Sensor-Based Path Planning and Tracking Control Scheme for Non-Holonomic Wheeled Mobile Robot

Mohamed Khan Aftab Ahmed Khan, Irraivan Elamvazuthi and Mumtaj Begam

1Universiti Industri Selangor, Kampus Bestari Jaya
Jalan Timur Tambahan, 45600 Berjuntai Bestari
1khan@unisel.edu.my

&

2&3Department of Electrical and Electronic Engineering
Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh, Perak
2irraivan_elamvazuthi@petronas.com.my,
mumtajbegam@petronas.com.my

Abstract

This paper presents a sensor-based path planning and tracking control scheme for non-holonomic wheeled mobile robot in environments without obstacles. It addresses the problem of active sensing, i.e., the selection of robot motion, which makes the robot arrive in its desired goal configuration with maximum accuracy, given the available sensor information. The implementation of the control scheme on the laboratory prototype wheeled mobile robot is described in detail. The performance of control scheme is verified through series of experiments.

Keywords: sensor, path planning, control scheme, non-holonomic wheeled mobile robot

1.0 Introduction

Wheeled mobile robots (WMRs) are increasingly present in industrial and service robotics, particularly when flexible motion capabilities are required on reasonably smooth ground and surfaces [1]. Kinematics study of several configurations of WMRs can be found in [2]. Beside the relevance in industrial applications, the problem of autonomous motion planning and control of WMRs has also attracted the interest of researchers in view of its theoretical challenges [3].

There are considerable research efforts towards solving the mobile robot navigation problem in different applications in in-door or outdoor environments (see [4] and [5]). New methods of control system for an autonomous robot using artificial intelligence can be found in [6] and [7]. The main goal of research on reactive navigation strategies is to allow autonomous units, equipped with relatively low-cost sensors and actuators, to perform complex tasks in uncertain or unknown environments [8]. These technologies have a wide range of potential application fields, which include the exploration of inaccessible or hazardous environments, industrial automation, and also biomedicine. In this research area, the development of the decision and control strategies necessary for autonomous operation plays a central role [8, 9]. In most mobile robot applications two basic position-estimation methods are employed together: absolute and relative positioning [10]. Relative positioning is usually based on dead-reckoning (i.e., monitoring the wheel revolutions to compute the offset from a known starting position). Dead-reckoning is simple, inexpensive, and easy to accomplish in real-time. The disadvantage of dead-reckoning is its unbounded accumulation of errors [11].

This paper presents a sensor-based path planning and tracking control scheme for a WMR. It addresses the problem of active sensing, i.e., the selection of robot motion, which makes the robot arrive in its desired goal configuration with maximum accuracy, given the available sensor information.
2.0 Differential Drive Kinematics

Dead reckoning is the process of calculating your position relying on a previously determined position. The mathematical relations used for dead reckoning differ from a robot to another due to the steering mechanism differences [12]. The derivation of a differential drive kinematics model of WMR is given below based on Figure 1. For detail derivation, see [13] and [14].

\[ r = \text{nominal radius of each wheel} \]
\[ L = \text{distance between \ Y_m \ o wheels} \]
\[ R = \text{instantaneous curvature radius of the robot trajectory, relative to the mid-point of the wheel axis} \]
\[ ICC = \text{instantaneous centre of curvature} \]
\[ R - (L/2) = \text{curvature radius of trajectory described by left wheel} \]
\[ R + (L/2) = \text{curvature radius of trajectory described by right wheel} \]

\[ v_r(t) = \text{linear velocity of right wheel} \]
\[ v_l(t) = \text{linear velocity of left wheel} \]
\[ w_r(t) = \text{angular velocity of right wheel} \]
\[ w_l(t) = \text{angular velocity of left wheel} \]

With respect to ICC the angular velocity of the robot is given as follows:
\[ w(t) = v_r(t)/(R + (L/2)) \] (5)
\[ w(t) = (v_r(t) - v_l(t))/L \] (6)

