

Application of A Grasping Autonomous Transport Robotic Car with Machine Vision in the Express Industry in China

Yukun Xiong *

Department of Ulster College, Shaanxi University of Science & Technology, Shaanxi, China

* Corresponding author: 2006030414@st.btbu.edu.cn

Abstract. In recent years, COVID-19 has badly affected people's lives, and most industries have been hit hard, the express industry is a typical example, if the staff needs to be quarantined for epidemic control, the work efficiency of the express transfer center would reduce. In this project, to solve problems in the express industry, an autonomous transport car with a robotic arm was designed by using the technology of machine vision, path planning, SLAM, etc. The robot was simulated in the ROS platform m replaced by the turtlebot3 robot. As the result, the robot will go to the position set by Python code in the gazebo, when the robot identified the target, it will keep going straight forward to the target while keeping the target in the center of the camera. If the distance between the camera and the target is 0.15 meters, the car will stop and the robotic arm would grasp the target automatically. After the grasp, the robot will go to the destination and put the target in the basket. The result of the simulation shows that this robot can be applied to the express transfer center.

Keywords: Express industry, ROS, Machine vision, Path planning, SLAM.

1. Introduction

The express industry has dramatically developed in recent years, and it plays an important role in the national economy [1]. But under China's current epidemic prevention and control policy, the pandemic could cause some troubles. For instance, if the staff in the express transfer center contacted patients who are confirmed, then they need to be quarantined which will lead to the reduction of the work efficiency of the express transfer center. To solve this question, this project aims to design an autonomous transport robot car with a robotic arm that has technology about SLAM, path planning, machine vision, and so on. This robot car can take on some sorting and transporting work when the employee cannot come back to work in time.

An autonomous transport robot is a kind of robot that has abilities about transporting and moving, autonomous transport robots are capable of navigating and moving goods to their destination locations without the need for physical or electromechanical guidance in an uncontrolled environment. In China, the study of the field of autonomous transport car started late but have rapid development, from 2013 to 2016, the annual compound growth rate of the sales volume of autonomous transport vehicles in China reached 38.64 percent. In 2017, the domestic sales volume reached 13,500 [2]. With the development of automation, autonomous transport car is now widely used in different industries. In the express industry, now express centers in a neighborhood would use autonomous transport robots to send express. However, the use in the express transfer center is not widespread, such as sorting packages. To figure out the feasibility of the grasping autonomous transport robot that can be applied in an express transfer center, this project will do a simulation of the robot car to identify and grasp the package and then deliver the target to the destination on ROS.

2. Methodology

2.1. Design of the Car

2.1.1. Process of building the car model

In the project, to make the stability of the robot car better when moving, a four-wheel trolley was chosen to design. The process of grasping, is not very demanding, just making sure the robotic arm

can grasp the target, in that case, a 3-links robotic arm was chosen and the camera was equipped on the arm. At the connection between the manipulator and the trolley, it is a coupling structure of two servos to realize the free rotation of the manipulator, which is convenient for the overall mechanism to grasp smoothly after identifying the target.

In order to use ROS to simulate our project, the SW-to-URDF plugin was used to convert the finished Solidworks model to URDF format. The plugin can directly convert SW to a URDF file, but this conversion is likely to cause problems, so it is necessary to calibrate the car.

Firstly, it is important to understand who is base_link and who is joint_link in the process of car movement. In the trolley, the wheels can be moved, and the body is fixed, but the number of base_link options is limited, so in the assembly process, the unmoving things were assembled into a whole to preserve, and the next time the whole vehicle is assembled, it is imported as a whole part. The step is to start to establish the point, axis, and coordinate system. The wheels, trolley underplate, robot arm plane, connecting shaft, and other parts were calibrated respectively and finally imported above things into URDF files. The figure was shown in Fig. 1.



