Robotiks WS17/17

Assignment 5

Name	MatrNr	Mail
Rémi Toudic	4318284	remitoudic@gmail.com
Sven Heinrichsen	4780388	s.heinrichsen@fu-berlin.de
Alexander Hinze-Huettl	4578322	hinze.alex@gmail.com

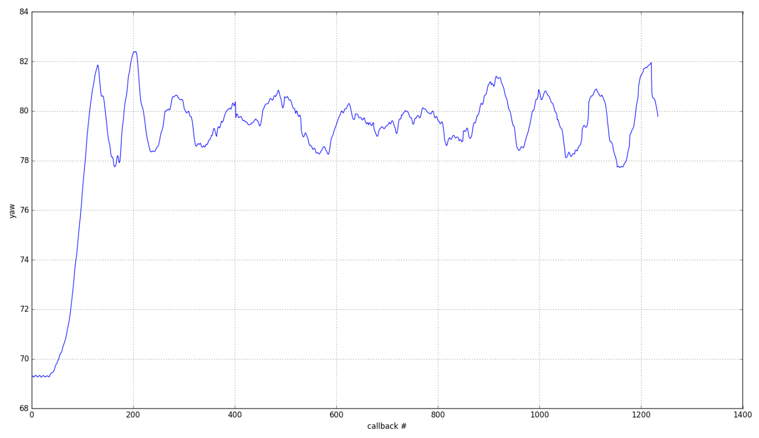
Control a car on a straight lane using a heading sensor

We subscribed to the topic `/model_car/yaw`
We calculated `u` wit the given formular:

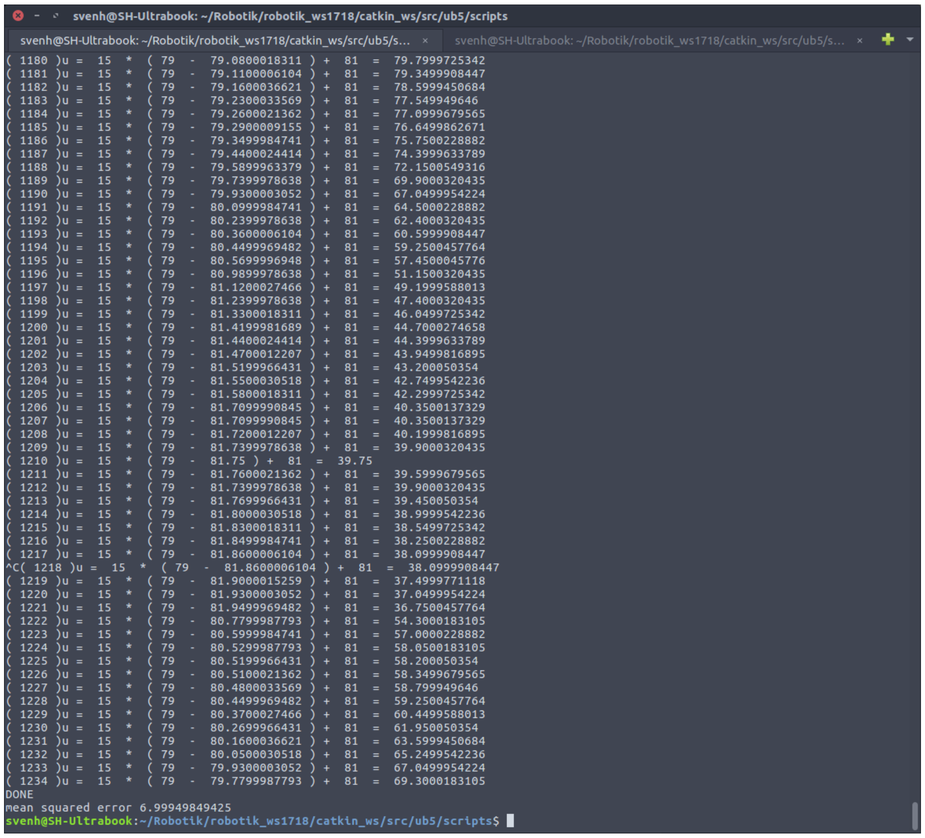
`u = KP *(desired_yaw - current_yaw) + CALIBRATED_ZERO_ANGLE`

We switched between a real car and the gazebo simulator. For the real car we used higher weight values because of the limited space in the lab.

