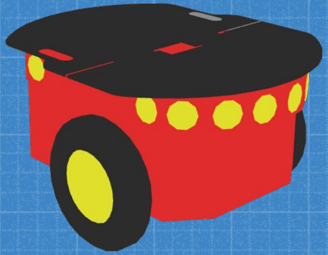


Week 3

Pioneer 3DX - SLAM
(Group 17)

Eufémio Marques
Ivan Figueiredo
Miguel Roldão
Pedro Matos

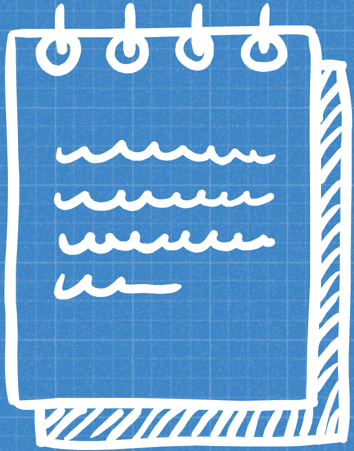
Plans for last week:

1. Create a simulation environment in Gazebo and explore existing topics.

2. Create a simple testing script to control the Pioneer and prevent it to exit a certain region.

3. Record data from simulated lasers and then use it on a new version of a script.

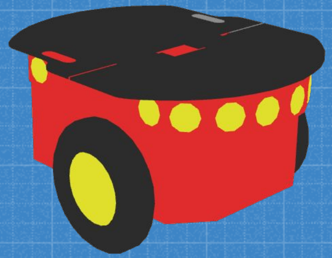
4. On EKF understand which are the necessary features…



Gazebo

We tested 2 different packages to simulate our pioneer:

- ***p2os*** maintained by Hunter Allen,
- ***pioneer_p3dx_model*** by mario-serna,



p2os maintained by Hunter Allen

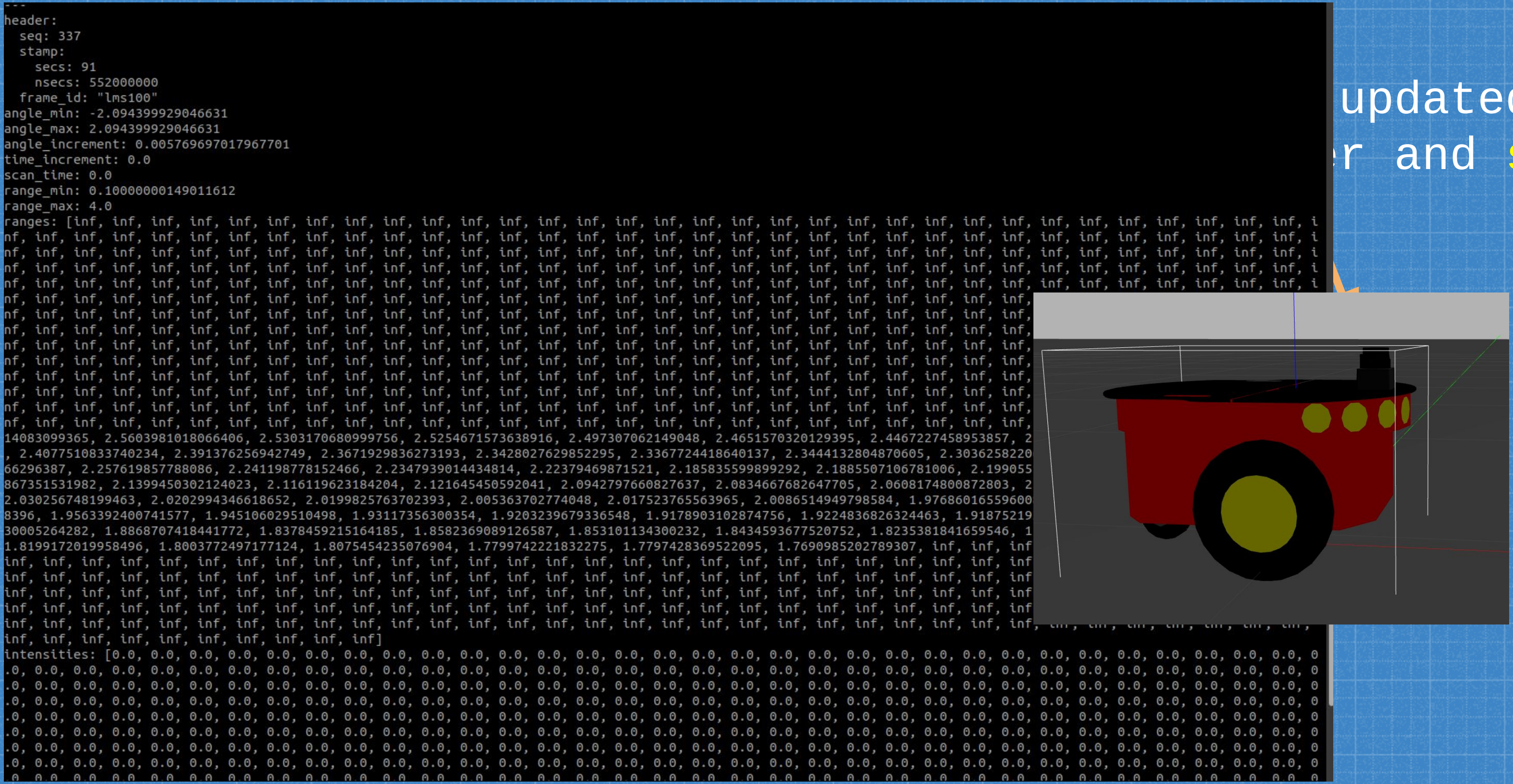
- We managed to control the robot using a script and read pose msgs.

