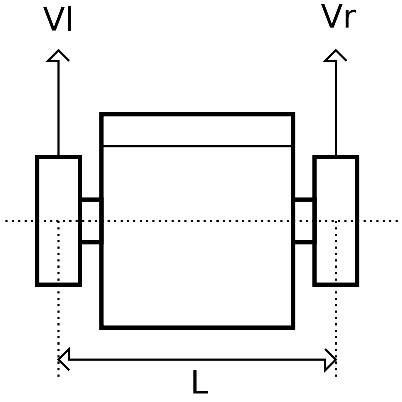
Interval Localization with Landmarks

Final TD for the Interval Analysis module

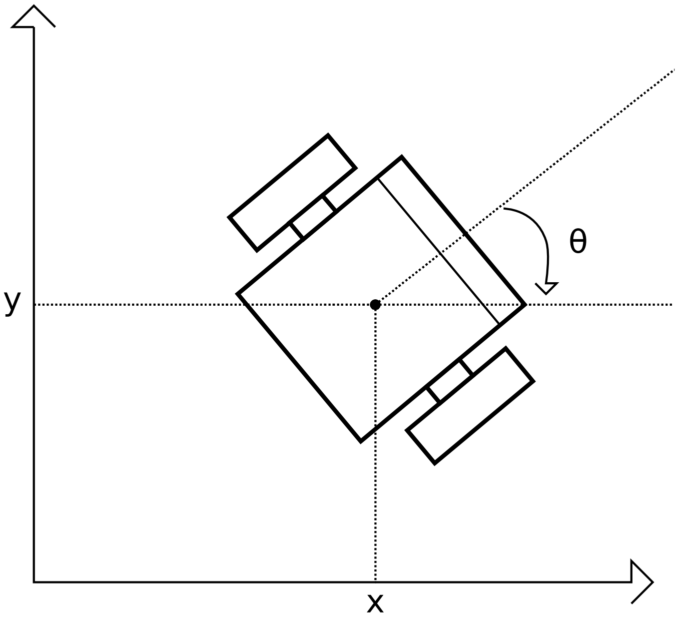
The robot

- Two wheeled differential robot
- Vl and Vr : linear wheel speed
- L : distance between the wheels