Figure 1. The model of the car

2.1.2. Data of the robotic arm links

Table 1. Data of the robotic arm links

Link	Length(m)
1	0.037
2	0.249
3	0.141
4	0.098

2.2. Algorithm or technique used in the project

2.2.1. Robotic arm grasp

Robot kinematics has two types: forward kinematics and inverse kinematics. In this project, the robotic arm equipped on the car is a RRR robot, in the process of grasping, the forward kinematics analysis is only used. The forward kinematics analysis is to determine the functional relationship between the position and attitude of the end effector and the base of the robot through calculation under the premise of knowing the variables of each joint [3]. The formula of the forward kinematics of the RRR robot is as follows:

$$p_e = \begin{bmatrix} a_1 \cos \theta_1 + a_2 \cos(\theta_1 + \theta_2) + a_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ a_1 \sin \theta_1 + a_2 \sin(\theta_1 + \theta_2) + a_3 \sin(\theta_1 + \theta_2 + \theta_3) \end{bmatrix} \quad (1)$$

The code of the grasping part is written in python. The grasping part of the robotic arm includes two main bodies. One is the service part to communicate and cooperate with the other functional part of the robotic arm. And the other is to use MoveIt in ROS to implement the grasp action of the robotic arm, which is the main reason for the grasp of the arm.

In this project, the car would stop if the distance between the target and the camera is 0.15 meters, because the position of the target is fixed in this project, the coordinate of the cylinder is known, so there is no need to use a machine vision when the robotic arm grasping the cylinder, the car just needs to stop in front of the target and the robotic arm would grasp automatically.

2.2.2. Building map based on SLAM

The circumstance of this project is Turtlebot3 world. In order to let the robot, move automatically, the first step is to draw a map of the environment. SLAM is the main technology to build the map, SLAM is to build an environmental map according to the robot's position and then determine the robot's position from the known map. The two complement each other. It can be described as using sensor information to simultaneously solve the problem of establishing an environmental map and determining its orientation [4]. It has multiple components: landmark extraction; data association; state estimation; state update and landmark update. Measurement devices such as Lidar, Sonar, and Vision are equipped.

Kalman Filter is the heart of the SLAM process. Kalman filtering is an optimal state estimation algorithm that can realize linear minimum variance estimation for linear systems [5, 6]. It updates where the robot thinks it is based on the landmarks, the KF corrects the odometry pose estimate in real-time considering the landmark measurement uncertainty. Kalman Filter Block Diagram is shown in Fig.2.

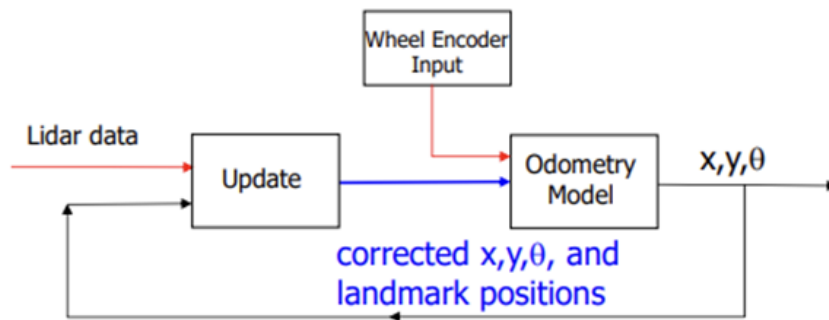


Figure 2. Kalman Filter Block Diagram

At the beginning of the drawing, the current position of the robot is set to a known starting point, and a 3×1 vector x_k and a 3×3 matrix C_k are built. x_k represents known points and C_k represents the relationship between known points.

$$x_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \quad (2)$$

$$C_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta^2 \end{bmatrix} \quad (3)$$

As the robot moves and environmental information is entered, the landmark information $m1$ and $m2$, etc, will be updated. Added to x_k , C_k .

Gmapping is the SLAM method that is being used in this project, gmapping uses the RBPF method, which has the advantage of accurate positioning [7]. After launching the simulation environment, the

gmapping was launched, and then use the keyboard to control the robot to draw the map. After finishing the drawing, the map was reserved. The results are shown in Fig. 3.

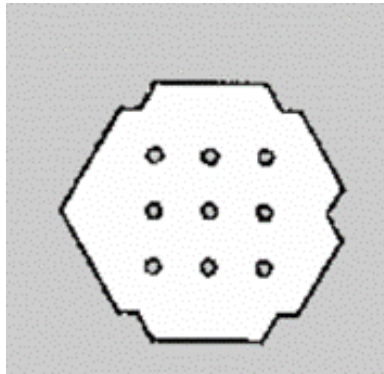


Figure 3. Map of the environment

2.2.3. Path planning

In the path planning part, Dijkstra's algorithm was mainly used. Dijkstra's algorithm is a method to find the shortest path [8].

