Study on the development of an open motion control platform for a differential mobile robot based on ROS

Jiwu Wang[1], Fangbo Liao[1],Sugisaka Masanori [2]

[1]Department of Mechanical and Electrical Engineering, Beijing Jiaotong University, Beijing, China, 100044
[2]*Alife Robotics Corporation Ltd., 1068-1 Oaza Oshino, Oita*, Japan

Abstract: Robust motion control is prerequisite for an advanced robot. Based on the ROS open source platform, wedeveloped a reliable platform for any differential mobile robot. This platform can be modified according to practical various requirements conveniently. That is, it is an open control platform for any sensors. The developed system has been verified with a mobile robot developed in our lab. The experiment results show its reliability and robustness.

Keywords: mobile robot, motion control

1 INTRODUCTION

Traditional robotic software structure is often determined by the specific hardware. As the variety of the hardware, this led to the difficulty of the common and reusable code platform.ROS (Robot Operation System) is a schedule platform published by Stanford and maintained by Willow Garage with BSD license. Robot developers can build their own app or deploy the shared libs, middleware, and tools provided by ROS. Here a new mobile robot is developed with a completed differential mobile robot based on ROS requirement.

Figure1 shown in the system using differential drive, as a platform of nonholonomic vehicles, two drive wheelsare paralleland a balance wheel realize differential movement and the body balance. Vehicle hardware components include: the two motors, the speed detection unit, the motor drive unit, the motion control unit, the power supply unit, and so on.

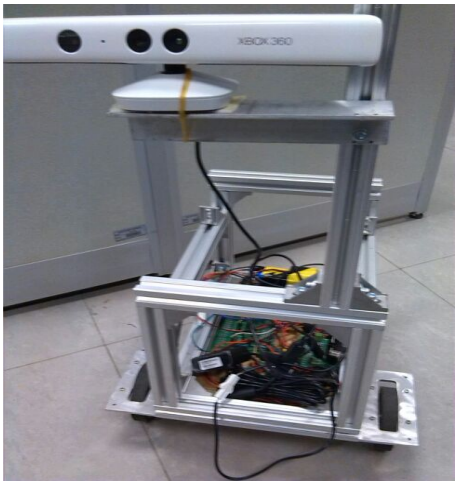

Figure1 Differential drive wheeled mobile robot

Figure2 depicts the relation between translation and rotation. The control of the robot at time t is expressed as linear and angular velocities.

According to planar motion characteristics of the robot, the robot control at time t is expressed as the linear and angular velocities $\begin{Bmatrix} v_t \\ \omega_t \end{Bmatrix}$

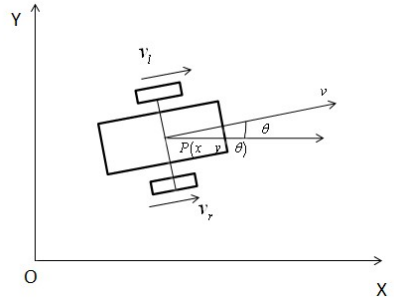

Figure2 Geometric relationships of the vehicle motion

Robot pose P has three degrees of freedom : $\begin{pmatrix} x & y & \theta \end{pmatrix}^T$, The mathematical relationship betweenc ontrol input u_t and pose P is as follows[1] :

$$\begin{bmatrix} x'(t) \\ y'(t) \\ \theta'(t) \end{bmatrix} = \begin{bmatrix} \cos\theta(t) & 0 \\ \sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix}$$

2 Motion Analysis

Since the trajectory of the differential mobile robot can be decomposed into three categories : linear, rotary, arc. Therefore, this system facilitates the kinematics conclusion to achieve the control on the three forms of motion[2].

Linear motion: when the speed of the two differential wheelsequals anddirections are the same, the trajectory of the robot is a straight line： $x(t)=\int_0^t v_r(t)dt$

Rotary motion: when the speed of the two differential wheelsequals and in the opposite direction, the robot moves around the original position： $\theta(t)=\frac{2}{L}\int_0^t v_r(t)dt$

Circular motion: when the directions of the two wheels are identical, speed remains unchanged, and the speed difference is constant, the trajectory of the robot is an arc.

