Laboratory Work #5 – Robot localization based on beacons with validation module

Consider a robot with differential traction and a laser scan on top.

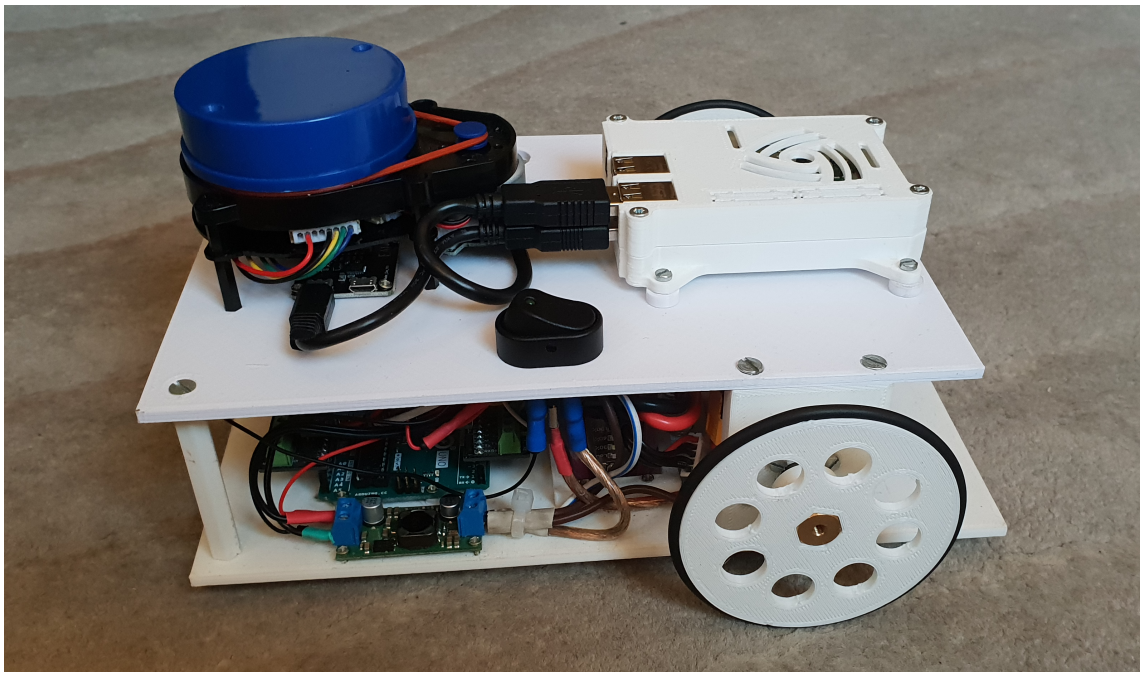


Fig 1 – Real robot

This robot was simulated in an environment with 3 circular beacons with 5 cm diameter placed at positions (-0.3, 1.3), (1.3, 1.3) and (0.5, 0.3), values in meters:

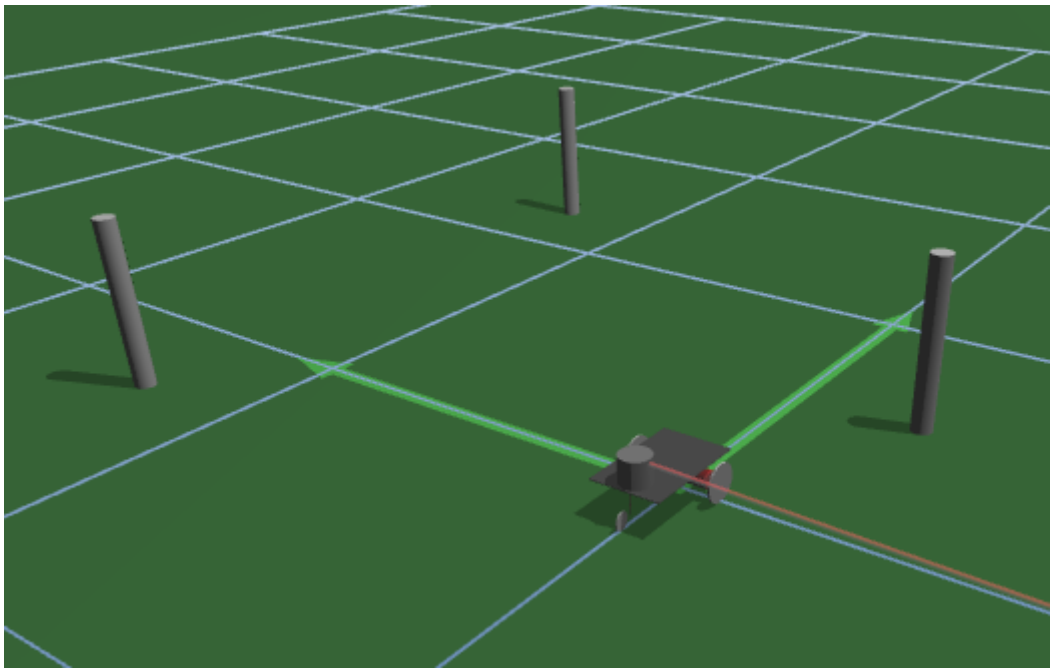


Fig 2- Simulation environment

Using the base code available at Sigarra contents:

a) Set the initial estimation for the robot state X equal to the real value (x = 0 m, y = 0 m and theta = 1.57 rad) at cells (33,3), (34,3) and (35,3). Use Sheet button "Global RESET", then "Chart" (turn ON Chart) and finally "FollowSquare". Look for the Graph and complete the full square trajectory. Notice the difference between the true robot position (red) and the estimated position (green) based only on the motion model.