The process of the dijkstra's algorithm:

- 1) Set the distance of the origin node as zero, and the distance between the origin node and other nodes as infinity.
- 2) Mark all nodes as unvisited and set the node of origin.
- 3) For the current node consider all of its unvisited neighbors and calculate its distance from the node of origin using the path through the current node. If the calculated distance of the unvisited neighbor through the path containing the current node is less than its distance value overwrite the distance value using the new value.
- 4) When all the neighbors of the current node have been considered flag the current node as visited. A visited node will never be checked again its value is final and minimal.
- 5) Set the unvisited node with the lowest distance value as the current node and continue from step 3.

In this project, four positions had been set in the simulation environment, landmark '1', '2', '3', and '4' had been set on the ground and their coordinates of the center point is known, position 1 is where the target cylinder is located, position 2, 3, 4 are the destinations where the baskets located. So in the project, the coordinates were written in the python code, after running the python code, the car will automatically go to the position and finish the whole grasp and transport work. During the way, the car will avoid obstacles based on Dijkstra's algorithm. The whole process can be suitably used in the express transfer center.

2.2.4. Machine vision

In the project, a depth camera realsense was equipped on the robotic arm to identify the object, the machine vision is based on OpenCV. OpenCV is a computer vision and machine learning software library which is written in C++, OpenCV has many general algorithms for image processing and computer vision is implemented [9]. In the project, we boxed different colors in our code by using the RGB method, by changing the three color channels of red (R), green (G), and blue (B) and superimposing them on each other, different color different RGB numbers, in this way, the Turtlebot3 can identify different colors. Results of identification is shown in Fig. 4.

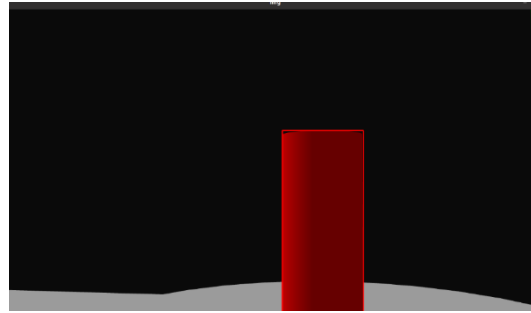


Figure 4. Results of identification

After the car get the target object, in order to keep the target in the middle of the camera, the PID controller was used. PID is the sum of the three links of proportion, integration, and differentiation and uses feedback to detect deviation signals and control the controlled quantity through deviation signals [10]. Proportional control increases open-loop gains of systems. This part of control is dominant, differential control is a leading mode of predictive regulation, forecasts system variations, increases system damping, and enhances phase margin, integral control manifests the effects of histories upon current systems [11]. The working principle of PID controller is shown in Fig. 5.

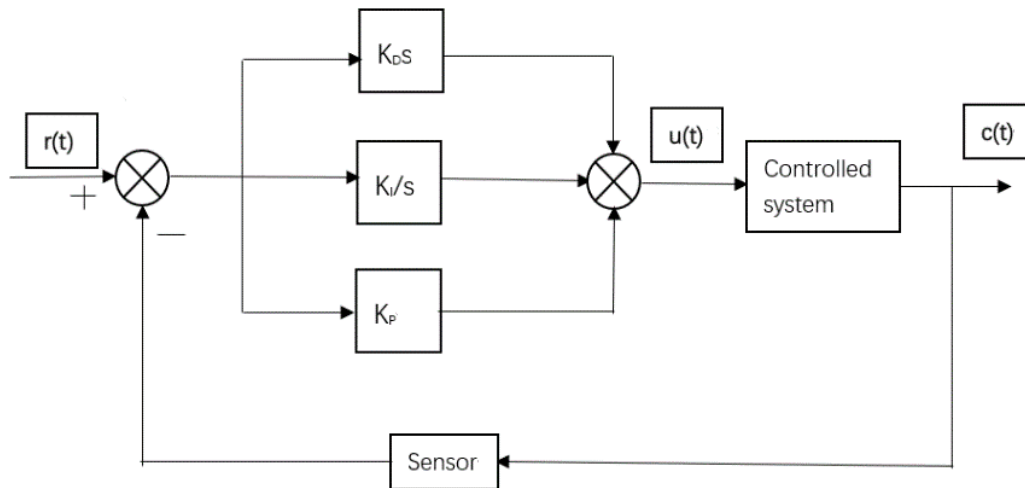


Figure 5. The working principle of PID controller

The basic control law of PID:

$$U(t) = K_p(err(t) + \frac{1}{T_I} \int err(t)dt + \frac{T_D de(t)}{dt}) \quad (4)$$

When the camera gets the target, the contour of the object will be covered with a rectangle, and the centroid of the rectangle will be tracked by the camera and make sure it is in the center of the vision. Then the car will approach the target slowly.

