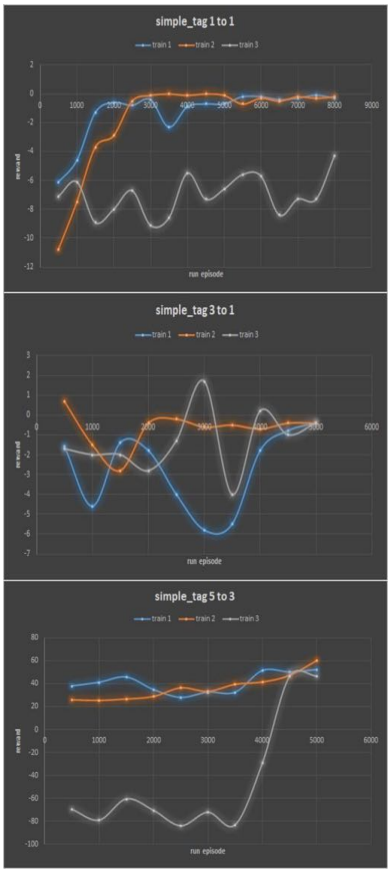


Fig. 11. The chasing problem of a)1 vs 1, b)3 vs 1, c)5 vs 3

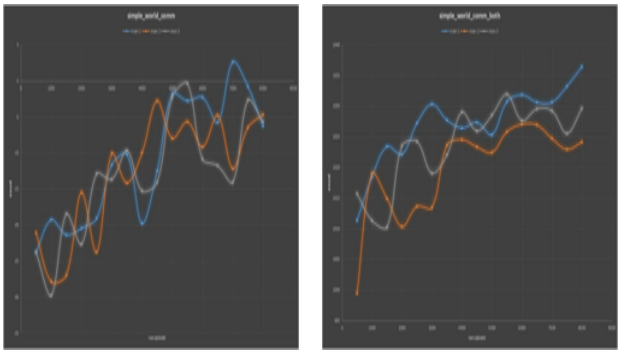


Fig. 12. Chasing problem with leaders

We first examine cooperative communication situations. In practice, we observed that listeners could learn to ignore the speaker and simply move to the middle of all observed landmarks. In fig.15 we plotted the learning curve over 2000 rounds. We read the literature and conjecture that the main reason for the failure of traditional RL methods in this (and other) multi-intelligence settings is the lack of consistent gradient signals. For example, the speaker is penalized if the correct signal is given when the speaker moves in the wrong direction. This problem becomes more severe as the time step increases: we observe that traditional strategic gradient methods can learn when the listener's goal is simply to reconstruct the observer in

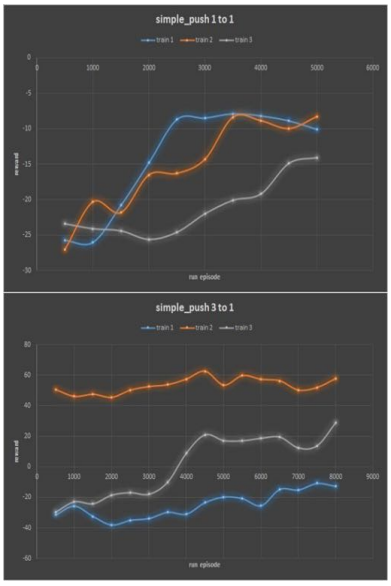


Fig. 13. The chasing problem of a)1 vs 1, b)3 vs 1.

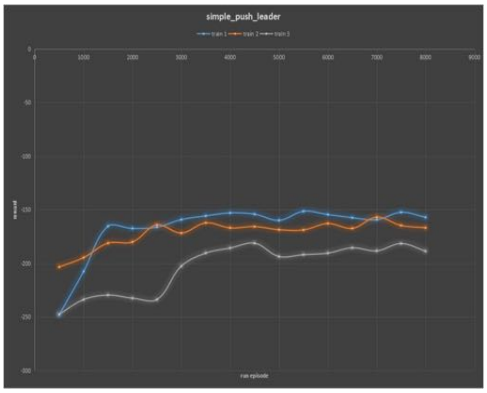


Fig. 14. Control problem with leaders

a single time step or whether the intelligences and landmarks are fixed and uniformly distributed. This suggests that many previously proposed multi-intelligence approaches for short time range scenes (e.g., [10]) may not be applicable to more complex tasks [1].

As for the scenarios with leaders, there performances is not good as the simple one. However, most of the cases can drop to a convergence in the end. From the simulation, we can see that the drones are already know the best policy to get the target in the most efficiency way, which you can see in the video demo in the support material. All in all, in terms of performance, the strategies used by the drone swarms with leaders were more effective than the drone swarms with equal relationships.