Setbacks:

- Some of the tutorials we found are not updated to noetic and Python3 (Major issues of incompatibility).

- We could not use the laser with this package.

pioneer p3dx model by mario-serna



updated

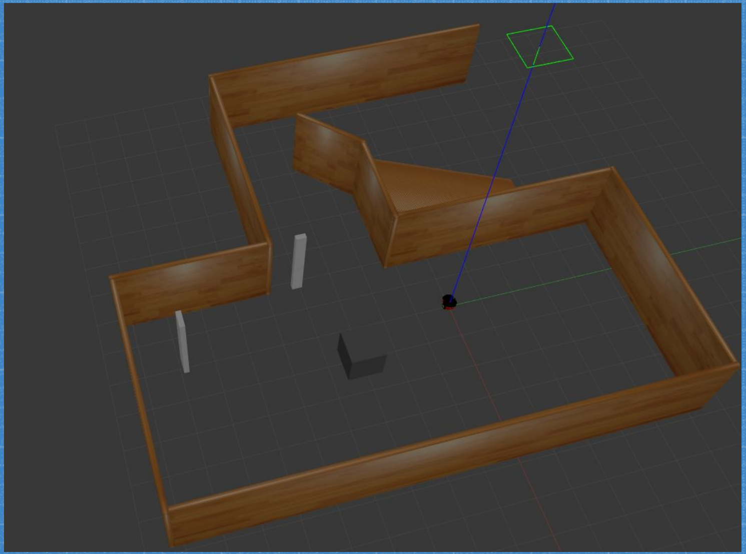
r and sonar.

First Scripts and launch files

Script that publishes a circular motion to /pioneer/cmd_vel

```
if __name__ == '__main__':

    rospy.init_node("draw_circle")
    rospy.loginfo("Hello from draw circle")

    pub=rospy.Publisher("/pioneer/cmd_vel", Twist,queue_size=10)


    #rospy.logwarn("Thos is a warning")
    #rospy.logerr("This is an error")

    #rospy.sleep(1.0)
    rate = rospy.Rate(1)

    while not rospy.is_shutdown():
        #publish cmd vel
        msg=Twist()
        msg.linear.x = 2.0
        msg.angular.z = 0.5
        pub.publish(msg)
        print(msg)
        rate.sleep()#rate of sleep 2hz



    rospy.loginfo("End of program")
```