2.1 Prediction model based on control

Suppose the input is $\begin{Bmatrix} v_t \\ \omega_t \end{Bmatrix}$, the state is $x_t=\left(x',y',\theta'\right)^T$ after Δt, also suppose the input is invariant in time Δt， so

$v=\omega * r$

$$x_c=x-\frac{v}{\omega}\sin\theta$$

$$y_c=y+\frac{v}{\omega}\cos\theta$$

The pose is as following after Δt

$$x_t=\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix}=\begin{pmatrix} x_c+\frac{v}{\omega}\sin(\theta+\omega\Delta t) \\ y_c-\frac{v}{\omega}\cos(\theta+\omega\Delta t) \\ \theta+\omega\Delta t \end{pmatrix}$$

2.2 Robot measurement model based on encoders

For the differential drive robot，robot state is x_t， control input is u_t， measurement is z_t， so the conditional probability of the robot state can be expressed as[3,4]：

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})$$

For convenience，we simplify the robot motion process to three stages，shown in figure 3：rotation-translation-rotation，represent the measured motion as：$\left(\delta_{rot1},\delta_{trans},\delta_{rot2}\right)^T$

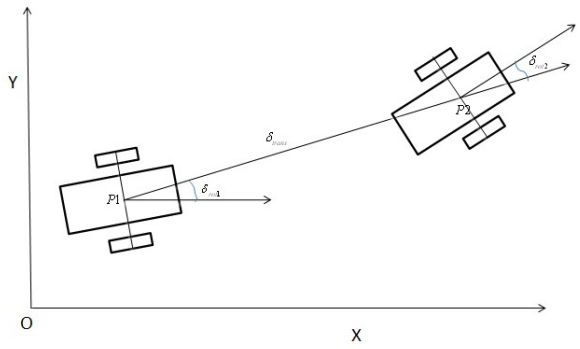

Figure3 vehicle motion analysis

Below is the algorithm to compute x_t :

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

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

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

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

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

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

$\hat{\delta}_{rot1},\hat{\delta}_{trans},\hat{\delta}_{rot2}$ stand for the true motion，wherein $sample(b)$ represents triangular or Gauss distribution with variance b zero mean， $\alpha_1,\alpha_2,\alpha_3,\alpha_4$ are the robot-specific error parameter，which characterize the robot cumulative error.

So the state $\bar{x}_t$ is computed as:

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

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

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

Wherein $x_t=\left(x',y',\theta'\right)^T$

2.3 Interface implementation

ROS controls the robot motion through the two messages：cmd_vel message implements the motion control from the high layer to the drive level；Odom message is collection of the motion state.

cmd_vel is of twist data type，twist data type is as the following：

geometry_msgs/Vector3 linear：float64 x，float64 y，float64 z

geometry_msgs/Vector3 angular：float64 x，float 64 y，float64 z

As shown above，Twist message contains two Vector3 type sub messages，in right-handed coordinate system, respectively represent the three directions of the line speed

and the angular velocity. For differential mobile robot , only the translation x and angular z are nonzero because of the freedom degree. The following analyses how the sent message is resolved and implemented:

As mentioned geometric motion analysis, when z is zero, x is the robot translation speed, low-level controller sends two identical channel PWM drive signal, via the motor drive amplifier, the robot pose is :

$$\left(x_i + d_i \cos\theta \quad y_i + d_i \sin\theta \quad \theta\right) \qquad \text{其中}$$

$$d_i = \int_0^t v(t)dt$$

When x is zero, z is the angular speed of the robot, the low-level controller send two channel PWM drive signal with opposite direction and same duty cycle.

The robot pose is $\left[x_i \quad y_i \quad \theta_i - \frac{2d_{Li}}{L}\right]^T$

When x, z are not simultaneously zero, we need to compute the value of v_x and v_y from the known x and known z:

Reverse solution vector U :

$$\begin{bmatrix} v_l(t) \\ v_r(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2}L & \frac{1}{2}L \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Send the velocity command of the two motors, the desired translation velocity angular velocity can be achieved.

Odom message represents the robot state, ROS publishes Odom message to indicate the estimate of the position and velocity, assuming we get $tick(t)$ from the left encoder within measured period $(t, t+\Delta t)$, the applied encoder parameter is η_1 , wheel diameter parameters is η_2 , further computation on these parameter and measured data, we can obtain one side translation and angular velocity,, similarity the right side[5]:

$$\omega_l(t) = \frac{tick(t)}{\Delta t} * \eta_1$$

$$v_l(t) = \frac{tick(t)}{\Delta t} * \eta_1 * \eta_2$$

And Odom message in ROS belongs to nav_msgs/Odometry type , the detail of the nav_msgs/Odometry is as following:

Header header
stringchild_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist

twist is as the previously described, we gain v_l and v_r from the two encoders, and translation and angular velocity can be deduced from the vector U .