The instantaneous curvature radius of the robot trajectory relative to the mid-point of the wheel axis is given as
\[ R = (L(v_r(t) + v_l(t)))/(2(v_l(t) - v_r(t))) \] (8)

Therefore the linear velocity of the robot is given as
\[ v(t) = w(t)R = (v_r(t) + v_l(t))/2 \] (9)

The kinematics equations in the world frame can be represented as follows:
\[ \dot{x}(t) = v(t)\cos(\theta(t)) \]
\[ \dot{y}(t) = v(t)\sin(\theta(t)) \]
\[ \dot{\theta}(t) = w(t) \] (10)

The above equations can also be represented in the following form:
\[
\begin{bmatrix}
\dot{\theta} \\
\dot{x} \\
\dot{y}
\end{bmatrix}
= \begin{bmatrix}
0 & 1 \\
0 & 0 \\
-\sin(\theta) & 0
\end{bmatrix}
\begin{bmatrix}
v \\
w
\end{bmatrix}
\] (11)

The controlled variables of the model are the position and orientation of the mobile robot, while the control variables are the angular velocities of the left wheel and the right wheel.

3.0 System Description

The laboratory prototype WMR shown in Figure 2 is divided into two major sub-systems, i.e., mechanical and control sub-systems. The mechanical structure of the robot is made of aluminium. The control subsystem of all the robots is comprised of a PIC Microcontroller Board, VEXTA DC Brushless motor, DC brush motor, IR sensors, solenoids, and limit switches.
The main function of the control system is to generate electrical signals to effect motion based on input given by the control software. Considering the requirements of the autonomous robot, it was decided that the simplest and most cost effective way of achieving the sequence control was to use a PIC 16F877A microcontroller board which acts as the heart of the robot [15]. The specification of PIC 16F877A which is used in this board is given in Appendix.

### 3.1 Line Following Algorithm

Navigation algorithms are very important for autonomous robot to successfully achieve its goals, whether it is requested to accomplish a mission or simply to survive in the environment. In order to navigate effectively, the autonomous robot must be able to carry out the tasks shown in table 1.

#### Table 1. Navigation tasks

<table>
<thead>
<tr>
<th>No</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>know where it wants to go</td>
</tr>
<tr>
<td>2</td>
<td>know where it is and what direction it is facing</td>
</tr>
<tr>
<td>3</td>
<td>determine the heading (direction to) its destination</td>
</tr>
<tr>
<td>4</td>
<td>steer to and maintain the heading to its destination</td>
</tr>
<tr>
<td>5</td>
<td>stop when it has reached its destination</td>
</tr>
</tbody>
</table>

The above mentioned tasks were accomplished by utilizing the ‘line-following’ ‘counting’ algorithms with the aid of sensors. ‘Line following’ algorithm requires the robot to maintain its moving direction on the line, and make adjustments whenever it detects that it is not on the line. Table 2 provides the logic for the ‘line following’ algorithm using three sensors.

#### Table 2. Sensor logic instruction

<table>
<thead>
<tr>
<th>SEN1</th>
<th>SEN2</th>
<th>SEN3</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>OPTION</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>STRAIGHT</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>LEFT</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>LEFT_2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>RIGHT</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>RIGHT_2</td>
</tr>
</tbody>
</table>

From table 2, it can be seen that there are status for robot manoeuvrability such as Option, Straight, Left, left_2, Right and Right-2. These six status will determine the position of robot and how much offset level is needed during robot’s forward mobility. All ones indicate that the sensors are ‘active’ or detecting white line while all zeros indicate the sensors are ‘not active’ or ‘does not detect line’. Figure 3 shows the top view position of motor 1, motor 2 and and sensors for ‘line following’.

#### Figure 3. Top view of robot

Figure 4 shows the top view position of motor 1, motor 2 and sensors for ‘line following’ for the status ‘straight’, ‘left’ and ‘right’.

