Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 4, No. 3, 2022

Path Planning and Parameter Adjustment

Jiaqi Cui*

School of Electrical and Control Engineering of North China University of Technology, Beijing, China
* **Corresponding author**: C2324360198@163.com

Abstract: By building ros and gazebo simulation environment in virtual machine, path planning, navigation, positioning and SLAM mapping were carried out for Ackerman chassis car, and traffic sign recognition and lane line detection were completed by combining machine vision, so as to control the intelligent car's turning. Test the navigation parameters with TEB plug-in to obtain the best parameters. After testing, the navigation effect is ideal, with reliability and accuracy.

Keywords: TEB, Yolov5, Opencv, DWA, SLAM, ROS, Gazebo.

1. ROS

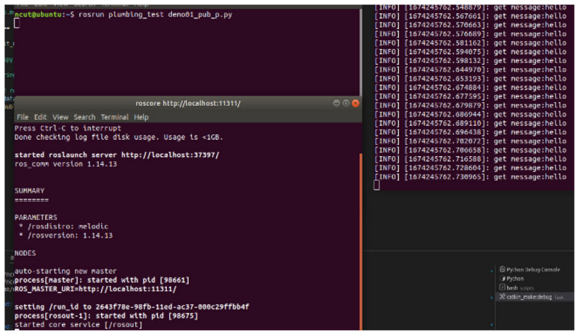

Figure 1. Subscribe and publish

(1) Subscribe message:
• Initialize the ROS system
• Subscribe to chatter
• Enter the self-loop and wait for the message to arrive
• The chatterCallback() function is called when the message arrives
(2) Release information:
• Initialize ROS nodes: Naming (unique)
• Instantiate the publisher object
• Organize the published data and write the logic to publish the data.

2. Path Planning

```
<launch>

        <!--  ************** Global Parameters ***************  -->
        <param name="/use_sim_time" value="true"/>

        <!--  ************** Navigation ***************  -->
        <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
                <rosparam file="$(find my_navigation)/params/costmap_common_params.yaml" command="load" ns="global_costmap" />
                <rosparam file="$(find my_navigation)/params/costmap_common_params.yaml" command="load" ns="local_costmap" />
                <rosparam file="$(find my_navigation)/params/local_costmap_params.yaml" command="load" />
                <rosparam file="$(find my_navigation)/params/global_costmap_params.yaml" command="load" />
                <rosparam file="$(find my_navigation)/params/teb_local_planner_params.yaml" command="load" />

                <param name="base_global_planner" value="global_planner/GlobalPlanner" />
                <param name="planner_frequency" value="1.0" />
                <param name="planner_patience" value="5.0" />
                <param name="GlobalPlanner/neutral_cost" value="40"/>
        <param name="GlobalPlanner/cost_factor" value="0.55"/>

                <param name="base_local_planner" value="teb_local_planner/TebLocalPlannerROS" />
                <param name="controller_frequency" value="5.0" />
                <param name="controller_patience" value="15.0" />
        </node>

          <!--  ************** Visualisation ***************  -->
          <node name="rviz" pkg="rviz" type="rviz" args="-d $(find ucar_sim)/rviz/ucar_sim.rviz"/>

</launch>
```

Figure 2. Launch file

Path planning includes global path planning and local path planning. Global path planning The global path is planned according to the given target final location and starting point. Use the A* algorithm. The local planner: planned escape routes based on nearby obstacles. In addition to DWA algorithm, there is a more suitable algorithm for robots on the chassis of Ackerman model, Teb algorithm.

3. TEB And Ackermann Model

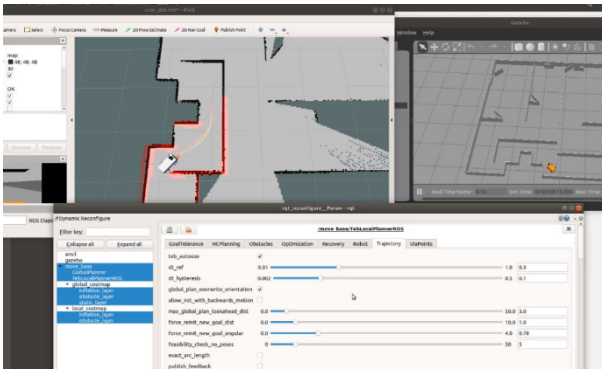

Figure 3. Route planning effect display

Rubber band theory: The state of the starting point and target point is obtained from the global planner. In the middle, N control points are inserted to control the shape of the rubber band. In order to display the kinematic information of the trajectory, the motion Time time is defined between points.