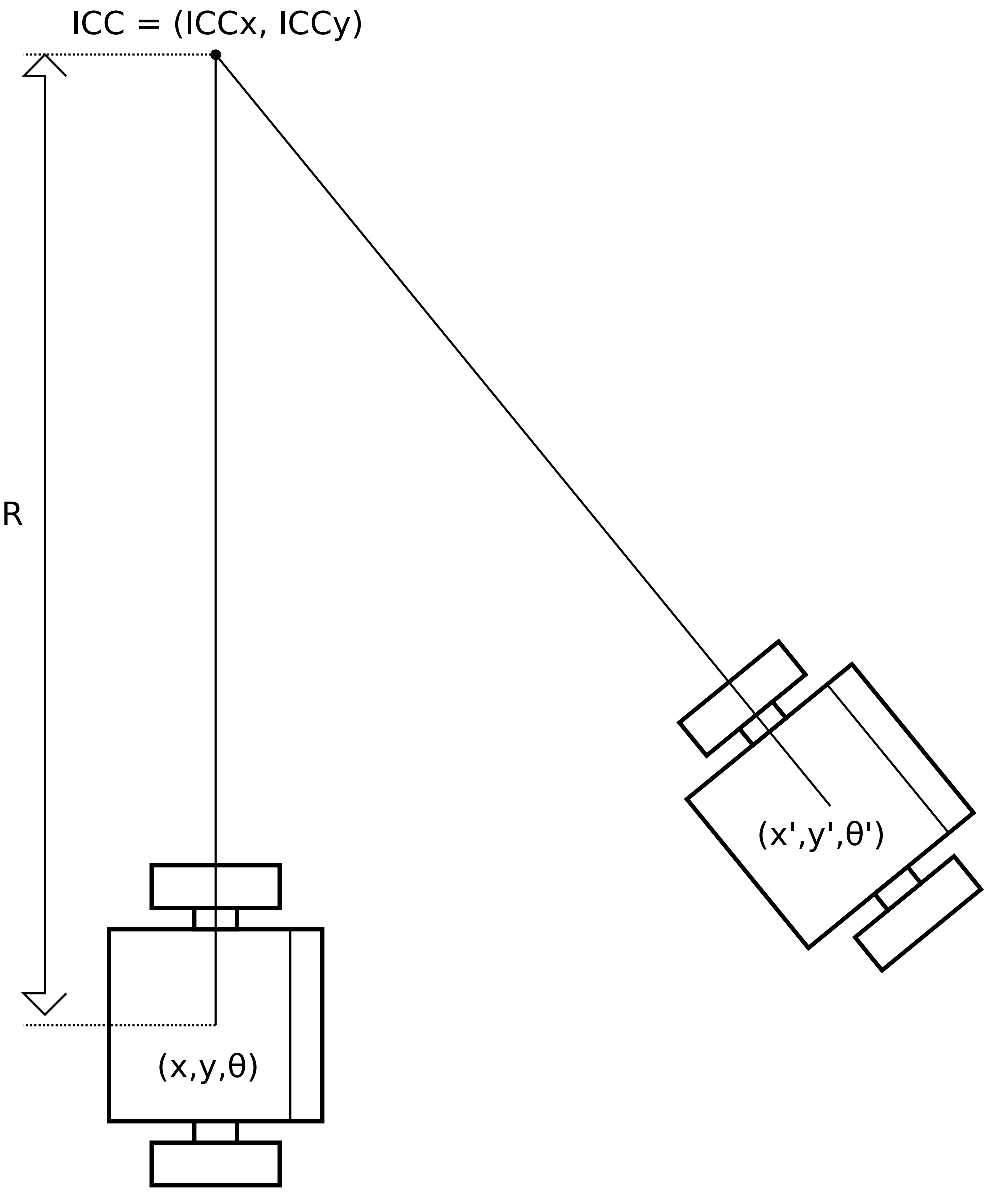
The robot

- Robot state : (x,y,θ)
- (x,y) position, θ orientation



The robot

- Robot dynamics



The robot

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} \cos(w.\delta_t) & -\sin(w.\delta_t) & 0 \\ \sin(w.\delta_t) & \cos(w.\delta_t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x - ICCx \\ y - ICCy \\ \theta \end{pmatrix} + \begin{pmatrix} ICCx \\ ICCy \\ w.\delta_t \end{pmatrix}$$

- With
  - δ_t : the time step between two poses

$$ICCx = x - R.\sin(\theta)$$
$$ICCy = y + R.\cos(\theta)$$
$$R = \frac{L}{2} . \frac{Vr + Vl}{Vr - Vl}$$
$$w = \frac{Vr - Vl}{L}$$

The simulated robot

- MoveRandomInBox()
  - Moves the robot radomly inside a box given in parameter
  - Returns « Vl_msr » and « Vr_msr », estimations of the Vl and Vr values such that

$$Vl \in [Vl_{msr} - \epsilon_{drift} ; Vl_{msr} + \epsilon_{drift}]$$
$$Vr \in [Vr_{msr} - \epsilon_{drift} ; Vr_{msr} + \epsilon_{drift}]$$

- getDrift()
  - Returns the drift ϵ_{drift} bounded error value
- getL()
  - Returns L_msr an estimation of the L value of the robot
- getLError()
  - Returns the error ϵ_L over the estimated L value such that