#### Figure 4. Position of sensors and motors for ‘line following’
From figure 4, it can be seen that by assuming that only one sensor is focused on white line while the rest are outside, meaning both SEN 1 and SEN 3 are LOW while SEN 2 is HIGH; this will make both motor 1 and motor 2 to be energised together and the robot will move forward in a STRAIGHT line. If the robot move a little bit to the right, both SEN1 and SEN2 are activated (detect white line), LEFT is executed and motor 1 will give little bit speed to the mobile unit to move a bit to the left. Right If SEN 2 and SEN 3 are HIGH while SEN 1 is LOW, meaning robot move little bit to the to the left, RIGHT is executed and motor 2 is energised and this makes the robot to move a bit to the right side.

In this work, tracking was done using three digital fiber sensors, NAVI FX-300 series from SUNX. NAVI is an abbreviation of New Advanced Sensor With visible Indicator [16]. Figure 5 shows position of three line following sensors, SEN1, SEN2 and SEN3 on autonomous robots.

Counting is a method that was applied to the robot especially when precision mobility is required. The idea of counting technique is focused on how to stop the robot at certain point when it starts moving from one point to another point. Figure 6 shows the exact position of sensor 4, SEN4 on the robot for counting purposes.

Programming the PIC is necessary to perform the desired application of the WMR. These codes enable the robot to do its explicit function, designed by the programmer depending on the tasks that are needed to be completed. The functional diagram of velocity control is shown in Figure 7 and the sample code is given in Appendix.
4.0 Discussion

A series of tests were carried out to determine the robustness of the algorithms. It was found that for line tracking, the position of the sensor and its arrangement on the WMR is very important due to its effect on the robot mobility. WMRs can have two drive techniques similar to today’s car which can be either front drive or back drive. Advantages of the front drive technique are that the WMR is able to accelerate faster with maximum torque and less vibration during motion. The advantage of the back drive is that offset can be reduced to minimum level so that sensors are able to respond faster to any state from STATUS condition. With this kind of arrangement, the WMR can move forward smoothly with minimum offset. The repeatability reaches 91% with slow trajectory profile. The efficiency of the movement is limited with the bandwidth of the sensor. The repeatability of fast trajectory was reduced to 73% with the same program implemented.

5.0 Conclusion

A sensor-based navigation algorithm for autonomous mobile robot was developed and tested. The robot was able to arrive in its desired goal configuration with maximum accuracy and perform the desired tasks effectively. The algorithm worked efficiently to correct the position of the robot even in high speed experiment. However, the efficiency of the movement is limited with the sensor bandwidth.

6.0 Acknowledgements

The authors are grateful to UNISEL and UTP for supporting this work.

7.0 References


Table A1. Specifications of PIC16F877A

<table>
<thead>
<tr>
<th>No</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40pin DIP, 40pin PLCC</td>
</tr>
<tr>
<td>2</td>
<td>35 instruction</td>
</tr>
<tr>
<td>3</td>
<td>200ns instruction timer (Tclock = 20MHz)</td>
</tr>
<tr>
<td>4</td>
<td>8k 14bit FLASH program memory</td>
</tr>
<tr>
<td>5</td>
<td>368 8bit data memory</td>
</tr>
<tr>
<td>6</td>
<td>256 8bit EEPROM</td>
</tr>
</tbody>
</table>

Appendix

Figure A1. PIC16F877A microcontroller diagram

```c
void main(void)
{
  // Init all function
  TRISA=0x00;
  TRISB=0x00;
  TRISC=0x0F;
  init_pwm();
  init_move();

  // Main Program
  moveLeft(300); // RAS1P until RST6 4.2 * 100 45% capacity hew
  moveRtl(300);  // raise wheels; right and left
  delayM(250);   // delay 2.5 second
  delayL(250);   // delay 2.5 second
  delayR(250);   // delay 2.5 second
  delayL(250);   // RAS1P
  delayR(250);   // RAS1P
  while(1);     // End program
}```

Figure A2. Sample pProgram in ‘C’