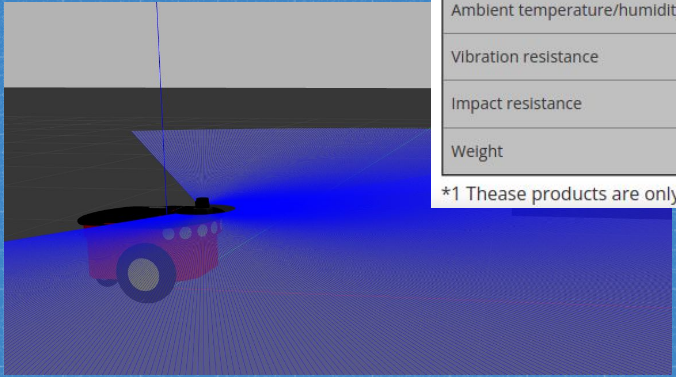
Created a launch file to spawn pioneer on a map



Hokuyo laser

Simulation:

- ~ ∞ update rat
  whole range {-



Model No.	URG-04LX-UG01
Power source	5VDC±5%(USB Bus power)
Light source	Semiconductor laser diode(λ=785nm), Laser safety class 1
Measuring area	20 to 5600mm(white paper with 70mm×70mm), 240°
Accuracy	60 to 1,000mm : ±30mm, 1,000 to 4,095mm : ±3% of measurement
Angular resolution	Step angle : approx. 0.36°(360°/1,024 steps)
Scanning time	100ms/scan
Noise	25dB or less
Interface	USB2.0/1.1[Mini B](Full Speed)
Command System	SCIP Ver.2.0
Ambient illuminance*1	Halogen/mercury lamp: 10,000Lux or less, Florescent: 6000Lux(Max)
Ambient temperature/humidity	-10 to +50 degrees C, 85% or less(No condensation, no icing)
Vibration resistance	10 to 55Hz, double amplitude 1.5mm each 2 hour in X, Y and Z directions
Impact resistance	196m/s2, Each 10 time in X, Y and Z directions
Weight	Approx. 160g

*1 Thease products are only for indoor applications. Strong sunlight may cause error output.

Vibration resistance	10 to 55Hz, double amplitude 1.5mm each 2 hour in X, Y and Z directions
Impact resistance	196m/s2, Each 10 time in X, Y and Z directions
Weight	Approx. 160g

*1 Thease products are only for indoor applications. Strong sunlight may cause error output.

Source: https://www.hokuyo-aut.jp/search/single.php?serial=166#spec

Extended Kalman Filter (EKF)

Simulataneous localization and mapping with the extended Kalman filter

'A very quick guide... with Matlab code!'

Joan Solà

October 5, 2014

Table 1: EKF operations for achieving SLAM

Event	SLAM	EKF
Robot moves	Robot motion	EKF prediction
Sensor detects new landmark	Landmark initialization	State augmentation
Sensor observes known landmark	Map correction	EKF correction
Mapped landmark is corrupted	Landmark deletion	State reduction

Extended Kalman Filter (EKF)

Odometry:

Knowing v and ω

estimate

$$(x_0, y_0, \theta_0) \rightarrow (x, y, \theta)$$

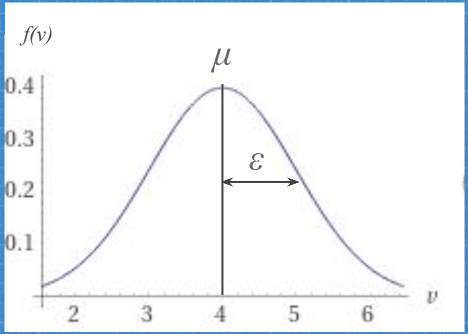
$$f = (x, y, \theta) = \begin{cases} x = x_0 + v \cdot cos\theta_0 \cdot \Delta t \\ y = y_0 + v \cdot sin\theta_0 \cdot \Delta t \\ \theta = \theta_0 + \omega \cdot \Delta t \end{cases}$$