VI. CONCLUSION

The multiagent-particle-envs we used in this project is a simple environment that can only simulate and perform some simple tasks, but in the foreseeable future, human-created machines and systems will become more and more complex and capable, and in many tasks can reach the level of human

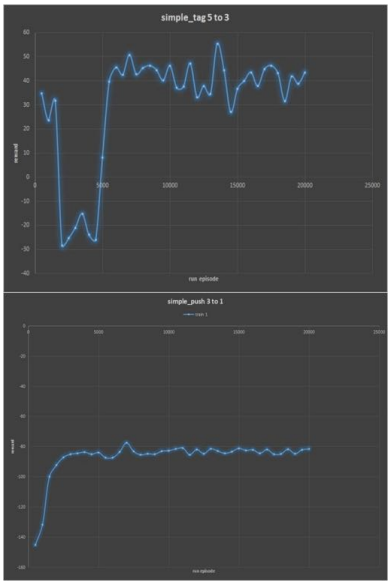


Fig. 15. Long training steps and time for two scenarios a)Tag 5 vs 3, b)Push 3 vs 1.

experts or even surpass them. Some people say that a machine is not conscious, it cannot think, it is just there mechanically executing algorithms without knowing the meaning of running them, and that the machine is not replicating human intelligence. But if we look at it from a pragmatic point of view, it doesn't matter whether the machine is conscious or not, whether it copies human "intelligence" or not, as long as the machine can do the task as required, just like a flying machine can fly because it is aerodynamic, and it is not necessary to understand how birds There is no need to understand how birds wave their wings and how their feathers work.

Drawing inspiration from the behavior of swarming insects or swarming species is valuable in many cases because the properties and behaviors exhibited by these natural clusters are the basis for arbitrarily intelligent robotic clusters: they are living evidence of the fact that self-organization can work universally, and they provide viable solutions to specific problems, such as how robotic clusters move in a coordinated fashion, distribute tasks, or make collective decisions. But our interpretation and understanding of them is also the hard part of our optimization, that is, how to focus for the tuning of our policy functions. Our direct control of the cluster is complex because understanding what the cluster is doing is very challenging due to the large amount of interaction that occurs within the cluster, which can be difficult to read for a human observer. Therefore, interpretability is critical. A possible solution might be built into the self-organizing mechanism of the cluster to allow the user to see the current state and goals of the cluster.

ACKNOWLEDGMENT

This project is completed by three members equally.

Yifei Chen is in charge of overall algorithm framework and research idea construction.

Jiajun Long is in charge of implementation of experimental simulation scenarios and visualization.

Yijun Fang is in charge of Specific implementation of the algorithm and training of the network.

REFERENCES

[1] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, Igor Mordatch, arXiv:1706.02275 [cs.LG] https://doi.org/10.48550/arXiv.1706.02275

[2] DeepMind AI reduces google data centre cooling bill by 40. https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/. Accessed: 2017-05-19.

[3] M. Abadi and D. G. Andersen. Learning to protect communications with adversarial neural cryptography. arXiv preprint arXiv:1610.06918, 2016.

[4] C. Boutilier. Learning conventions in multiagent stochastic domains using likelihood estimates. In Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence, pages 106–114. Morgan Kaufmann Publishers Inc., 1996.

[5] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforce- ment learning. IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews, 38(2):156, 2008.

[6] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: a bayesian approach. In Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pages 709–716. ACM, 2003.

[7] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In Advances in neural information processing systems, pages 271–271. Morgan Kaufmann Publishers, 1993.

[8] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. arXiv preprint arXiv:1705.08926, 2017.

[9] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. CoRR, abs/1605.06676, 2016.

[10] A. Lazaridou, A. Peysakhovich, and M. Baroni. Multi-agent cooperation and the emergence of (natural) languag[]e. arXiv preprint arXiv:1612.07182, 2016.

Visual Control Robotic Arm and Force Tactile Feedback of Gripper

Jia-jun Long, Yi-jun Fang, Yi-fei Chen, Ke-run Xiang, and Zi-meng Wang
Department of Mechanical and Energy Engineering
The Southern University of Science and Technology, Shenzhen, China

Abstract–This paper introduces the invention of the 5-DOFs robot arm's architecture, visual control by Python, calculation of forward and inverse kinematics and work, procedure of trajectory planning and MATLAB simulation, operation of pressure sensor and the future solution for the visual control robot arm. Although our team didn't work on this topic before, we decided to study and explore more contents which will be covered in this article.

***Index Terms*—MATLAB simulation, pressure sensor, trajectory planning, visual control, forward kinematics**

I. Introduction

NOWADAYS the application of robotic arm in production and life has been more and more extensive. With its high precision and low operating cost, industrial robot arm gradually replaces manual operation, and is mainly used in petrochemical, textile, steel and other large production volume and harsh production environment. It is a trend of contemporary times that robotic arm is integrated into every aspect of production and life. Research on visually controlled robotic arms is also under way.