Among them, the location of Robot footprint model, the Current goal of the target point of the car and the black path are obtained from the global path. n control points are inserted in the middle. The contents of these points are the pose of the robot, including x coordinate and y coordinate and the heading information of the car. Each point is directly equal in time. From the above information, the velocity of the car in a short distance can be calculated. After differentiating, the acceleration can be obtained, as well as the Angle and angular velocity.

Deformation of rubber bands refers to obstacles. When obstacles are moved, the dist1-dist3 changes. In order to maintain the original state, the path changes, just like when rubber bands are deformed by external forces. There are also forces between adjacent points.

4. Parameter Adjustment

① max_global_plan_lookahead_dist: Consider the maximum length of the optimized subset of the global plan
② min_obstacle_dist: indicates the minimum distance between an obstacle_dist and an obstacle
③dt_ref: the resolution of local path planning, that is, the

time resolution, the time between two adjacent poses

④costmap_obstacles_behind_robot_dist; When planning, consider obstacles within n meters behind

⑤inflation_dist: buffer around obstacles

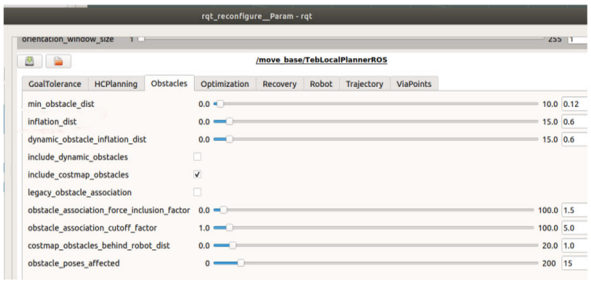

Figure 4. Parameter adjustment

roslaunch ucar_sim ucar_robot.launch
roslaunch my_navigation gmapping.launch
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
roslaunch my_navigation saveMap.launch
roslaunch my_navigation loadMap.launch
roslaunch my_navigation amcl.launch
roslaunch my_navigation omnidir_movebase.launch
rosrun rqt_reconfigure rqt_reconfigure

5. Lane Detection

```
def do_color(img):
    img = np.where(img < 100, 0, 255).astype(np.uint8)
    _, conts, _ = cv.findContours(img, mode=cv.RETR_EXTERNAL, method=cv.CHAIN_APPROX_SIMPLE)
    img = cv.cvtColor(img, cv.COLOR_GRAY2BGR)
    img = img.copy()
    cv.drawContours(img, conts, -1, (0, 255, 0), 1)
    return img
```

Figure 5. Contour recognition of lane lines

```
def do_center(image):
    gray = image
    blurred = cv.GaussianBlur(gray, (5, 5), 0)
    thresh = cv.threshold(blurred, 60, 255, cv.THRESH_BINARY)[1]
    cnts = cv.findContours(thresh.copy(), cv.RETR_EXTERNAL,
                           cv.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    for c in cnts:
        M = cv.moments(c)
        if M["m00"] == 0:
            continue
        cX = int(M["m10"] / M["m00"])
        cY = int(M["m01"] / M["m00"])
        cv.drawContours(image, [c], -1, (0, 255, 0), 2)
        cv.circle(image, (cX, cY), 7, (255, 255, 255), -1)
        cv.putText(image, "center", (cX - 20, cY - 20), cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
        posex = cX
        posey = cY
        data_get = posex
        print("        (x=", posex, ",y=", posey, ")")
    return image
```

Figure 6. Center coordinate extraction

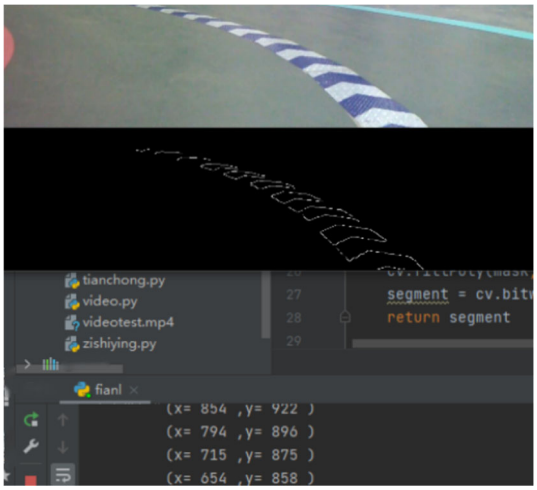

Figure 7. Lane detection effect

The lane lines were separated by binarization, retest expansion, area of interest extraction, contour extraction and other algorithms. Through the video shot by the camera, the coordinates of the center point are accurately output, and the error rate is 0.17%.