$$\frac{\partial f}{\partial r} = \begin{bmatrix} 1 & 0 & -v\sin\theta\Delta t \\ 0 & 1 & v\cos\theta\Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

with $\varepsilon = n \qquad n = (\varepsilon_v, \varepsilon_\omega)$

$$\frac{\partial f}{\partial n} = \begin{bmatrix} \frac{\partial v}{\partial \varepsilon_v}\cos\theta_0\Delta t & 0 \\ \frac{\partial v}{\partial \varepsilon_v}\sin\theta_0\Delta t & 0 \\ 0 & \frac{\partial \omega}{\partial \varepsilon_\omega}\Delta t \end{bmatrix}$$

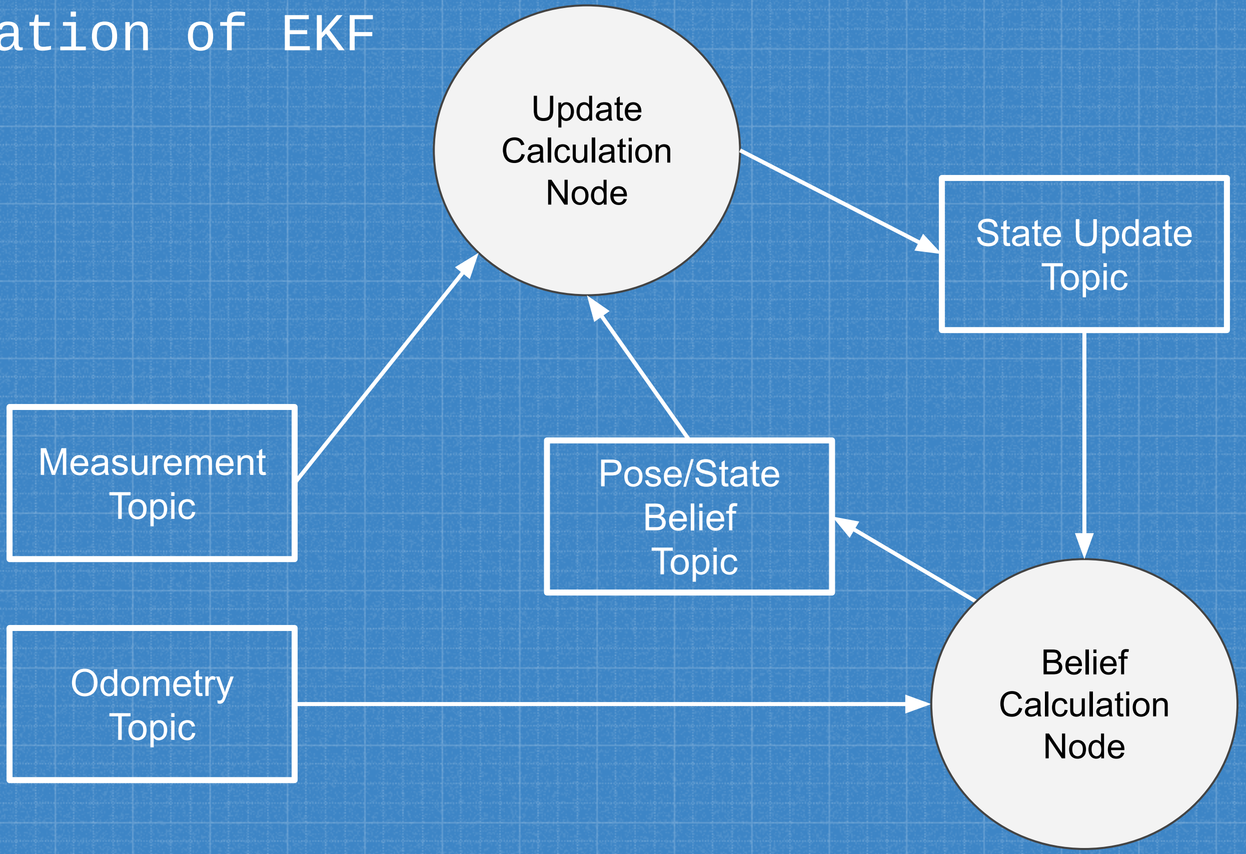
$$f(v) = \frac{1}{\varepsilon_v\sqrt{2\pi}}\exp\{-\frac{(v-\mu)^2}{2\varepsilon_v^2}\}$$