2.3. Demo

After all the Python and C++ codes had been written, a demo was established as shown in Fig. 6.

```
1 #!/bin/bash
2
3 source /opt/ros/noetic/setup.bash
4 source ~/catkin_ws/devel/setup.bash
5
6 rosservice call /move_place "place: 1"
7 rosservice call /track_obj "obj: 'red'"
8 rosservice call /grasp_object "type: 1"
9 rosservice call /move_place "place: 2"
0 rosservice call /grasp_object "type: 2"
```

Figure 6. Demo code

In the demo, the place number can be changed if the user tries to send the target to different destinations, and the color of the object can be also changed if the color is in the code of the machine vision. In this way, the robot car can grasp different targets with different colors.

Service is a call of a synchronous cross-process function, when the model of the client/server was used, the client nodes can call the functions that run-in server-side nodes [12]. The client calls this service through a local proxy request. Synchronization: The client sends the request data, and the server returns the reply data after the processing is complete. After the client sends the request, it blocks until the server returns the results. This mode is more suitable for bidirectional synchronous information transfer, as shown in Fig. 7.

The process to define a service:

- 1) Establish an SRV file.
- 2) Write a service node (Python).
- 3) Write a client node.

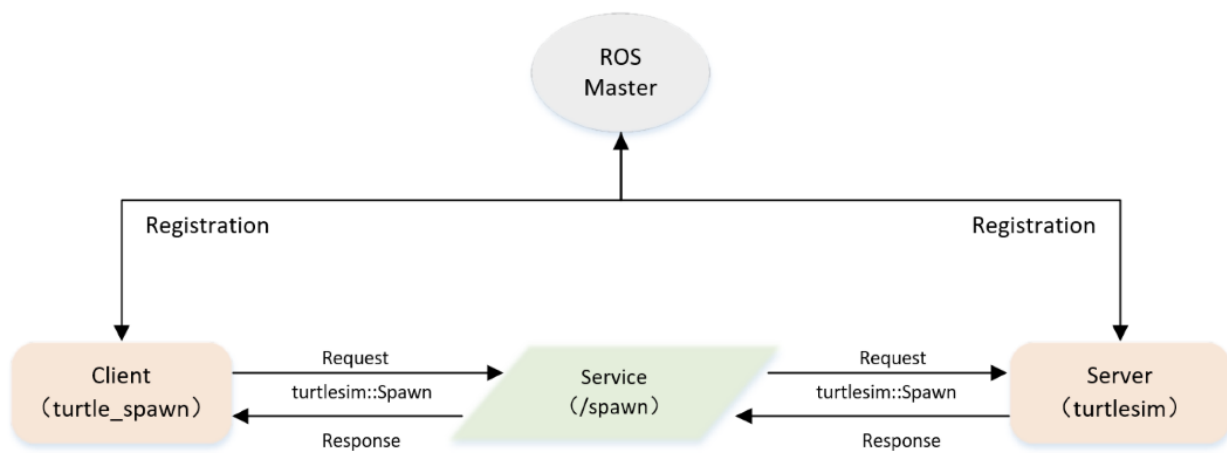


Figure 7. The model of service

3. Comparison and Analysis

3.1. Experiment results

When the program is running, the car would go to the center of rectangle 1, then it would turn left and search for the target. When the car finds the target, it will focus on the target and make sure the target is in the middle of the vision. When the distance between the camera and the target is 0.15m, the car would stop and the robotic arm would grasp the target cylinder automatically. In this project, the destination is the blue basket in position 2. After the robotic arm grasps the cylinder, the car will go to the basket and drop it. The process of this project is just like that. Simulation environment is shown in Fig. 8.

