

COEX-1

Cooperative Explorer

Aurélien Werenne
Advisor: Prof. B. Boigelot

University of Liège

Link to code: [Github](#)

March 12, 2019

Overview

1 General

2 Components

3 Control

4 Exploration & Mapping

5 Code

Overview

1 General

2 Components

3 Control

4 Exploration & Mapping

5 Code

The goal of the robot, named COEX-1, is to map an unknown environment in a centralized multi-agent setting.

The main features of the robot are:

- Following a black line
- Computing the travelled distance
- Detecting, classifying and handling intersections
- Avoiding obstacles
- Communicating with a central unit

Non-exhaustive list of used components:

- Microcontroller (Arduino Nano)
- Reflectance sensor array (Pololu QTR-MD-06A)
- Digital distance sensor (Sharp GP2Y)
- Magnetic encoders
- DC Motor (Pololu 150:1 micro metal gearmotor MP)
- Motor driver (L298N dual H-bridge)
- Battery (NiMH 7.2V)
- Bluetooth module (HC-05)

General

Structure (1/3)

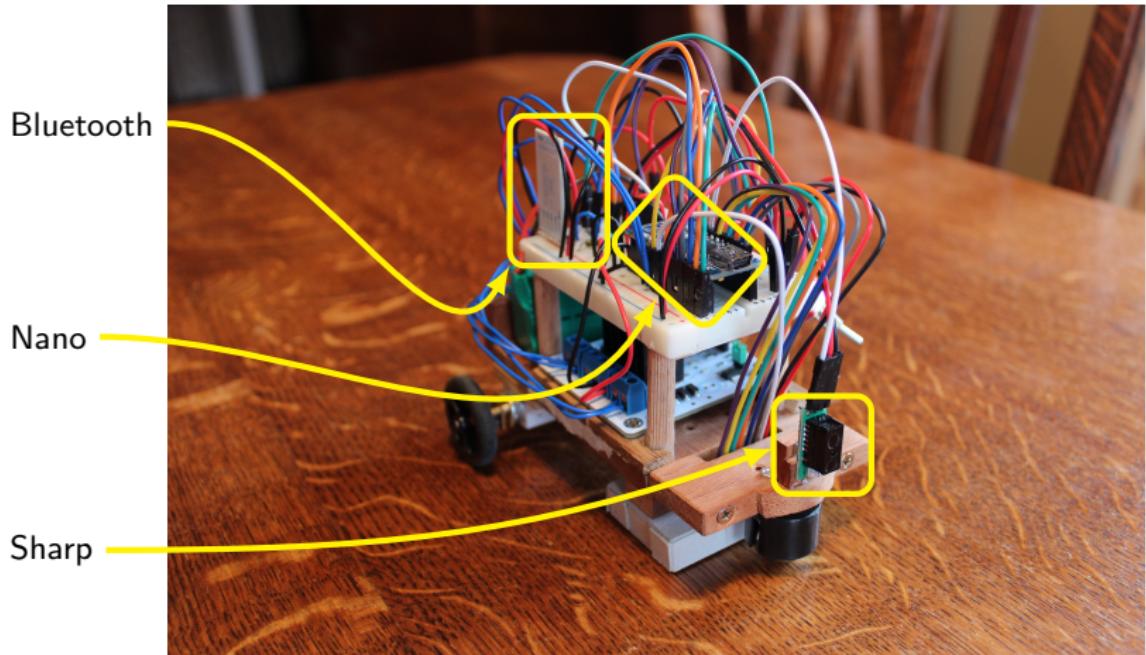


Figure: Front view of COEX-1

General

Structure (2/3)