6. Traffic Signs Detect and Control Turning

```
def talker(ok):      ######################
    pub = rospy.Publisher('che', String, queue_size=10)#
    rospy.init_node('talker', anonymous=True)###########
    msg = String()###################
    while not rospy.is_shutdown():###################
        msg.data="hello"###################  ok
        pub.publish(msg)###################
    return ok

if __name__ == '__main__':
    a = opencv(0)
    try:###################
        talker(a)###################
    except rospy.ROSInterruptException:#################
        pass###################
```

Figure 8. ros publishing function

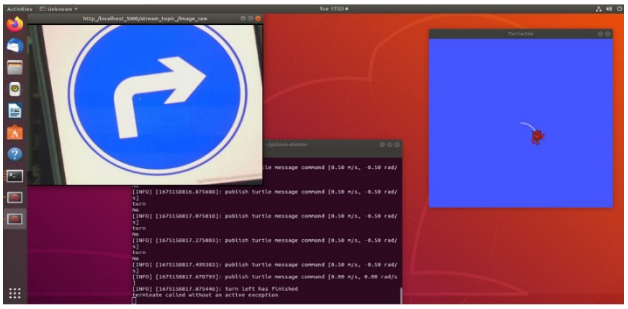

Figure 9. Turn Right

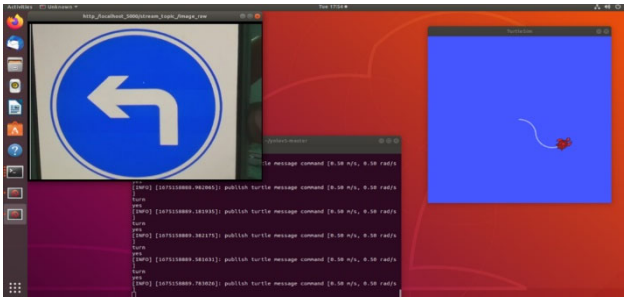

Figure 10. Turn left

Open the camera in the virtual machine, publish the information identified by yolov5 in combination with the subscription and publication function of ros, and control the turning of the intelligent car. When a detected traffic sign turns left, it makes a successful left turn, with an accuracy of 90 percent.

Table 1. Identification accuracy

	1	2	3	4	5	6	7
left	**91%**	**97%**	**95%**	**93%**	**92%**	**97%**	**90%**
right	**91%**	**92%**	**97%**	**90%**	**97%**	**90%**	**88%**
back	**91%**	**97%**	**99%**	**94%**	**95%**	**92%**	**97%**
stop	**89%**	**88%**	**96%**	**92%**	**88%**	**88%**	**96%**
wait	**92%**	**97%**	**93%**	**94%**	**94%**	**95%**	**88%**

roscore
rosrun uvc_camera uvc_camera_node
roslaunch web_video_server web_video_server.launch

source activate py24
cd yolov5-master
python3 detect1.py
rosrun turtlesim turtlesim_node

7. Ackerman Chassis Control

Encoder speed measurement: There are two ways to perform encoder speed measurement.

```
void BianMaqi_LEFT(void)
{
  actual_speed_L = (((float)TIM1->CNT - 32765) /  ENCODER_TOTAL_RESOLUTION/ REDUCTION_RATIO) * 100.0f*60.0f;
  TIM1->CNT =32765;
}

void BianMaqi_RIGHT(void)
{
  actual_speed_R = (((float)TIM4->CNT - 32765) /  ENCODER_TOTAL_RESOLUTION/ REDUCTION_RATIO) * 100.0f*60.0f;
  TIM4->CNT = 32765;
}
```

Figure 11. Encoder configuration

(1) The first method is the encoder mode of timer.

Calculation formula: Current speed = calculated value per unit time/encoder resolution * time coefficient

(2) The second is the external interrupt mode: Set external interrupts on the IO ports corresponding to the four A-phases of the encoder respectively, and carry out rising edge detection. After entering the interrupt, detect the high and low level of phase B, and judge the positive and reverse according to the level difference of phase AB.

(3) The first reason is that the external interrupt mode is unstable, so the simulation debugging is carried out on the chip of stm32f103, and the adjustment coefficient process is complicated in the PID control process. The second applies to chip does not support timer setting encoder mode.