The detail of the pose type:

geometry_msgs/Point position: float64 x , float64 y, float64 z
geometry_msgs/Quaternion orientation : float64 x , float64 y, float64 z, float64 w
float64[36] covariance

Therein,the values of x and y in positionare computed as follows:

$$\Delta x = [v_x(t)\cos\theta - v_y(t)\sin\theta] * dt$$

$$\Delta y = [v_x(t)\sin\theta + v_y(t)\cos\theta] * dt$$

$$\Delta\theta = \omega * dt$$

$$x_{i+1} = x_i + \Delta x$$

$$y_{i+1} = y_i + \Delta y$$

$$\theta_{i+1} = \theta_i + \Delta\theta$$

And orientation can be gotten:

odom.pose.pose.orientation
=tf::createQuaternionMsgFromYaw(θ);

3 Experiment design

In order to verify the accuracy of the robot motion, we design two forms of motion to check the constancy of the implemented and the desired results. The two algorithms are respectively translation motion drivex(distance, speed), rotation motion turnx(angle, angularSpeed).

divex(distance, speed)algorithm: send velocity command to robot, record the motion start point:startPose, and update the current pose in real time: CurrentPose, so the distance is:

dx = currentPose.x–startPose.x
dy =currentPose.y–startPose.y
distanceMoved = math.sqrt(dx * dx + dy * dy)

compare the value of distanceMoved and desired distance, controlling the robot to a desired distance can be realized.

Turnx(angle , angularSpeed)algorithm : send angular velocity command cmd_velto robot, record the motion start point:startPose, and update the current pose in real time: CurrentPose, so the rotation angular is:

dx = currentPose.x–startPose.x
dy = currentPose.y–startPose.y
angleTurned = currentAngle + angleOffset–startAngle

whereinangleOffset is offset to $2n\pi$,we can control the robot at desired angular pose through comparing angleTurned and angle.

The Experiment design the trajectory of the robot is square, we evaluate the motion error by comparing the pose of the

start point and the end point.
Design trajectory: square with 1m side length, translation velocity is 0.2m/s, angular velocity is 0.7rad/s.
Pseudo code of test Algorithm :

```
Repeat  4 times:
        divex（1，0.2）
    turnx（radius(90)，0.7）
End repeat
```

The experiment result shows our approach is stable and applicable. Figure4 is the actual trajectory of our robot, from the trajectory we can conclude that our means can construct the desired differential wheeled mobile robot platform.

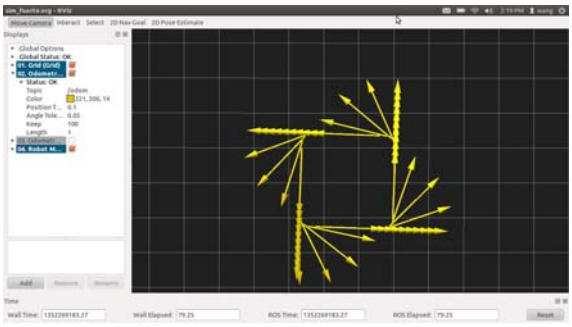

Figure4 vehicle motion analysis

In the next step, some further tests will be carried out, such as navigation with complicated trace, obstacle avoidance etc.

}

REFERENCES

[1] Gregor Klanč ar , DragoMatko, Saš o Blaž ič , A control strategy for platoons of differential drive wheeled mobile robot[J],Robotics and Autonomous Systems 59 (2011) 57–64.

[2] Soonshin Han, ByoungSuk Choi, JangMyung Lee. A precise curved motion planning for a differential driving mobile robot[J]. Mechatronics 18 (2008) 486–494.

[3]Thrun S，Fox D, Burgard Probabilistic Algorithms and theInteractive Museum Tour-guide Robot Minerva[J], InternationalJournal ofRobotics Research,2000,19(1-1)：972-999.

[4]Thrun S, Fox D, Burgard W. Probabilistic Robotics [M]. Massachusetts:MIT Press,2005. 167-193.

[5]Ming Tan, De Xu. Advanced Robot Control [M].Beijing:Higher Education Press,2007. 400-410.