This is our plot of the current yaw against the callbacks for the real car with `KP = 15`:



After 1234 callbacks we had a mean suared error of `7`. The car starts with a yaw of 70° and oscillates around 79°:



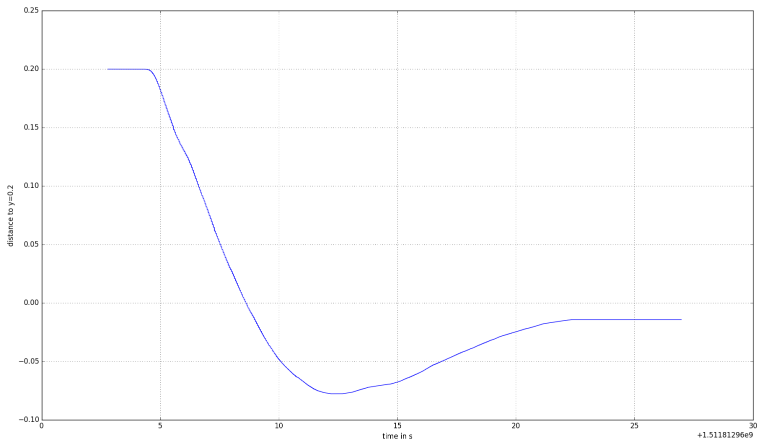
```
svenh@SH-Ultrabook: ~/Robotik/robotik_ws1718/catkin_ws/src/ub5/scripts
svenh@SH-Ultrabook: ~/Robotik/robotik_ws1718/catkin_ws/src/ub5/s...   svenh@SH-Ultrabook: ~/Robotik/robotik_ws1718/catkin_ws/src/ub5/s...
( 1180 )u =  15  *  ( 79  -  79.0800018311 ) +  81  =  79.7999725342
( 1181 )u =  15  *  ( 79  -  79.1100006104 ) +  81  =  79.3499908447
( 1182 )u =  15  *  ( 79  -  79.1600036621 ) +  81  =  78.5999450684
( 1183 )u =  15  *  ( 79  -  79.2300033569 ) +  81  =  77.549949646
( 1184 )u =  15  *  ( 79  -  79.2600021362 ) +  81  =  77.0999679565
( 1185 )u =  15  *  ( 79  -  79.2900009155 ) +  81  =  76.6499862671
( 1186 )u =  15  *  ( 79  -  79.3499984741 ) +  81  =  75.7500228882
( 1187 )u =  15  *  ( 79  -  79.4400024414 ) +  81  =  74.3999633789
( 1188 )u =  15  *  ( 79  -  79.5899963379 ) +  81  =  72.1500549316
( 1189 )u =  15  *  ( 79  -  79.7399978638 ) +  81  =  69.9000320435
( 1190 )u =  15  *  ( 79  -  79.9300003052 ) +  81  =  67.0499954224
( 1191 )u =  15  *  ( 79  -  80.0999984741 ) +  81  =  64.5000228882
( 1192 )u =  15  *  ( 79  -  80.2399978638 ) +  81  =  62.4000320435
( 1193 )u =  15  *  ( 79  -  80.3600006104 ) +  81  =  60.5999908447
( 1194 )u =  15  *  ( 79  -  80.4499969482 ) +  81  =  59.2500457764
( 1195 )u =  15  *  ( 79  -  80.5699996948 ) +  81  =  57.4500045776
( 1196 )u =  15  *  ( 79  -  80.9899978638 ) +  81  =  51.1500320435
( 1197 )u =  15  *  ( 79  -  81.1200027466 ) +  81  =  49.1999588013
( 1198 )u =  15  *  ( 79  -  81.2399978638 ) +  81  =  47.4000320435
( 1199 )u =  15  *  ( 79  -  81.3300018311 ) +  81  =  46.0499725342
( 1200 )u =  15  *  ( 79  -  81.4199981689 ) +  81  =  44.7000274658
( 1201 )u =  15  *  ( 79  -  81.4400024414 ) +  81  =  44.3999633789
( 1202 )u =  15  *  ( 79  -  81.4700012207 ) +  81  =  43.9499816895
( 1203 )u =  15  *  ( 79  -  81.5199966431 ) +  81  =  43.200050354
( 1204 )u =  15  *  ( 79  -  81.5500030518 ) +  81  =  42.7499542236