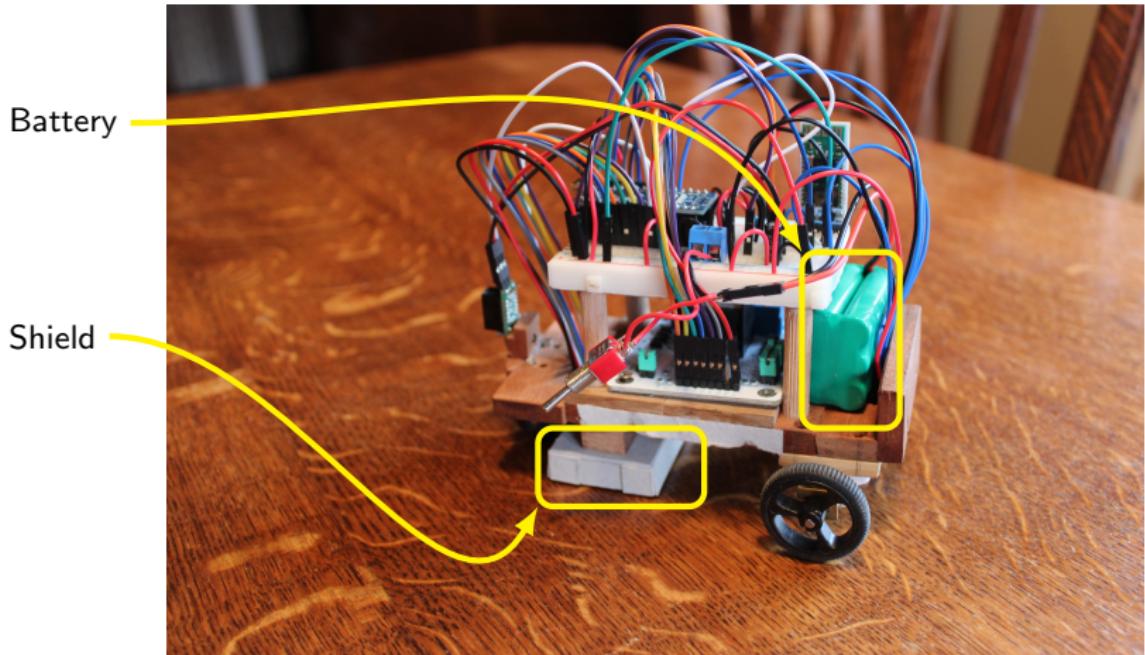


Figure: Side view of COEX-1

General

Structure (3/3)

Reflectance array

Motors

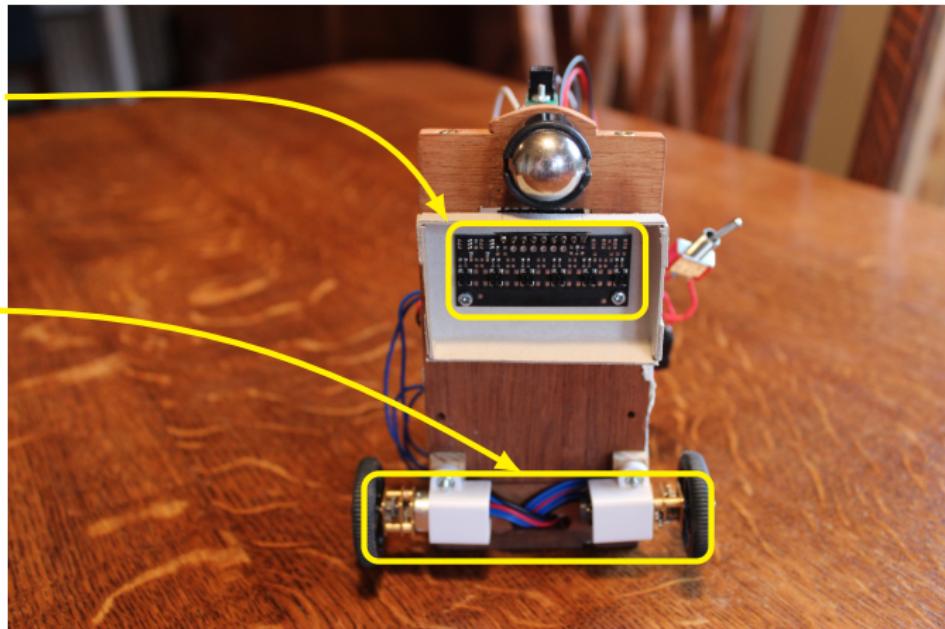


Figure: Bottom view of COEX-1

General Nomenclature

Let us explain some notions:

- error line: ...
- target speed: ...
- progress speed: ...
- measured speed: ...

Overview

1 General

2 Components

3 Control

4 Exploration & Mapping

5 Code

Components

Sharp sensor

Clean, no noise. Note we had to use bypass capacitor. Threshold at 2 volt, lower than middle because it is preferred to have False Positive than False Negatives.