$$L \in [L_{msr} - \epsilon_L ; L_{msr} + \epsilon_L]$$

- getDeltat()
  - Returns δ_t the time step

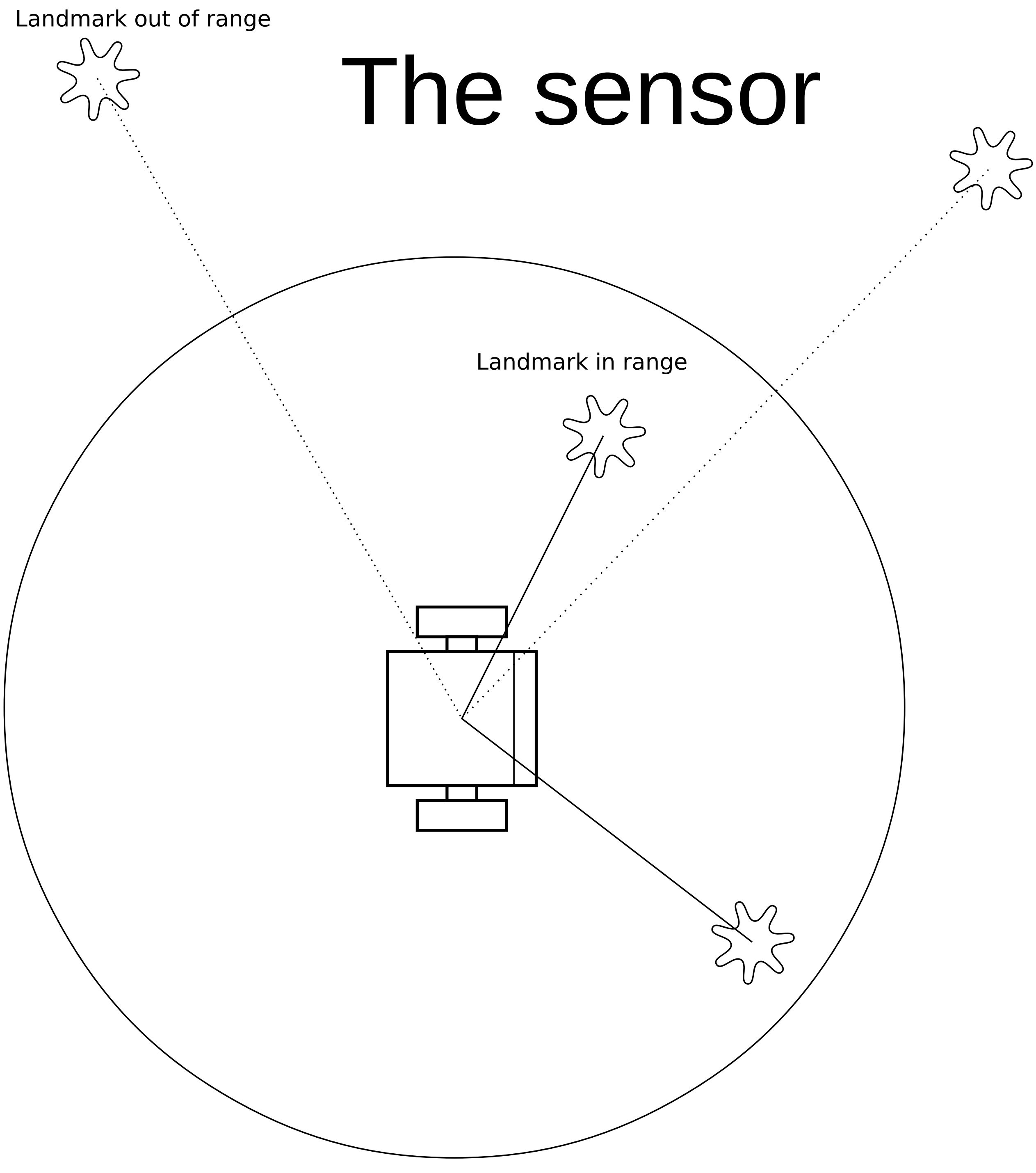
The simulated robot

- getOdometry()
  - Returns the odometry value (the distance between the old and the new position)
- getErrorOdometry()
  - Returns the odometry ϵ_{odo} error such that $\sqrt{(x-x')^2+(y-y')^2} \in [odo-\epsilon_{odo}; odo+\epsilon_{odo}]$
- getCompass()
  - Returns an estimation of the robot's direction
- getErrorCompass()
  - Returns the bounded error of the compass such that $\theta \in [\theta_{compass}-\epsilon_{compass}; \theta_{compass}+\epsilon_{compass}]$

The robot

- Global Variables :
  - gv.ROBOT : the robot variable
    - Use the previously defined functions
  - gv.I_POSE : the pose estimation of the robot (to keep updated)
    - gv.I_POSE.x : Interval for the x position
    - gv.I_POSE.y : Interval for the y position
    - gv.I_POSE.theta : Interval for the orientation theta

The sensor



The map

- Global variable
  - gv.l_LANDMARKS : an array of all the known positions of the landmarks

The sensor

- Global variables
  - gv.SENSOR : the sensor
- getMeasurement(idx)
  - Returns the value of the idx measurement
  - Returns -1 if the landmark is out of range
- getError()
  - Returns the value of the bounded sensor error

Work 2 do

- c_dst()
  - Distance contractor, already done before
- compute_i_pose()
  - Update the gv.I_POSE variable according to
    - The previous value of gv.I_POSE
    - The odometry data
    - The wheel command
    - The compass value
- Expected results : https://youtu.be/fZqS4Xxg1Co
- Using the simulator:
  - ‘m’ key to show/hide the landmarks and the robot
  - ‘Down’ key to randomly move the robot