$$\frac{\partial f(v)}{\partial \varepsilon_v} = \frac{1}{\varepsilon_v\sqrt{2\pi}}\exp\{-\frac{(v-\mu)^2}{2\varepsilon_v^2}\}\frac{(v-\mu)^2}{\varepsilon_v^3} - \frac{1}{\varepsilon_v^2\sqrt{2\pi}}\exp\{-\frac{(v-\mu)^2}{2\varepsilon_v^2}\}$$

$$\Longleftrightarrow \frac{\partial f(v)}{\partial \varepsilon_v} = \frac{1}{\varepsilon_v^2\sqrt{2\pi}}\exp\{-\frac{(v-\mu)^2}{2\varepsilon_v^2}\}[\frac{(v-\mu)^2}{\varepsilon_v^2} - 1]$$

Implementation of EKF



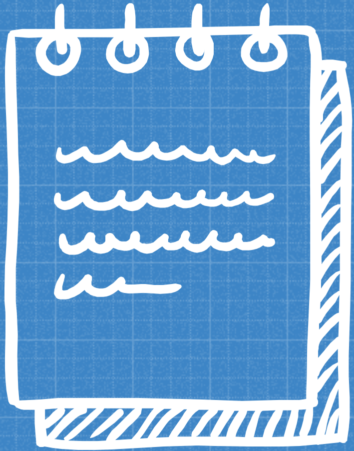
Implementation of the odometry belief

```python
if __name__ == '__main__':
    global t_final, positionx, positiony, theta

    t_final=0
    positionx=0
    positiony=0
    theta=0

    rospy.init_node("Belief_calculation")
    rospy.loginfo("Hello from belief_calculation")

    pub=rospy.Publisher("/pioneer/belief_calculation", PoseWithCovariance,queue_size=10)

    #sub_pose=rospy.Subscriber("/pioneer/pose_belief", PoseWithCovariance)

    sub_odometry=rospy.Subscriber("/pioneer/odom", Odometry, callback = belief_calculati

    rospy.spin()#keeps the node alive
```

```python
def belief_calculation_callback(msg_toreceive: Odometry):
    global t_final, positionx, positiony, theta

    if(t_final!=0):

        #rospy.loginfo(msg_toreceive)
        msg_tosend=PoseWithCovariance()
        velocity=msg_toreceive.twist.twist.linear.x
        angular_velocity=msg_toreceive.twist.twist.angular.z

        t_inicial=t_final
        t_final=msg_toreceive.header.stamp
        delta_t=(t_final-t_inicial).to_sec()

        positionx=velocity*delta_t*math.cos(theta)+positionx
        positiony=velocity*delta_t*math.sin(theta)+positiony

        theta=(theta+angular_velocity*delta_t)

        msg_tosend.pose.position.x=positionx
        msg_tosend.pose.position.y=positiony
        msg_tosend.pose.orientation.z=theta

        rospy.loginfo(msg_tosend)
        pub.publish(msg_tosend)
    else:
        t_final=msg_toreceive.header.stamp
        positionx=msg_toreceive.pose.pose.position.x
        positiony=msg_toreceive.pose.pose.position.y
        theta=msg_toreceive.pose.pose.orientation.z
```

Plans for next week:

1. Use the real Pioneer and Hokuyo laser and check connections.
2. Understand how to use the pose updating algorithm with the real model.
3. Record data from simulated lasers and then use it on a new version of a script.
4. On EKF, validate our results testing the pose updating algorithm. Understand how to include the laser readings/landmarks.



Thank you!

ANY QUESTIONS or Suggestions (please)?

eufemiomarques@tecnico.ulisboa.pt

ivan.figueiredo@tecnico.ulisboa.pt

miguel.roldao@tecnico.ulisboa.pt

pedromatoss@tecnico.ulisboa.pt