b) Implement a detection and validation module for the beacons (see Control procedure, cycle "for i:= firstRay to LastRay"). The laser data vector starts with a distance measure correspondent do -180º. The bean angle step is 1º and the laser frame is centred with the same orientation than the robot frame (placed in the middle of the wheels with the X axis pointing forward and the Z axis pointing up) at a distance of 14 cm (see figures 1 and 2).

   To calculate the distance from the robot to the beacon, in the cycle "for j:=1 to NBEACONS", use the function Dist(x,y) that calculates the distance from the origin to the point (x, y) and in the angle, do not forget to use the function NormalizeAngle().

   Show in cells (1,1) to (3,3) the number of points detected in each beacon and the coordinates of the beacons in the global frame taking into account the estimated robot pose and the laser measures.
   Use the already created structure TClusterPos and variables BeaconCluster and BeaconPos.

c) Uncomment in the Control() procedure the line that calls the procedure "LocationFromSensors". Verify that the prediction phase of the Extended Kalman Filter is implemented with the procedures "PredictPosition(.)" and "EKF_MotionModel(.)" and the update phase with the procedure "EKF_Update(.)". Look for the code and compare with the equations presented in the theoretical lesson's slides.
   Set the initial estimation for the robot state X equal to x = 0.05 m, y = 0.05 m and theta = 1.57 rad (wrong initial estimation).

   Go to procedure "Initialize", near "//Extended Kalman Filter" and fill all the missing parts:
   qV := ...
   qOmega := ...
   rSensD := ...
   rSensA:=...

   The sensor distance noise is Gaussian with zero mean and standard deviation sensD_stddev = 0.005 m, defined at "const" section. The angle noise has also zero mean and standard deviation "sensA_stddev = 0.009 rad". With that values initialize the variables rSensD and rSensA (covariance matrix R).
   Tune motion model noise covariance (Q), using the constants lin_stddev and omega_stddev; Test in a full square trajectory the values 1E-6, 1E-2 and 1E-1.

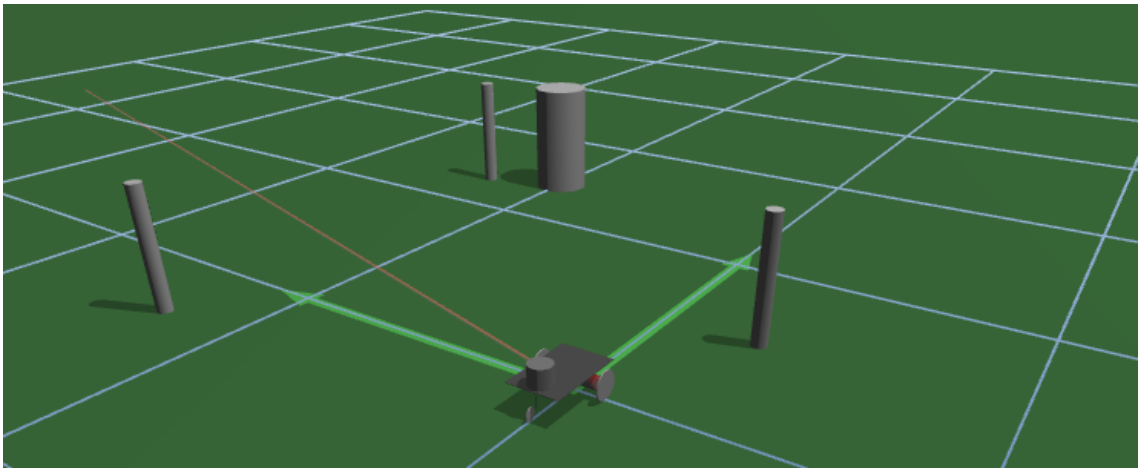
   Initialize also state estimation covariance matrix P (see ResetEstimation(.) procedure). Compare the initial evolution of the estimation with the initial values for P:

   i) cov(x) = cov (y) = 1E-4, cov(theta) = 3E-4
   ii) cov(x) = cov (y) = 1E-8, cov(theta) = 3E-8

d) Using the Scene editor (Ctrl + S) insert or remove a false beacons with the code:

```
<cylinder>
  <ID value='BeaconFalse'/>
  <size x='0.1' y='0' z='0.4'/>
  <pos x='1.4' y='1' z='0.2'/>
  <color_rgb r='128' g='128' b='128'/>
</cylinder>
```

Test the robustness of the validation module in the presence of new objects in the scene.



e) Using the Scene editor (Ctrl + S), or clicking and dragging, insert or remove one of the three beacons and check the evolution of the robot state estimation.