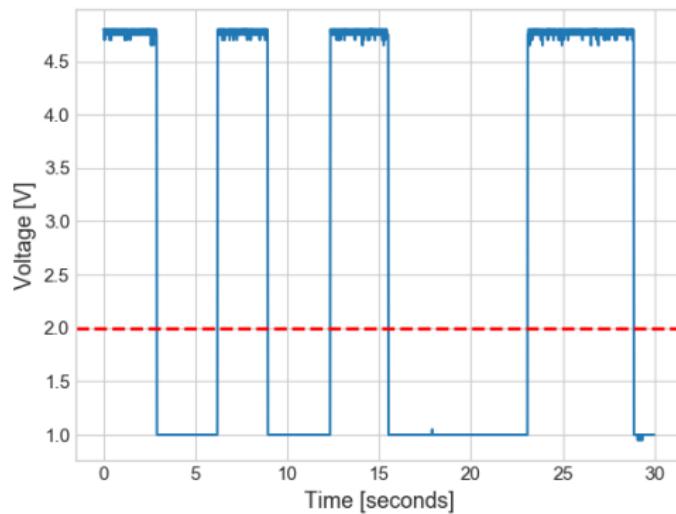


Figure: Back and forth movement of obstacle going in and out of reach of the sensor

Components

Reflectance sensor array (line sensor)

Experimentally, we found it is best to average four reading at 100 Hz. On Fig.12 we can see the detection varies smoothly as desired.

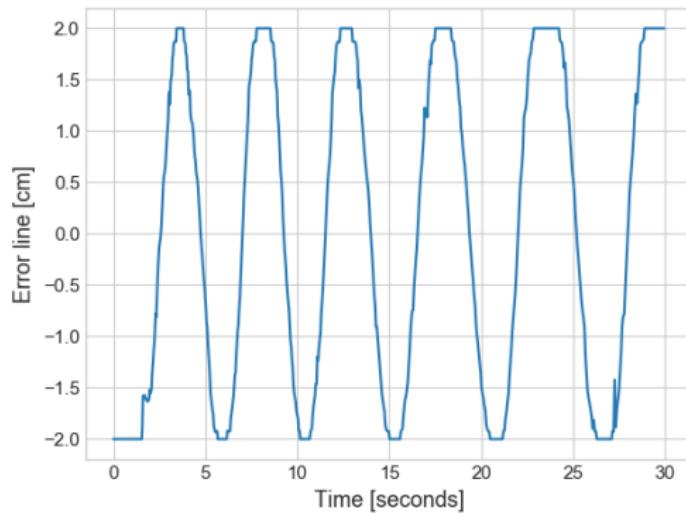


Figure: Moving black line back and forth between left and right side of the sensor

Components

Shield

In order for the robot to be able of climbing hills, the line sensors are fixed vertically well above the recommendations of the manufacturer.

This decision made the reflectance sensor particularly sensitive to ambient light interferences. As a solution a shield was constructed.

Components

Bluetooth module (1/5)

Sinus test with diagram. Why sinus, fitting etc. Conclusion.

Components

Bluetooth module (2/5)

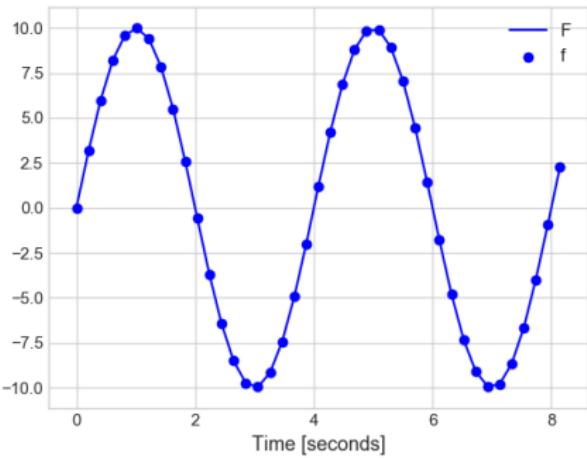


Figure: Sinus test at 5Hz

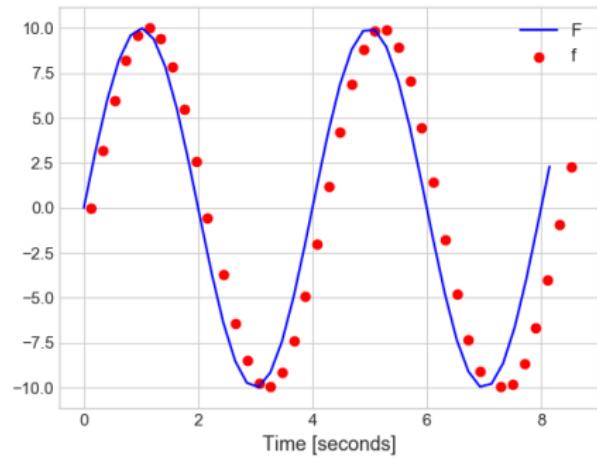


Figure: Sinus test at 5Hz

Components

Bluetooth module (3/5)