At present, the mechanical arm is mainly used in the industrial field, but it is seldom used in daily life. In this respect, our project can do:

1. Remote control

In case of force majeure, such as the outbreak of epidemic, distance and objective natural conditions greatly hinder our work development. Our mechanical arm can get rid of space constraints and complete remote tasks by mapping human movements.

2. Fix AI issues

AI remote control cannot reproduce the power and experience exerted by human workers, and force-haptic control can accurately complete some delicate operations.

3. VR connection

The development of virtual reality and 5G has opened up a broader use of VR equipment, which was originally entertainment, to recognize human posture through visual recognition, so as to realize the interaction between virtual world and real world. Create control modes for VR+ robots. The control and drive of a single robot arm or manipulator can be applied to more human-like robots in the future, and even create VR controlled robot doppleganger.

mds
August 26, 2015

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here

Subsubsection text here.

Manuscript received December 1, 2012; revised August 26, 2015. Corresponding author: M. Shell (email: http://www.michaelshell.org/contact.html).

II. Experiment Methods

A. Mechanism Design and Assembly

First, we roughly constructed a five-axis robotic arm based on the human arm to facilitate the subsequent mapping of the human arm joints. Combining the parts of the given kit and the parts matching tutorial attached with the kit manual and the assembly steps of some robots provided by ROBOTIS official website, and considering the performance of the motor, we designed a motor parallel manipulator and completed the parts of the main part of the install of the robotic arm.

Fig. 1. The arm

Fig. 2. The robotic arm

Since the parts of the given kit cannot fully satisfy our fixation of the structure of the robotic arm part, we bought a double-headed nut to connect and fix the two parallel robotic arm parts.

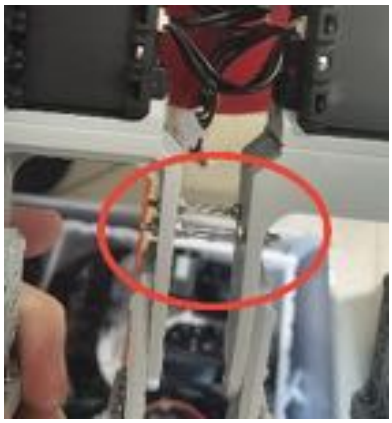


Fig. 3. The materials

Fig. 4. How to use

In order to obtain the complete modeling of the robotic arm and the quality data of each joint and reduce the part error so that the 3D printed parts and the kit parts can be well matched. We found the step files of all parts on the official website, and successfully converted the step files to solidworks part format through a file conversion website, realizing almost zero error for all parts.

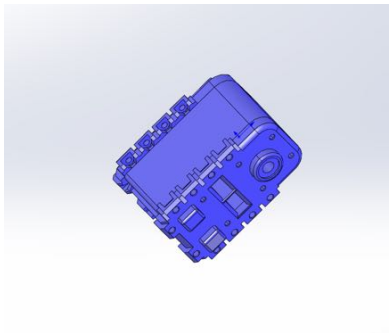


Fig. 5. Electrical Machinery

Combined with our part size, we designed a turntable to achieve the fixation of the robotic arm and the rotation of the uppermost joint. And assembled with the main body of the robot arm.

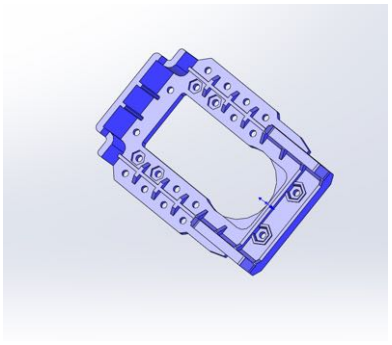


Fig. 6. Electric frame

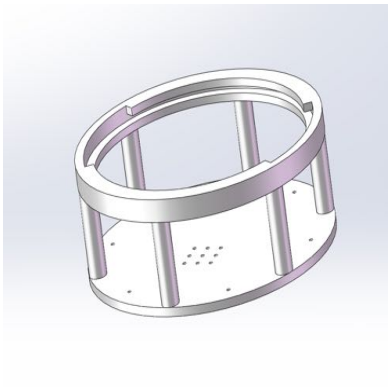


Fig. 7. Pedestal

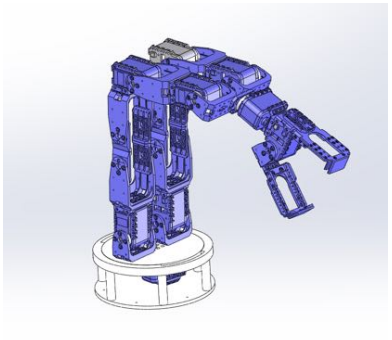


Fig. 8. The pressure sensor install1

Through 3D printing, we made the designed turntable and combined it with the robotic arm. Since the wiring length given by the kit is limited, we planned the wiring through the extension cords purchased by ourselves to ensure that each joint can be rotated to the greatest extent without being limited by the wiring length.