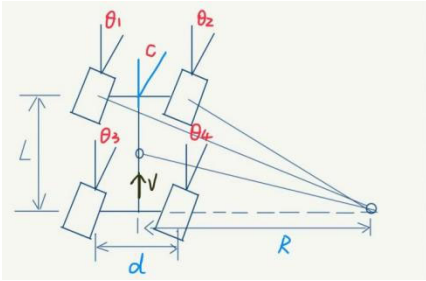

Figure 12. Ackerman model

c: Center corner

R: Turning radius

L: The conductor

D: Distance between front and rear wheels

V: The speed of the car

(1) The front left wheel turning Angle θ1 and front right wheel turning Angle θ2 can be calculated when c and car constant are known.

$$\theta 1 = \arctan \frac{L}{R+\frac{d}{2}}$$

$$\theta 2 = \arctan \frac{L}{R-\frac{d}{2}}$$

$$c = \arctan \frac{L}{R}$$

(2) Given the vehicle motion speed V and its own constant, the left motor speed Vz and the rear right motor speed Vy are calculated

$$Vz = V \frac{R+\frac{d}{2}}{R}$$

$$Vy = V \frac{R-\frac{d}{2}}{R}$$

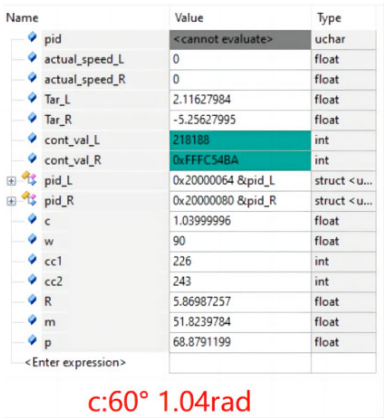

Figure 13. Simulation test result

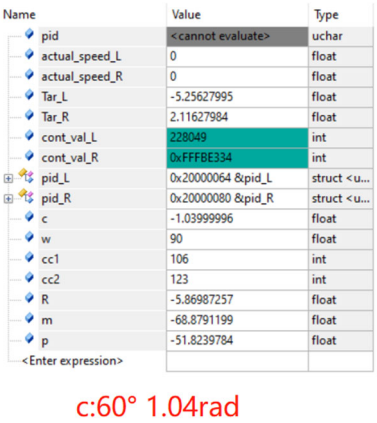

Figure 14. Simulation test result

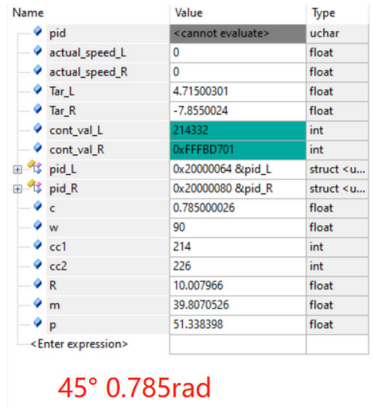

Figure 15. Simulation test result

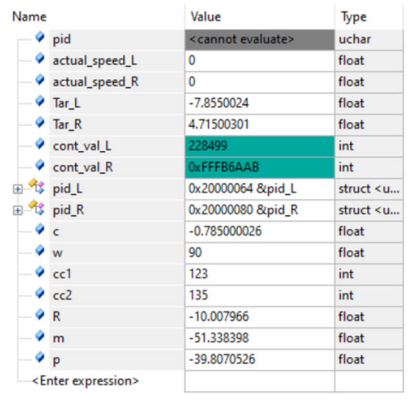

Figure 16. Simulation test result

8. Conclusion

By building ros and gazebo simulation environment and adjusting parameters of TEB algorithm, the navigation and positioning of map construction were completed. Combining opencv and deep learning machine vision, lane detection and traffic sign detection are completed. It has high accuracy and reliability in the process of intelligent vehicle running.

References

[1] J.T. Platt ,K.N. Ricks:Comparative Analysis of ROS-Unity3D and ROS-Gazebo for Mobile Ground Robot Simulation[J]. Journal of Intelligent & Robotic Systems,Vol. 106(2022) No.4.

[2] L. Guo,G.D. Qi,Y.B. Zhao,L. Huang:Research on Robot multi-task navigation scheduling based on A~* and TEB algorithm[J]. Journal of Huazhong University of Science and Technology,Vol. 51 (2023)No.2, p.82-88.

[3] Q.X. Gao:Research on trajectory Planning of mobile robot based on TEB algorithm[J]. Agricultural Equipment and vehicle Engineering, Vol. 60 (2022) No.7, p.126-129.

[4] C.S. Wei,Y. Cong, Y.J. Xiao,G.Q Zhang,S.Y. Zu, W.Q. Fu:Measurement method and test of slip rate based on Ackermann steering principle[J]. Chinese Journal of Agricultural Mechanization, Vol. 43 (2022)No.12, p.118-124.

[5] P.C. Shi,X. Chen,A.X. Yang,L. Zhang:4WID-4WIS intelligent vehicle Ackerman steering trajectory planning and position estimation[J]. Journal of Engineering Design, Vol. 29 (2022)No.2, p.123-132.

[6] J.Z.Cheng, X.C.Wu,J.L Wu,P.K. Wu:YRNet: A multi-target vehicle detection method based on hybrid neural network [J]. Laser Journal, Vol. 44 (2023)No.1, p.48-55.