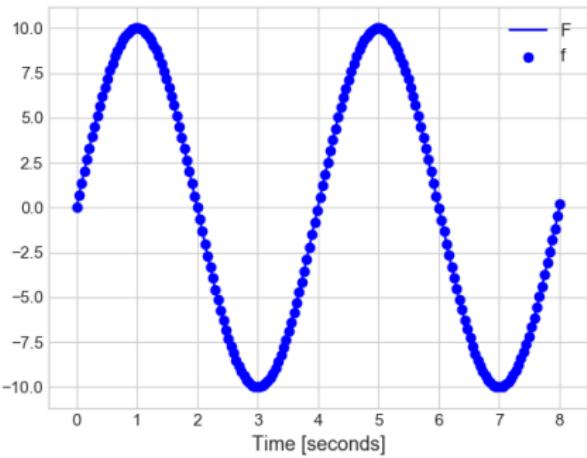


Figure: Sinus test at 25Hz

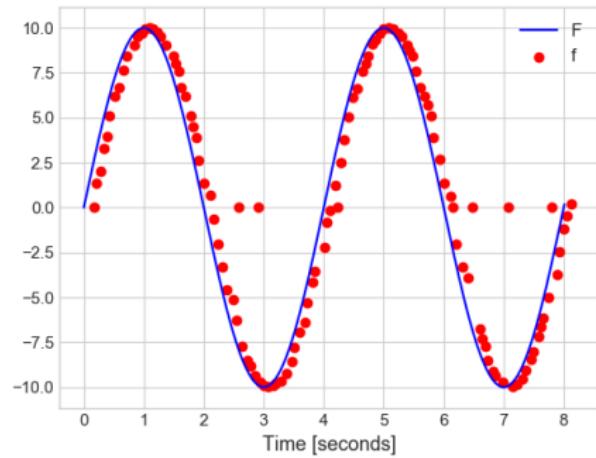


Figure: Sinus test at 25Hz

Components

Bluetooth module (4/5)

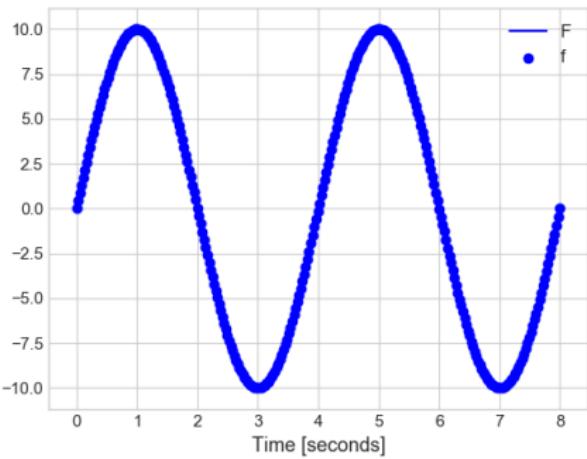


Figure: Sinus test at 40Hz

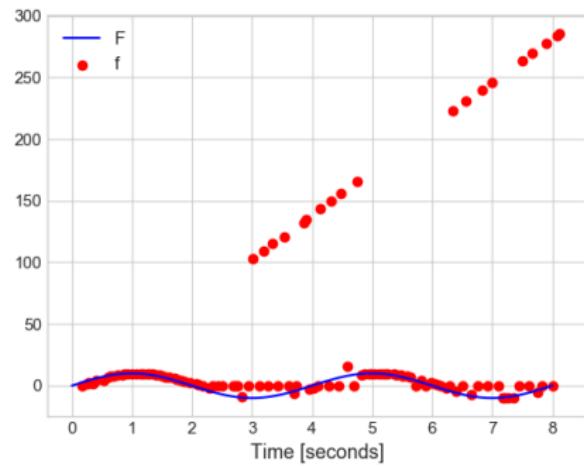


Figure: Sinus test at 40Hz

Components

Bluetooth module (5/5)

A voltage divider was used because of the difference in logic level voltage between the Arduino (0 – 5 Volt) and the bleutooth module (0 – 3.3 Volt).

$$V_{arduino} = \frac{R_2}{R_1 + R_2} V_{bleutooth}$$
$$= \frac{2.2}{3.2} V_{bleutooth}$$

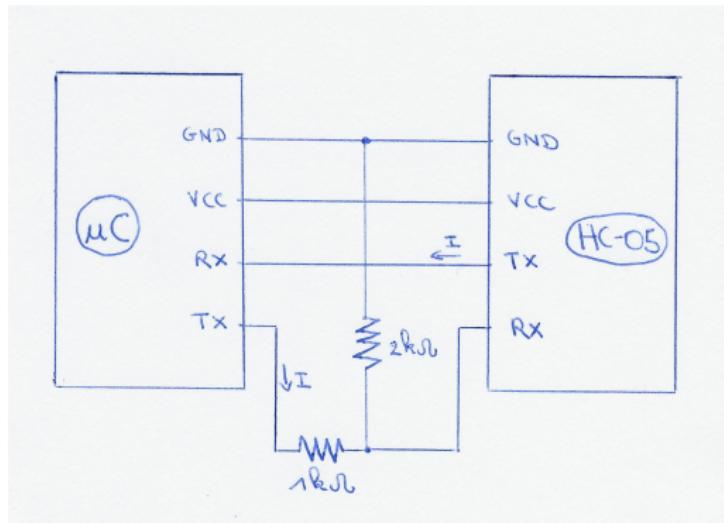


Figure: Voltage divider

Components

Quadrature encoders (1/2)