Since the clamping jaws installed with the parts of the kit are relatively simple, and the size of the objects to be grasped is relatively high, we have added sponges and used the concavities of the sponges to adapt to a wider variety of object sizes. At the same time, our pressure sensor needs to be fixed on a certain force surface, and the original set of jaws has no flat force surface, which cannot meet the installation requirements of the pressure sensor. Therefore, after adding a sponge, it is shallower from the outer surface of the sponge.

Fig. 9. The pressure sensor install2

Dig a hole to integrate the pressure sensor in. Although the pressure sensor is not in direct contact with a hard surface, we can still roughly measure the force of the gripper based on the force transmission.

Fig. 10. The base effect

Fig. 11. The base effect

Finally, we completed the construction of the complete model of the robotic arm.

Fig. 12. The final robot arm

B. Visual Control and Implementation

Considering that Python is quite mature in terms of visual recognition, Opencv of Python is used for visual reading and analysis. However, our final goal is to realize the mechanical arm in space, which requires the coordinates of each joint on the arm. However, considering that the laptop camera can only read the coordinates of a specific point on the plane, we need to get the three-dimensional coordinates of a specific point from continuous plane images.

After surfing the Internet, we decided to use Mediapipe, an open source application framework for multimedia machine learning models developed by Google Research. Meidapipe takes real-world 3D training data, uses AR synthetic data generation, uses ML pipes for 3D object detection, and places virtual objects into scenes with AR session data. This allows us to generate physically possible positions using camera posture, detected planes, and estimated lighting. And use lighting that matches the scene.

In addition, compared to expensive 3D cameras, Mediapipe uses a more convenient solution, requires less camera, and is more in line with our lightweight design.

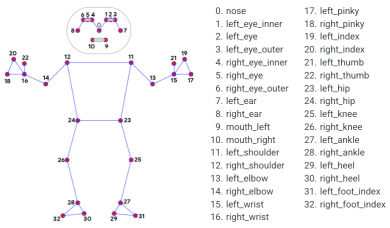


Fig. 13. The model

This is the official definition given by mediapipe to invoke various joints in the Holistic part of human body. What our team needs to establish is a five-axis mechanical arm. Here, we choose to use the right arm of human body to read data and output points 12,16,18,20,22,24 and so on. We can get the spatial coordinates of the corresponding angles, and use forward kinematics to control the manipulator.

We define a poseDetector class to encapsulate the call to MP:

It defines various methods for calling Mediapipe:

First, we define the $findPose()$ function

First, we will create a copy of the original image so that nothing of the original image is lost during preprocessing.

We will convert the blue-green-red format to red-green-blue format later because it is widely used in computer vision.

We will now use the Process function to handle posture detection for the transformed image format.

It is time to do some validation and check if landmarks are detected in the image. If landmarks are detected, we will use the draw_landmarks function to draw them on the image.

Previously we first validated before drawing the landmark, now we will check whether the display parameters allow us to display the results and input images.

If the above condition is true, the original image is displayed along with the generated image. Otherwise, it just returns the output image and results.

Second, we define a function to $findPosition()$

Build an array to store the coordinates of key nodes

Check whether self.results.pose_landmarks is empty,

Use the for loop to loop through self.results.pose_landmarks in enumerate

Get the ids of each joint and their positions and append them to the lmList.

At the same time, in order to better output data, we encapsulated the output method:

During initialization we modulate mediapipe's various parameters:

Static_image_mode: Here we need to stream video, so we set the value to False.

min_detection_confidence and min_tracking_confidence are both set to 0.5, and confidence levels are set for the detection and prediction models, which are also the initial values given on the official website.

min_tracking_confidence This is the confidence of the tracking level, that is, it will detect whether people are detected according to the confidence provided by us, so that the problem of high robustness will not occur after detection, and the stability of the system will be improved. This confidence level is ignored when static_image_mode is True.

Considering that MATLAB has a part of the function can not be implemented in Python, and through the toolkit call, can make the code more efficient and clear. We need to transmit the obtained data between Pycharm and MATLAB. Finally, after the output file is selected, MATLAB reads the data again for data transmission.

Code flow:

First we instantiate VideoCapture and set up to read the video stream directly from the camera or read the file from the file according to our needs. Then we instantiated the poseDetector and initialized all the confidence levels. After reading the image, we used our instantiated detector to call $findPose()$ method to get each joint of human body. Then call $findPosition()$ to get the three-dimensional coordinates of each key in space, then we output according to the required, and provide it to MATLAB for analysis and subsequent control.

