AMR Opdr1 Labbook

Wolf Vos 10197923, Tim Bloeme 6327303

November 2013

1 Introduction

The goal of this lab excercise was to program a robot wich is aware of where it is in a 2-dimensional plane while moving.

2 Odometry

The sampling time Δt will affect the precision of the robot in the global x and y coordinates in such a way that if you take a large sampling time the measurement will be less precise because you can only apply motor error correction to that interval. If you can do more correction on the same distance the location will be more precise. So a small Δt is better than a large Δt. But on the other hand if you take a Δt that is too small the algorithm will have to do many calculations that might slow down the process. The matrices below are used to convert the robots plane coordinates to the global reference plane coordinates. We use the inverse rotation matrix to compute this since:

$$Global * rotationmatrix = robot$$

$$Global = inverserotationmatrix * robot$$

θ is the angle between the robots x axis and the x axis in the global reference plane.

$$\begin{Bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{Bmatrix}$$

This is the matrix for the robots plane and where it is on its own x,y plane. φ is the power of the robots wheels. φ_1 for wheel 1 and φ_2 for wheel 2.
The l is the distance from the wheel to the center of the robot.

$$\begin{Bmatrix} \frac{2\varphi_1}{2} + \frac{2\varphi_2}{2} \\ 0 \\ \frac{2\varphi_1}{2l} + \frac{-2\varphi_2}{2l} \end{Bmatrix}$$

3 Implementation

We wrote a master function which used two functions for turning and going straight:

```
function [Z] = Turn(rad, draaicirkel, Z)
function [Z] = Straight(cm, Z)
```

We use these functions so that the robot can make turns and go straight for a certain distance. For going straight we just used the same amount of motor power and the same amount of rotations. For the turn we had to calculate the distance for each wheel and the power. The relation between the two different distances and the two different powers has to be the same. In order to calculate

the different distances that the two wheels have to go forward in order to make a turn we used the formulas below:

```
distB = rad*pi*(abs(draaicirkel-l));
distC = rad*pi*(abs(draaicirkel+l));
rotB = distB*360/17.854;
rotC = distC*360/17.854;
```

The value of 'draaicirkel' is the radius of the circle the robot will drive on. If the radius is bigger the turn ratio will be smaller and if the radius is smaller the turn ratio will be bigger. In order to enable the robot to turn both ways we made it possible to give a negative radius for turning the other way. The 'rad' variable will determine if the robot has to drive forward of backwards.

4 Problems we encountered

4.1 Communication

We found out that matlab sends everything at the same time wich causes problems for the motors. Since everything is send at the same time all the motor commands are being excecuted at the same time. This leads to an intresting problem. Only the first command recieved is actualy done all the other commands are running in the background of the NXT. When the first command is done it goes on to do the remainder of a task wich is running in the back ground. As a result the robot does not behave as suspected or required. The problem was solved by putting a pause command after each motor command.

5 Results

Our robot can drive on a certain predetermined line. We did not manage to implement the feedback control for the robot, so he might over or undershoot the target location after going straight or after turning. We had a problem with implementing the matrix system. Because of this it was of no use to implement feedback control since we did not know where the robot was in the world. Because of this it was impossible to determine the error of the motions of the robot.

We did however try to implement the matrixes but we had a problem with understanding the kinetics model. So as a result we implemented the motion controll for the robot with a different algrabraic algorithm