Magnetic quad encoders using Hall effect to count the number of revolutions of the motor shaft: n_L and n_R . The angular velocity of the wheel is computed as follows:

$$w_L = 2\pi f_L \quad \text{with} \quad f_L = \frac{n_L}{N' G_b \Delta t}$$

with N' the number of counts per revolution per channel, and G_b the gearbox ratio. The robot can be modelled as a differential steering wheel such that,

$$v = \frac{v_L + v_R}{2} = \frac{R(w_L + w_R)}{2}$$

Components

Motors

On Fig.20 we observe small differences between left and right motor and it even depends on the charge. To obtain a robust robot, a regulated system need to be designed.

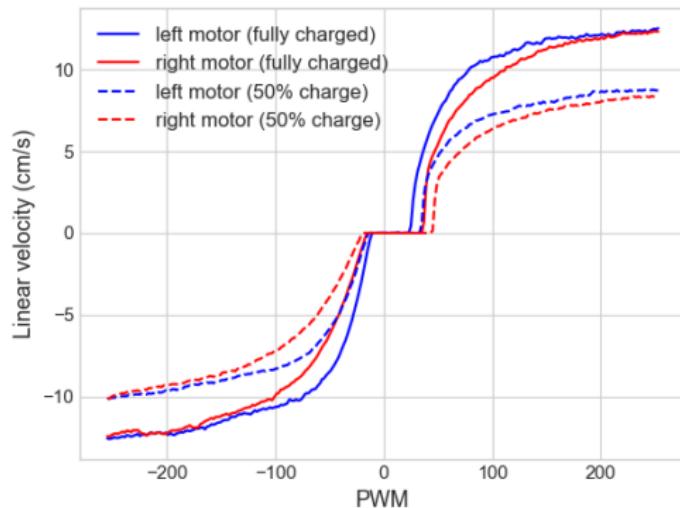


Figure: Relationship between PWM (input) and measured speed (output)

Overview

1 General

2 Components

3 Control

4 Exploration & Mapping

5 Code

An auto-regulated system is achieved by using several PID controllers. The output of a PID controller can be expressed as

$$o_n(e_n; e_1 \dots e_{n-1}) \leftarrow K_p e_n + K_d \frac{e_n - e_{n-1}}{\Delta t_{n-1:n}} + K_e \sum_{i=0}^n e_i \Delta t_{n-1:n}$$

with the following two precautions: anti-windup and anti-derivative kick:

$$o_n = \begin{cases} \max & \text{if } o_n > \max \\ \min & \text{if } o_n < \min \\ o_n & \text{otherwise.} \end{cases} \quad \text{and} \quad K_d = \begin{cases} 0 & \text{if } n < T \\ K_d & \text{otherwise.} \end{cases}$$

General framework (2/2)

The three main controllers are used to control the speed, direction and aligning when turning (more explanations further).

The pwm value is computed from the three controllers outputs:

$$\left\{ \begin{array}{l} \alpha = o(e_{direction}) \\ \beta = o(e_{speed}) \\ \gamma = o(e_{align}) \end{array} \right. \Rightarrow \left\{ \begin{array}{l} pwm_L = (\beta - \gamma) + \alpha \\ pwm_R = (\beta - \gamma) - \alpha \end{array} \right.$$

Direction control (1/2)

We define three type of desired direction behaviours:

- *forward*: going in straight direction, setpoint value put such that
 $v_L = v_R \Rightarrow e_{direction} = v_R - v_L$
- *line-following*: adapt direction in order that line always centered,
 $e_{direction} = err_{line}$
- *turning*: to obtain pure rotation, the following constraint must be satisfied,
 $v_L = -v_R \Rightarrow e_{direction} = v_L + v_R$

Control

Direction control (2/2)

On Fig.12, different trajectories of line-following on a straight line with initial disalignment are plotted.