Run into difficulties:

When debugging we also encountered many difficulties, such as the determination of coordinate system for mediapipe itself coordinate system is set among human hip to the origin, but go ahead with output when we want to coordinate system into the world, but for x axis and y axis, there are certain the size of the magnification, For the Z-axis of the connection between the camera and the human body, there was no distance calibration. Finally, we considered that direct calculation of the Angle would have no influence on which coordinate system the coordinates belonged to, so we finally decided to directly output the spatial coordinates of the coordinate system under the camera.

Fig. 14. Images obtained during testing

In addition, when encapsulating classes, I learned Python code specifications from the Internet and configured OpencV with Anaconda, which can be equipped with a virtual environment, because I had never been exposed to Python syntax. Only after many mistakes were made, our visual reading had almost no error.

Moreover, Since the results of visual recognition are all the world coordinates relative to the camera, when the human body keeps the same movement, the depth prediction will be affected by external interference factors and will change to a certain extent. Due to the size difference between the human arm and the mechanical arm, the conversion coefficient calculated by us is only based on our theoretical calculation. So a small change in the depth of recognition is likely to result in a large movement of the arm. We found these movements by testing flat motion, such as bending the elbows and wrists. When we test the stereo position, such as the rotation of the shoulder and the rotation of the forearm, the deflection of the manipulator itself will increase.

Our human posture recognition is carried out using python, while the control is using MATLAB. Because of the incompatibility of the software, the way of real-time data transmission is very rough. Real-time reading of python exported data by Matlab will cause a delay in the control of the manipulator, which will behave badly when running 4min.

C. MATLAB Simulation

1) Import of Robotic Arm

In this part, we will introduce how we use the MATLAB-robotic-toolbox made by Peter Cork to do the simulation. First,

we use MATLAB to establish a five-axis manipulator model and add dynamics parameters to the manipulator model. All the parameters were read from the model from SolidWorks. These parameters include the mass of each joint, the position of the center of mass, and the inertia tensor of each joint. Then the simulation of robotic arm is ready. These parameters will provide great convenience and help for Lagrangian dynamics calculation later.

2) Kinetic Analysis

In this part, we conducted DH convention analysis on the basis of the five-axis manipulator model established. In setting DH parameters, we encountered some difficulties. For example, under the definition of DH convention, the coordinate system of joint 4 could not be directly connected with that of joint 5, resulting in no solution. After collecting relevant information on the Internet, it is found that such a situation can be solved by setting joint 4 and joint 5 at the same coordinate origin and putting the length of the manipulator in the latter DH for output.

The following figure shows our DH parameter table and schematic diagram of the manipulator:

Joint	θ_i	d_i	a_i	α_i
1	θ_1^*	l_1	0	-π/2
2	θ_2^*	0	l_2	0
3	θ_3^*	0	l_3	0
4	θ_4^*	0	0	π/2
5	θ_5^*	l_5	0	0

Fig. 15. DH Convention

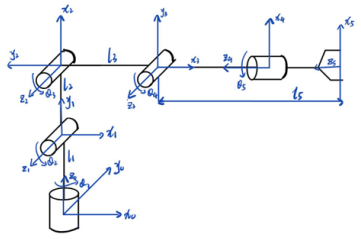


Fig. 16. Schematic diagram of robot arm

Secondly, on the basis of the problem of the manipulator model, we carry out the kinematics analysis of the manipulator.

Let's first analyze the forward kinematics.

The transformation matrix of each joint was obtained by DH parameters. Based on the 3d coordinates of each joint obtained in Python, I made angle conversion through MATLAB to obtain the change angle of each joint of the human arm. Then the angle is substituted into the transformation matrix to obtain the end-effector pose. The specific process is shown below:

$$A_1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & l_2 \cdot c_2 \\ s_2 & c_2 & 0 & l_2 \cdot s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} c_3 & -s_3 & 0 & l_3 \cdot c_3 \\ s_3 & c_3 & 0 & l_3 \cdot s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_2 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then we calculate the end-effector pose:

$T_5 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5$

$$T_5 = \begin{bmatrix} \delta_{11} & \delta_{12} & \delta_{13} & \delta_{14} \\ \delta_{21} & \delta_{22} & \delta_{23} & \delta_{24} \\ \delta_{31} & \delta_{32} & \delta_{33} & \delta_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$\delta_{11} = c_1 \cdot c_{234} + s_1 \cdot s_5$
$\delta_{12} = -c_1 \cdot c_{234} \cdot s_5 + s_1 \cdot c_5$
$\delta_{13} = -c_1 \cdot s_{234}$
$\delta_{14} = c_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_23 + l_2 \cdot c_2$
$\delta_{21} = c_1 \cdot c_{234} - s_1 \cdot s_5$
$\delta_{22} = -s_1 \cdot c_{234} \cdot s_5 + c_1 \cdot c_5$
$\delta_{23} = -s_1 \cdot s_{234}$
$\delta_{24} = s_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2$
$\delta_{31} = -s_{234} \cdot c_5$
$\delta_{32} = s_{234} \cdot s_5$
$\delta_{33} = -c_{234}$
$\delta_{34} = l_1 - l_2 \cdot s_2 - l_3 \cdot s_{23} - l_5 \cdot c_{234}$

