**University of Burgundy**

MsCV

**AUTONOMOUS ROBOTICS**

Lab 2 ODOMETRY

by

Gopikrishna Erabati and Pamir Ghimire

Supervisor: Dr. Xevi Cufí

**1. Introduction**

Odometry is the use of data from motion sensors to estimate change in position over time. It is used in robotics by some legged or wheeled robots to estimate their position relative to a starting location. This methodology is useful in small displacements. The advantage of this methodology is it's easy to use but it also prone to errors which would integrate on time.

**2. Theory**

One way of sensing the data from motion of robot is by the use of wheel encoders. In the e-puck robot, *there are no explicit encoders* but the software running would emulate encoders[1] which can be used to know the angular position of the wheel. The steps of two encoders are 16 bit registers which has a range of 0 - 65535.

This is done by the use of stepper motors to drive the wheels. The e-puck robot has 2 miniature steppers motors with gear reduction. The motor has 20 steps per revolution and the gear has a reduction of 50:1. When the stepper motor moves 20 steps the motor axis will rotate 1 round. When the motor axis moves 50 rounds, the wheel will rotate 1 round. Therefore, 1000 motor steps are necessary to rotate the wheel 1 round.

Therefore, one turn of wheel = 1000 steps.

one turn of wheel = = 2 \* 3.14 \* 0.0205 m = 0.12874 m.

Therefore,

**3. Compute Odometry**

The change in position will depend on the rotation of the wheels, on its radius and the width between them. The following figure shows all the relevant information necessary to compute the odometry [2]:



Figure 1 To know the things related to calculations

where,

{F} - Fixed coordinate system

{R} - Robot coordinate system at time 't'

{Rt+1} - Robot coordinate system at time 't+1'

rt = {rx,t + ry,t + rθ,t} - position and orientation of {Rt}

rt+1 = {rx,t+1 + ry,t+1 + rθ,t+1} - position and orientation of {Rt+1}

w - distance between wheels

dr - distance travelled by right wheel

dl - distance travelled by left wheel

d - distance between {R} and {Rt+1}

We can compute the next state {Rt+1} using:

**4. Tasks**

*Note: The code is well documented for all the tasks and written in modular approach with functions for each task. Uncomment the functions in main function to perform the tasks. Webots [3] software is used for simulations.*

**4.1 Computing the position of robot**

We initialize the position of robot as r = {0,0,0}. In the code *getPosition()* function makes the robot to move around a circle and we can check in console the position vector of robot and when it comes back to initial position after completion of 360 degrees the position vector almost reaches the initial position of robot.

**4.2 Translation only movement**

In the code, *move10cms()* function implements this task using odometry module.

**Logic**: Firstly we shall store the current pose of robot and move the robot forward and check the Euclidean distance between updated pose and initial pose and if reaches 10 cms with a given tolerance the robot stops.

We implemented this task on the real e-puck robot practically and obtained the following readings in Fig. 2.

Figure 2 Histogram of distances travelled by robot before stopping ahead of 10cms

*The mean of distances is 9.22 cms and standard deviation is 0.13 cms.*

**4.3 Rotation only movement**

In the code, *rotate90cw()* implements this task using odometry.

**Logic**: Firstly we shall store the current angle of robot and rotate the robot clockwise and check the difference between updated angle and initial angle and if it reaches 90 degrees with a given tolerance the robot stops.

**4.4 Perform a square trajectory**

In the code, *moveSquare()* function implements this task using the move10cms and rotate90cw modules.

All the tasks are implemented in simulation and tried on real robot. The video links of simulation and real are as given below:

For simulation video [click here](https://youtu.be/r9PT5lEyPI8)

For real video [click here](https://youtu.be/KJ7HBOvCNN8)

**5. Errors in Odometry**

There are two types of errors: Systematic (deterministic) and Occasional (non-deterministic).

* Systematic errors may occur due to:

1. Small differences between the diameters of the wheels

2. Incorrect alignment beetwen the two wheels

3. Discrete resolution of encoders

* Occasional errors may occur due to:

1. Displacements in irregular terrains

2. Slippage and skid of wheels.

These errors can be eliminated by:

* Deterministic errors can be eliminated by proper caliberation of system.
* Non-determinsitic errros have to be described by error models.

**6. Conclusion**

We used odometry to estimate the position of robot and perofrm the required tasks. As the odometry is prone to integrate errors, it is not recommened to use odometry explicitly without use of other sensory information, proper caliberation of system and compensating errors using error models.

**References**:

[1] E-puck Mini Doc

[2] Lecture notes of Dr. Xevi Cufí

[3] Webots tutorial