Control

Speed control (1/5)

The error in speed control (e_{speed}) is the difference between progress and measured speed. For a uniform acceleration as in Fig.26, the progress speed is computed as follows.

$$v_{n+1} \leftarrow v_n + A \Delta t_{n:n+1} \quad \text{with} \quad A = \frac{v_{target}}{\bar{T}}$$

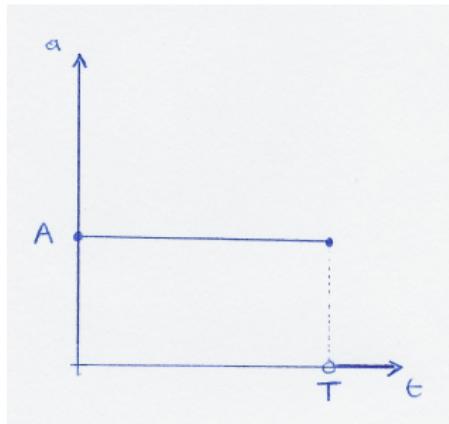


Figure: Profile of uniform acceleration

Control

Speed control(2/5)

For a smoother transition (based on instinct) the robot accelerates following the profile as Fig.27. However, we want the robot to achieve the target speed in the same amount of time as uniform. This is translated in eq ref.

$$\Leftrightarrow \int_0^T a(t) dt = \int_0^T a'(t) dt$$
$$\Leftrightarrow A T = \underbrace{d B}_{\text{triangles}} + \underbrace{(T - 2d)B}_{\text{rectangle}}$$

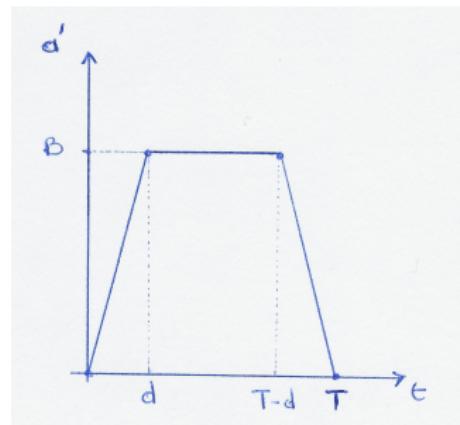


Figure: Profile of smooth acceleration

Control

Speed control (3/5)

By introducing the parameter $\psi = \frac{d}{T}$, equation ref can be rewritten as

$$B = \frac{A}{1 - \psi}$$

The update rule for the progress speed becomes

$$v_{n+1} \leftarrow v_n + \int_n^{n+1} a'(t) = v_n + \int_0^{n+1} a'(t) - \int_0^n a'(t)$$

Note that splitting the integrals permits us to simplify calculations by making use of fixed surfaces.

Control

Speed control (4/5)

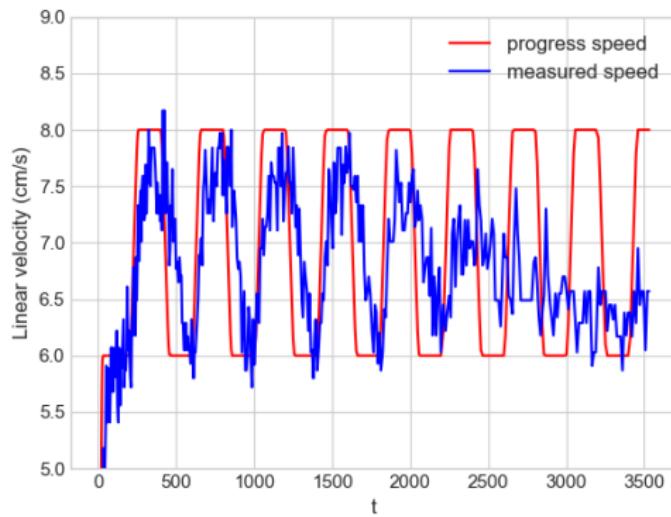


Figure: Full view

Control

Speed control (5:5)

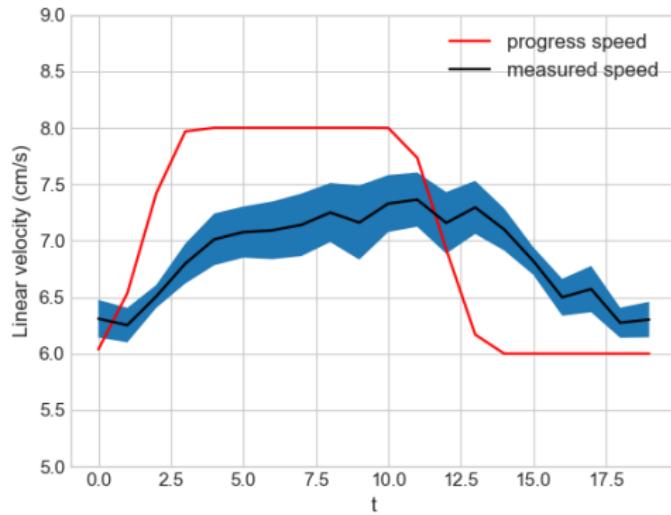


Figure: +- 2 standard error with mean

We model the COEX-1 as a two-wheel robot as in Fig.31. By integrating the angular velocity under the assumption that $v_R = -v_L$, we obtain