Gripper's position is defined by vector p_5 while orientation defined by x_5, y_5, z_5provided by the forward kinematics[1]:

$$p_5 = \begin{bmatrix} c_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2 \\ s_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2 \\ l_1 - l_2 \cdot s_2 - l_3 \cdot c_{23} - l_5 \cdot c_{234} \end{bmatrix}$$

$$x_5 = \begin{bmatrix} c_1 \cdot c_{234} \cdot c_5 + s_1 \cdot s_5 \\ s_1 \cdot c_{234} \cdot c_5 + c_1 \cdot s_5 \\ -s_{234} \cdot c_5 \end{bmatrix}$$

$$y_5 = \begin{bmatrix} c_1 \cdot c_{234} \cdot s_5 + s_1 \cdot c_5 \\ -s_1 \cdot c_{234} \cdot s_5 + c_1 \cdot c_5 \\ s_{234} \cdot s_5 \end{bmatrix}$$

$$z_5 = \begin{bmatrix} -c_1 \cdot s_{234} \\ -s_1 \cdot s_{234} \\ c_{234} \end{bmatrix}$$

And then we're going to do the inverse kinematics. Solution to the inverse kinematics problem is reduced to search of the arguments q_1, q_2, q_3, q_4, q_5 based on the gripper's position and orientation. The equations above can be rewritten in the form:

$x_{5x} = c_1 \cdot c_{234} \cdot c_5 + s_1 \cdot s_5$
$x_{5y} = s_1 \cdot c_{234} \cdot c_5 + c_1 \cdot s_5$
$x_{5z} = -s_{234} \cdot c_5$

$y_{5x} = -c_1 \cdot c_{234} \cdot s_5 + s_1 \cdot c_5$
$y_{5y} = -s_1 \cdot c_{234} \cdot s_5 - s_1 \cdot c_5$
$y_{5z} = s_{234} \cdot s_5$
$z_{5x} = -c_1 \cdot s_{234}$
$z_{5y} = -s_1 \cdot s_{234}$
$z_{5z} = c_{234}$
$p_{5x} = c_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2$
$p_{5y} = s_1 \cdot (-l_5 \cdot s_{234} + l_3 \cdot c_{23} + l_2 \cdot c_2$
$p_{5z} = l_1 - l_2 \cdot s_2 - l_3 \cdot c_{23} - l_5 \cdot c_{234}$

Based on the equations above, we can easily get the position of each joint:

$i = l_1 - l_5 \cdot c_{234} - p_{5z}$
$j = p_{5x} \cdot c_1 + p_{5y} \cdot s_1 + l_5 \cdot s_{234}$

$$q_1 = \arctan(\frac{p_{5y}}{p_{5x}})$$

$$q_2 = \arctan(\frac{i \cdot (l_2 + l_3 \cdot c_3) - j \cdot l_3 \cdot s_3}{i \cdot l_3 \cdot s_3 + j \cdot (l_2 + l_3 \cdot c_3)})$$

$$q_3 = \arccos(\frac{i^2 + j^2 - {l_2}^2 - {l_3}^2}{2 \cdot l_2 \cdot l_3})$$

$$q_4 = 0 - q_2 - q_3$$

$$q_5 = c_{234} \cdot q_1 - 2 \cdot \arctan(\frac{x_{5y}}{x_{5x}})$$

From above, we get the rotation angle of each joint, and we can calculate the angular velocity and angular acceleration at the moment by the angle, so the forward and inverse kinematics calculation has been basically completed.

However, in the process of forward and inverse kinematics, we have encountered many difficulties and tried many methods to calculate the kinematics.

In the use of the most convenient MATLAB robotic toolbox, its own $fkine()$ and $ikine()$ two functions, can be very convenient to solve the forward kinematics and inverse kinematics. However, in the experimental process, we found that it has limitations. In some special positions, the array will be out of bounds error, and this function is extremely difficult to check and repair, so we found other methods on the Internet. As we ended up using, we calculated directly from DH parameters, bypassing the toolbox functions and making the final trajectory smoother and more normal.

3) Trajectory Planning

In this part, we use MATLAB-robotics-toolbox to do the trajectory planning.

We used specific data for testing. After several experiments, we could get the curves of Angle, angular velocity and angular acceleration of each joint with time. It was found that the curves were smooth and had certain effects.

4) Control Procedure

The actuator we used is the AX-12A dynamixel connected with u2d2.

