[2.5cm] Autonomous Mobile Robots

First assignment

OpenLoop Steering Assignment

Ruben Janssen, 10252657
David van Erkelens, 10264019
Laurens Verspeek, 10184465

Department of Computer Science
University of Amsterdam

November 15, 2013

Contents

1 Introduction

In this report the first findings of programming the NXT to follow a certain path given by circles and lines will be described.

2 Kinematic Models

The following formula is given:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\dot{\varphi}_1, \dot{\varphi}_2, l, r, l_c, d, r_c, \ldots) \tag{1}$$

When applying this to the differential drive robot, r denotes the diameter of the wheels and l denotes the distance between the origin of the robot and the wheels. A l and r with an underscore c denotes these distances for the castor wheel. The d denotes the distance between the wheel contact point and the center of rotation of the castor wheel. φ_1 and φ_2 denote the movement of the wheels. In figure 1, the allocation of φ_1 and φ_2 to the wheels is shown. In case of the omni-drive robot, φ_n denotes the movement of the n^{th} wheel and l and r have the same meaning as with the differential-drive robot.

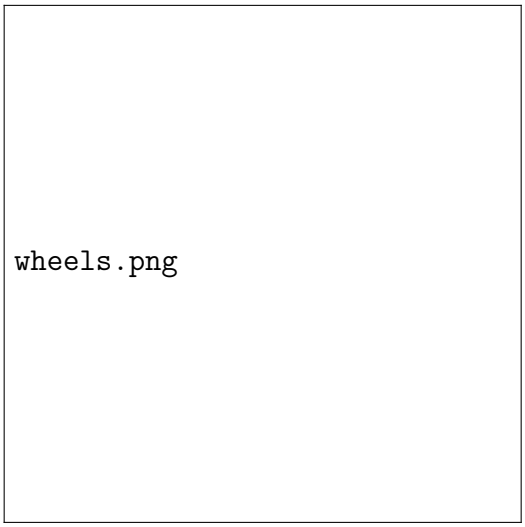

Figure 1: The allocation of φ_1 and φ_2

2.1 Establishing the kinematics

Moving onwards with formula 1, we can establish the kinematics of the differential-drive robot (shown in formula 2) and the omni-drive robot (shown in formula 3).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \dot{\varphi_1}, \dot{\varphi_2}) \tag{2}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \dot{\varphi_1}, \dot{\varphi_2}, \dot{\varphi_3}) \tag{3}$$

In the formula for the differential drive robot, l_c is left out since the influence of the castor wheel on the movement is negligible.

2.1.1 Odometry of differential drive robot

a

$$\xi_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r\dot{\varphi_1}}{2} + \frac{r\dot{\varphi_2}}{2} & 0 \\ 0 & 1 \\ \frac{r\dot{\varphi_1}}{2l} + \frac{-r\dot{\varphi_2}}{2l} & 0 \end{bmatrix} \tag{4}$$

b The larger Δt, the less precise the estimation of the position is. The rotation w increases when the difference between φ_1 and φ_2 increases (φ_1 and φ_2 are the speed of the wheels). The heading speed v increases when the sum of φ_1 and φ_2 increases.

3 Preparation for a real experiment

a First, the circle to be followed has to be determined, using a radius r. Then, the number of degrees to follow the circle should be determined, as g. To move the robot as desired, the distance to travel with each wheel should be calculated.

$$\begin{aligned} D_1(r) &= 2\pi \times (r - l) \\ D_2(r) &= 2\pi \times (r + l) \end{aligned} \tag{5}$$

In which D_1 notes the wheel closest to the center of the circle, and D_2 the wheel farthest from the center of the circle. Now the number of rotations to travel this distance should be calculated. This is done by dividing by the diameter d of the wheels:

$$\begin{aligned} Rotations_1(r) &= \frac{D_1(r)}{d_{wheel}} \\ Rotations_2(r) &= \frac{D_2(r)}{d_{wheel}} \end{aligned} \tag{6}$$

The speed of the wheel closest to the center of the circle will be lowered, so $\frac{\varphi_1}{\varphi_2} = \frac{Rotations_1(r)}{Rotations_2(r)}$. Then, φ_1 will become $\varphi_1 \times \frac{Rotations_1(r)}{Rotations_2(r)}$. After this, the `Tacho`-limit of the of the wheels will be set to the values of $Rotations_1(r)$ and $Rotations_2(r)$. Since $\frac{\varphi_1}{\varphi_2} = \frac{Rotations_1(r)}{Rotations_2(r)}$, both wheels will finish their rotations on the same moment.

To make the robot drive in a straight line, only the `Tacho`-limit is of importance. The `Tacho`-limit can be defined as a function f in a distance to travel t:

$$f_n(t) = 360 \times Rotations_n(r) \tag{7}$$

b In order to implement a feedback-loop, the `Tacho`-limit is removed. Instead, every 0.1 the value of both wheels is read to check if they already meet the value which would have been the `Tacho`-limit. If the values are incorrect, movement is reversed or continued. If the read value equals the desired regarding a certain threshold, movement of the wheel is stopped.

c For the Matlab implementation, see `circle.m`, `straightLine.m`, `drive.m` and `henk.m` (named after the robot).

4 Experiments

In order to test the code and formulas, the robot was tested in the *Robolab*. In the *Robolab* lies a football field used for Naos, but that field was perfectly suitable for testing our NXT. The NXT was programmed to follow the circle in the center of the field, which has a diameter of 60cm. In figure 2, the path of the NXT across this line is shown. On the pictures the NXT has a small offset from the line, but this can be explained by a faulty starting position.

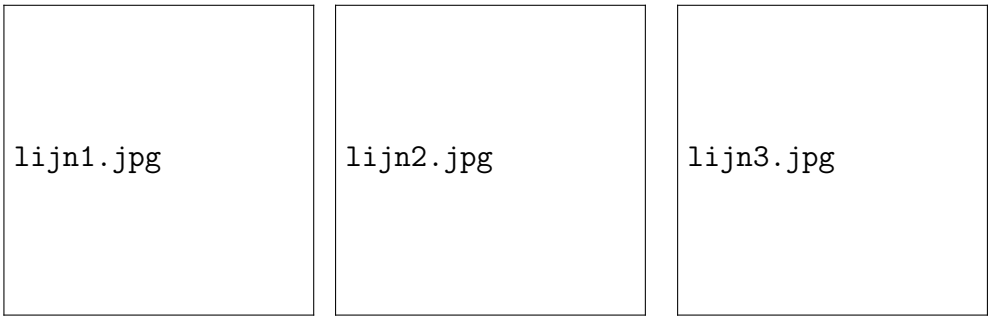

Figure 2: The NXT driving along the line

5 Conclusions

Looking at the experiments, it can be concluded that the formulas and code function, because the NXT was able to follow the line on the field pretty closely.