$$\theta(t) = \frac{2vt}{b} + \theta_0$$

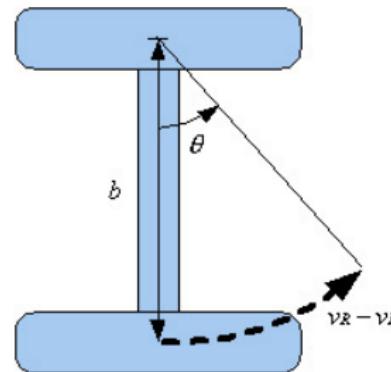


Figure: Two wheel model

Based on equation ref, we can make the robot turn Θ radians as follows:

- 1: $\theta \leftarrow 0$
- 2: TURN(v_{target})
- 3: **while** $\theta < \Theta$ **do**
- 4: $\theta_{n+1} \leftarrow \frac{2 v_n \Delta_{n:n+1}}{b}$
- 5: **end while**
- 6: STOPMOTORS

Some error because not pure rotation, noise etc. Error seems reasonable 5 to 10 degrees. However not good enough for robust turning when exploring environment. Other method developed in section ref.

Overview

1 General

2 Components

3 Control

4 Exploration & Mapping

5 Code

To explore an unknown environment structured as a maze it needs to be able to:

- Compute the distance travelled since the last intersection (slide ref)
- Detect an intersection (slide ref)
- Turn to the desired intersection (slide ref)

Exploration & Mapping Distance (1/5)

First method: Simple equation - explain without mathematics.

Second method: Upperbound of error can be reduced due to relation with distance.

plot relationship

Exploration & Mapping Distance (2/5)

Plot comparison with error bars of two methods on test set.

Conclusion.

Exploration & Mapping Distance (3/5)

Explain we could try to quantify uncertainty and above certain threshold not accept it.

$$MSE = \sum_{i=0}^N err_{line}^2$$

obtained iteratively by rolling mean method. One plot

Exploration & Mapping Distance (4/5)

Explain simpler choice is to discretize from second method discretization.

Reward 7.5-12.5

Plot discretization on test set.

Exploration & Mapping

Distance (5/5)

Example of resulting big map. Explain annotation

Map.

Exploration & Mapping

Intersection detection

Approach developed to intersection detection is based on majority vote idea. x_{loc} takes value 1 if a black line is detected at location loc. From moment anomalie detected we start following procedure.

$$y_{loc} = \underset{x \in \{0,1\}}{\operatorname{argmax mode}}(x_{loc})$$

Boolean value for intersection based on decision rule as follows

$$(y_{center} = 0) \vee (y_{left} = 1) \vee (y_{right} = 1)$$

Exploration & Mapping

Intersection classification

We use the values computed during intersection detection, and we make the robot go a bit further forward. The type of intersection is determined and takes values in following set:

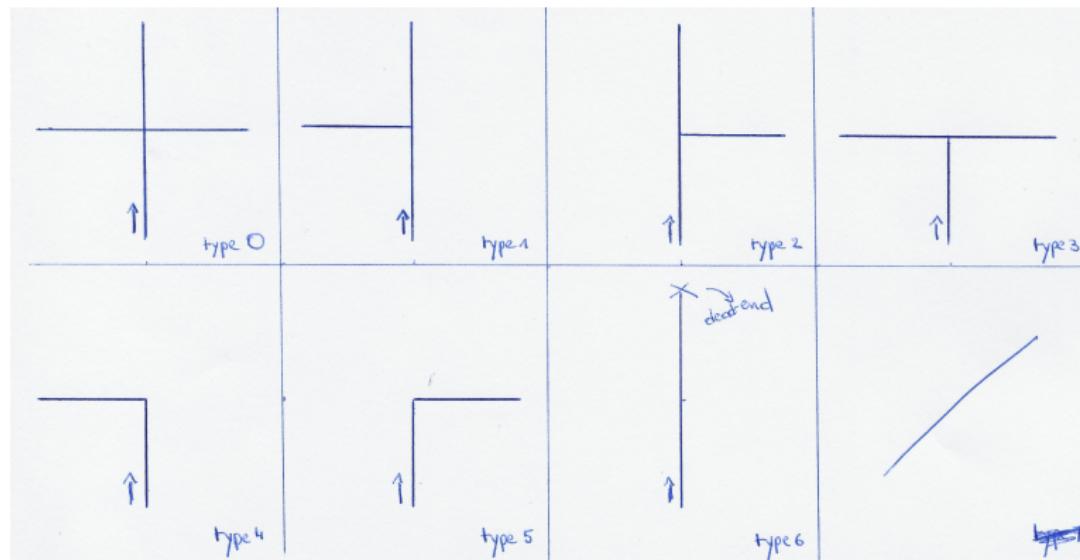


Figure: The 7 types of intersections

Exploration & Mapping

Turning & alignment (1/2)

Compute quadratic interpolation to fi points $(\pm 2, v_{init})$ and $(\pm \epsilon, v_{min})$.

$$v(x; a, b) = a(x - \epsilon)^2 + b$$

with initial conditions $v(x_{init}) = v_{init}$ and $v(\epsilon) = v_{min}$. Thus

$$a = \frac{v_{init} - v_{min}}{(2 - \epsilon)^2} \quad \text{with} \quad b = v_{init}$$

Exploration & Mapping

Turning & alignment (2/2)

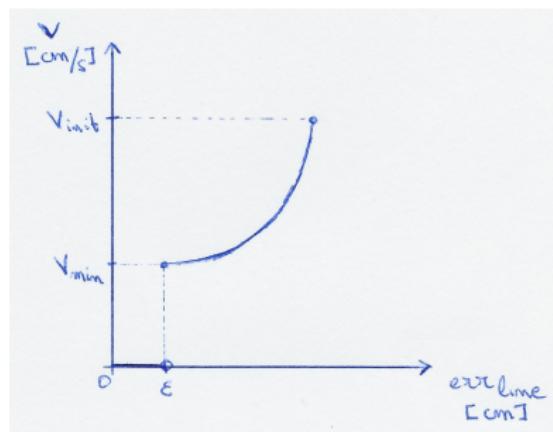


Figure: The 7 types of intersections

Explain why not such a good idea.

Overview

1 General

2 Components

3 Control

4 Exploration & Mapping

5 Code

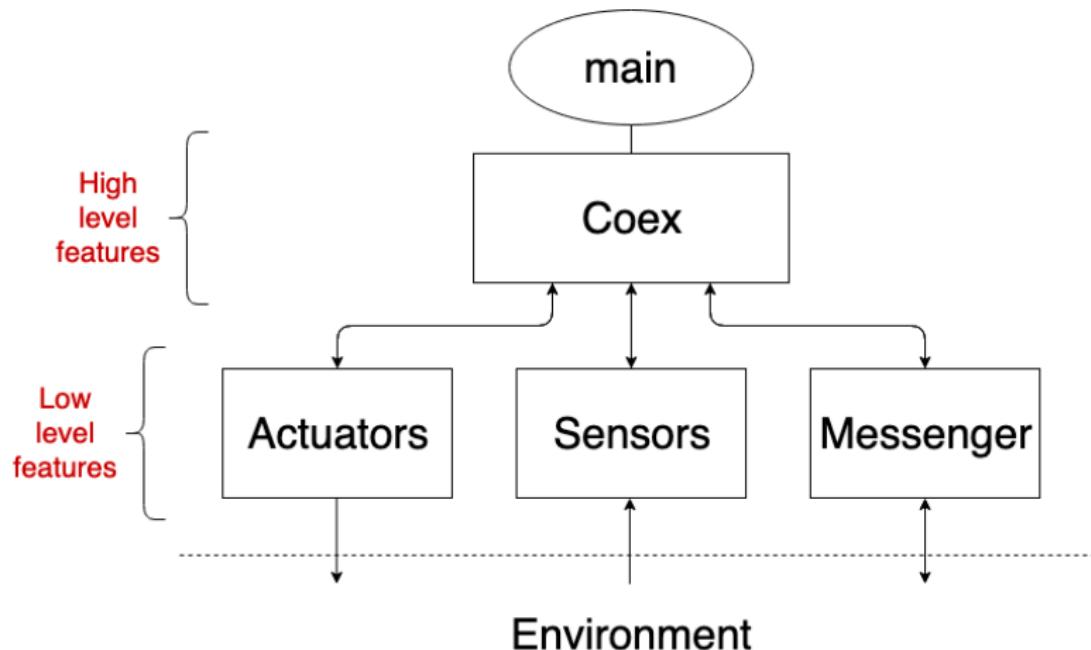


Figure: Architecture of the code