To control the actuator with MATLAB, we use matched dynamixeSDK package provided in the dynamixel official website. With the package we could simply control the actuator with the method $write2ByteTxRx$ which could write the goal position and moving speed in the actuator after initializing the baud rate and port number with your computer. To decrease the error and detect whether the actuator reach its goal position as written ,we use $read2ByteTxRx$ method to read the present pose to monitor. Since the two robot arm are in parallel, its necessary to cooperate the torque in both side or it wont rotate. To cooperate, we set complementary goal position in every two actuator. In practice ,since the camera catch the motion of people in a very high fps and the delay time due to read and write between VScode and MATLAB, we write the moving speed as a relatively slow number and pause after 4 seconds to avoid huge amount of tedious statistics. Also, the boundary is necessary in case the inverse kinematics calculated before gives a unreachable pose due to the restriction of actuator and mechanical configuration limitation.

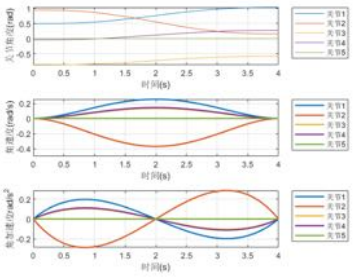


Fig. 17. Trajectory planning

Fig. 18. Control Process

5) Pressure Sensor

In this project, We successfully realized the remote control of the robot arm to move through visual recognition of human body posture. For real application, the one-way controlling is not enough, the manipulator needs to obtain the parameters perceived by the end-effector of the robot arm during operation, such as visual and tactile information. We try to add the tactile sensing module, so that we can get the tactile information of the end-effector when operating the object, so that the operator can get the tactile feedback, just like the real operation of an object in our daily life.

Due to the time constraints of this course project, we have placed the implementation of the tactile module last in our priority list. After completing the visual recognition part of attitude tracking with my teammembers, I began to study the force tactile sensing module of the manipulator end-effector.

The first step is to select the appropriate pressure sensing module and sensing mode. According to the specifications of the teaching kit we used and the working capacity of the robotic arm we assembled, we selected the pressure sensor measuring range of 20g – 6000g. This range also meets the pressure requirement of our daily grip on small objects. The sensor we use is the thin film pressure sensor on Taobao, whose response time is less than 10ms.

The second step is to detect the serial port information of the pressure sensor. In this project, the operation of the mechanical arm and the force tactile detection were carried out simultaneously, so I used Arduino as the independent detection of the pressure sensor in order not to interfere with the process.

For the convenience of reading data through serial port, we use the resistance-voltage conversion module of the thin-film pressure sensor, which can output analog signals, and adjust the magnification and comparison threshold to adjust the reading range. We use the AO pin (which is suitable for detecting changes in pressure trends and for measuring pressure values) to read the sensing to detect values, which range from 0.1V to 3.3V. In order to test the accuracy of the results, we use AO_res potentiometer to adjust the gain. In this step, we conducted a precision adjustment experiment by applying a known force and adjusting the potentiometer so that the serial port output weight is a given pressure. From the experimental measurement, we can get the relationship between the output voltage of the sensor module and the output resistance of the thin film pressure sensor:

Where U_0 is the output voltage of the sensor module, r_AO-res is the feedback resistance, and R_x is the output resistance of the thin film pressure sensor. I wrote a relatively simple program for every 10ms output voltage and pressure magnitude.

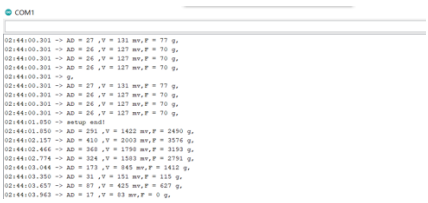


Fig. 19. Code of control

The third step is to apply the membrane pressure sensor to the end-effector grip force feedback. When the end-effector executes the command of grasping, we can obtain the force applied by the actuator to the object through the pressure sensor, so as to protect the object we grasp. For example, with brittle materials, too much force will cause the object to break.

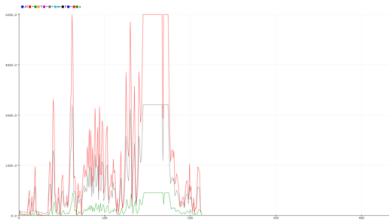


Fig. 20. Pressure curve

III. Experiment Results

1) Results

After numerous failures on everyone's part, we were able to integrate what we had and ultimately achieve much of what we had originally envisioned.

We calculated the visual control of the human arm in Python at the present, and exported the CSV file, which was the joint coordinate with equal interval. MATLAB read the real-time CSV file and then calculated the forward and inverse kinematics to get the rotation angle of each joint of the manipulator. In MATLAB by $while$ loop to control the motor, to each joint a target angle and angular velocity of motor, smooth mechanical arm's reach manpower needs constant motion position and the current position, and through the change of the hand movements to determine whether to the person in grasping object, and through the size of the pressure sensor in real time adjustment, so that the grasping object will not be damaged. The final experiment results are good, basically meet our needs.