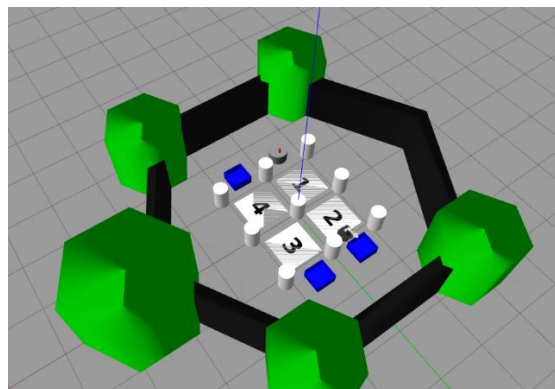


Figure 8. Simulation environment

3.2. Discussion

In this project, there are some limitations. Firstly, when the URDF file of the car was tried to import into the simulation environment, it failed, so we used open source Turtlebot3 robot to replace it, but it may cause some problems. For example, when the car is used in the simulation, the car may not grasp the target because the height is not suitable for the car. Secondly, the obstacles in this simulation are fixed, but in the express transfer center, there may have staff on the way where the robot is going which will cause the collision. Thirdly, the shape of the target is a cylinder, but the shape of the package is a cube commonly, so the robotic arm may not grasp the package correctly. In this project, many technologies have been used. The main technologies are SLAM, machine vision, path planning, and robotic arm grasp. The simulation results show that the car can do the transport work and can do it automatically. But there still have some problems in the simulation that need to be improved. If the problems about bypassing the staff, grasping the cube target, and so on can be solved, the project can be better. According to the results from the simulation, this robotic car can be used in the express transfer center. The transfer center usually does not have many obstacles, so the route that has been set in the project can be used successfully in the transfer center environment. The robotic arm has a strong ability to snatch, so it can deal with different shapes of the package. Above all, these are the reason why the robotic car can be used in the express industry.

4. Conclusion

In order to improve the project, this project needs to do more experiments. Instead of using the environment of the Turtlebot3 world, an express transfer center environment can be built in the gazebo and it can add some moving obstacles to represent the staff in the transfer center, which will make the simulation more reliable. In order to avoid the moving obstacles when the robot is going to the destination, some functions can be added to the machine vision code to achieve the goal. For other problems, the gripper size can be increased to grasp the different shapes of the package. The more target cylinder can be added in the simulation environment to show that the robot can identify the different colors of the target. After solving these problems, the results of the simulation can be more reliable. The robot car can be used in the express industry.

References

- [1] Luo C, Wu L, Liu N. Study based on contactless distribution patterns under the outbreak [C]//IOP conference series: earth and environmental science. IOP Publishing, 2020, 526 (1): 012204.
- [2] Rioux A, Suleiman W. Autonomous SLAM based humanoid navigation in a cluttered environment while transporting a heavy load [J]. Robotics and Autonomous Systems, 2018, 99: 50 - 62.
- [3] Zuo J, Zhu X, Peng Y, et al. A new method of posture recognition based on WiFi signal [J]. IEEE Communications Letters, 2021, 25 (8): 2564 - 2568.
- [4] Dai Yue. Research on Robot Positioning and Navigation Algorithm Based on SLAM [J]. Wireless Communications and Mobile Computing, 2022, 2022.
- [5] Lai X, Zhu G, Chambers J. A fuzzy adaptive extended Kalman filter exploiting the Student's t distribution for mobile robot tracking [J]. Measurement Science and Technology, 2021, 32 (10): 105017.
- [6] Xu X, Pang F, Ran Y, et al. An indoor mobile robot positioning algorithm based on adaptive federated Kalman Filter [J]. IEEE Sensors Journal, 2021, 21 (20): 23098 - 23107.
- [7] Norzam W A S, Hawari H F, Kamarudin K. Analysis of mobile robot indoor mapping using GMapping based SLAM with different parameter [C]//IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2019, 705 (1): 012037.
- [8] Rachmawati D, Gustin L. Analysis of Dijkstra's algorithm and A* Algorithm in shortest path problem [C]//Journal of Physics: Conference Series. IOP Publishing, 2020, 1566 (1): 012061.
- [9] Bradski G. The openCV library [J]. Dr. Dobbs Journal: Software Tools for the Professional Programmer, 2000, 25 (11): 120 - 123.

- [10] Knospe C. PID control [J]. IEEE Control Systems Magazine, 2006, 26 (1): 30 - 31.
- [11] Zhou Yi. A Summary of PID Control Algorithms Based on AI-Enabled Embedded Systems [J]. Security and Communication Networks, 2022, 2022.
- [12] Zimmermann P, Axmann E, Brandenbourger B, et al. Skill-based engineering, and control on field-device-level with opc ua [C] //2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, 2019: 1101 - 1108.