Here are some of the renderings:

Fig. 21. Rendering1

Fig. 22. Rendering2

2) Difficult and unresolved issues

Since this project is a course project, the time is relatively tight, and the time for us to try to implement the scheme is very short, so we need to spend more time to test the optimal scheme in some places.

One of the major problem is about robot arm control. We found in the test that when the human is not doing attitude control, the robot arm will have a certain Angle of deflection. This deflection is fine in the early stage, but after a certain period of operation, this effect will gradually accumulate, so that we can not distinguish whether the movement of the manipulator is controlled by our posture or its own influence. We have explored this problem to a certain extent and concluded three reasons for its influence. The first is the transmitted data. Since the results of visual recognition are all the world coordinates relative to the camera, when the human body keeps the same movement, the depth prediction will be affected by external interference factors and will change to a certain extent. Due to the size difference between the human arm and the mechanical

arm, the conversion coefficient calculated by us is only based on our theoretical calculation. So a small change in the depth of recognition is likely to result in a large movement of the arm. We found these movements by testing flat motion, such as bending the elbows and wrists. When we test the stereo position, such as the rotation of the shoulder and the rotation of the forearm, the deflection of the manipulator itself will increase. The second possible reason is the influence of the control mode of the manipulator. There are two ways to move our manipulator, one is constant speed movement, using angle control, the other is through angular speed control. In the process of operation, under the first constant speed control, the movement range of the manipulator will be very large, and there will be a certain degree of tremor, but the response speed will be very high. However, after the speed decreases, the motion stability of the manipulator will be improved. The second angular velocity control is more accurate and stable, but the response speed is slower than that of angle control. More qualitative experiments are needed to analyze this most suitable control parameter.

The third possible reason, similar to the first, is about the transmission of data. When the pose data is transmitted to Matlab for motion calculation and motor control, all the identified human posture information is transmitted, and the response speed of the manipulator is constrained by the hardware, so there is no way to respond quickly to all samples. This leads to the influence of the next action, especially the noise, when one action is not finished steadily in the angle control. We modify the manipulator by reducing the sampling frequency, at this time, the action of the manipulator will be more stable, but we have not found the most suitable method to filter the data, so we will filter out some important attitude information.

The second problem is about data transmission. Our human posture recognition is carried out using python, while the control is using matlab. Because of the incompatibility of the software, the way of real-time data transmission is very rough. Real-time reading of python exported data by Matlab will cause a delay in the control of the manipulator, which will behave badly when running 4min. We only use python in she xiang, but we haven't found an optimized motion solution for pytho. This problem needs to be solved by a better solution.

3) Future Expectation

At present, the mature control mode of mapping human arm to robotic arm is to use wearable equipment. The posture of the human arm is sensed by sensors, and then mapped to the robotic arm, such as the current industry-leading Haptex. Such a control method requires the operator to wear heavy sensing and control modules on the hand, although it can bring the operator the most fine tactile perception, but its weight and complicated operation make the comfort poor, and it is difficult to popularize and apply. Using visual recognition to obtain the attitude information of the operator can greatly reduce the complexity and cost of the equipment, especially when we can directly identify the depth information through the ordinary camera, we can popularize this operation in the largest range, so that ordinary users without professional equipment can also use this solution for remote control or virtual control of VR. At present, the attitude recognition data acquisition has been relatively perfect, and it is the part of the kinematics scheme that needs to be optimized. We will test different algorithms and find that the mapping relationship is most suitable for human and six-axis manipulators. We plan to rely on unity and steam VR for further hand posture modeling to achieve accuracy and stability.

APPENDIX A
DIVISION OF LABOR AND CONTRIBUTION

Member	Student ID	Contribution
Jia-jun Long	12011624	20 %
Yi-fei Chen	12010502	20 %
Yi-jun Fang	12011301	20 %
Zi-meng Wang	12011711	20 %
Ke-run Xiang	12012005	20 %

ACKNOWLEDGMENT

The authors would like to thank for the course ME331 which gave us time and equipment to do this project of visual control robotic arm and force tactile feedback of gripper. Also, the authors would like to thank the teacher and TA of this course, who gave us a lot of help. Again, thanks a lot!

REFERENCES

[1] V.N.Iliukhin,K.B.Mitkovskii,D.A.Bizyanova,A.Akopyan(2017), The Modeling of Inverse Kinematics for 5 DOF Manipulator,*Procedia Engineering*,176,498-505

[2] $website : https : //google.github.io/mediapipe/solutions/holistic\#python-solution-api$