( 1205 )u =  15  *  ( 79  -  81.5800018311 ) +  81  =  42.2999725342
( 1206 )u =  15  *  ( 79  -  81.7099990845 ) +  81  =  40.3500137329
( 1207 )u =  15  *  ( 79  -  81.7099990845 ) +  81  =  40.3500137329
( 1208 )u =  15  *  ( 79  -  81.7200012207 ) +  81  =  40.1999816895
( 1209 )u =  15  *  ( 79  -  81.7399978638 ) +  81  =  39.9000320435
( 1210 )u =  15  *  ( 79  -  81.75 ) +  81  =  39.75
( 1211 )u =  15  *  ( 79  -  81.7600021362 ) +  81  =  39.5999679565
( 1212 )u =  15  *  ( 79  -  81.7399978638 ) +  81  =  39.9000320435
( 1213 )u =  15  *  ( 79  -  81.7699966431 ) +  81  =  39.450050354
( 1214 )u =  15  *  ( 79  -  81.8000030518 ) +  81  =  38.9999542236
( 1215 )u =  15  *  ( 79  -  81.8300018311 ) +  81  =  38.5499725342
( 1216 )u =  15  *  ( 79  -  81.8499984741 ) +  81  =  38.2500228882
( 1217 )u =  15  *  ( 79  -  81.8600006104 ) +  81  =  38.0999908447
^C( 1218 )u =  15  *  ( 79  -  81.8600006104 ) +  81  =  38.0999908447
( 1219 )u =  15  *  ( 79  -  81.9000015259 ) +  81  =  37.4999771118
( 1220 )u =  15  *  ( 79  -  81.9300003052 ) +  81  =  37.0499954224
( 1221 )u =  15  *  ( 79  -  81.9499969482 ) +  81  =  36.7500457764
( 1222 )u =  15  *  ( 79  -  80.7799987793 ) +  81  =  54.3000183105
( 1223 )u =  15  *  ( 79  -  80.5999984741 ) +  81  =  57.0000228882
( 1224 )u =  15  *  ( 79  -  80.5299987793 ) +  81  =  58.0500183105
( 1225 )u =  15  *  ( 79  -  80.5199966431 ) +  81  =  58.200050354
( 1226 )u =  15  *  ( 79  -  80.5100021362 ) +  81  =  58.3499679565
( 1227 )u =  15  *  ( 79  -  80.4800033569 ) +  81  =  58.799949646
( 1228 )u =  15  *  ( 79  -  80.4499969482 ) +  81  =  59.2500457764
( 1229 )u =  15  *  ( 79  -  80.3700027466 ) +  81  =  60.4499588013
( 1230 )u =  15  *  ( 79  -  80.2699966431 ) +  81  =  61.950050354
( 1231 )u =  15  *  ( 79  -  80.1600036621 ) +  81  =  63.5999450684
( 1232 )u =  15  *  ( 79  -  80.0500030518 ) +  81  =  65.2499542236
( 1233 )u =  15  *  ( 79  -  79.9300003052 ) +  81  =  67.0499954224
( 1234 )u =  15  *  ( 79  -  79.7799987793 ) +  81  =  69.3000183105
DONE
mean squared error 6.99949849425
svenh@SH-Ultrabook:~/Robotik/robotik_ws1718/catkin_ws/src/ub5/scripts$
```

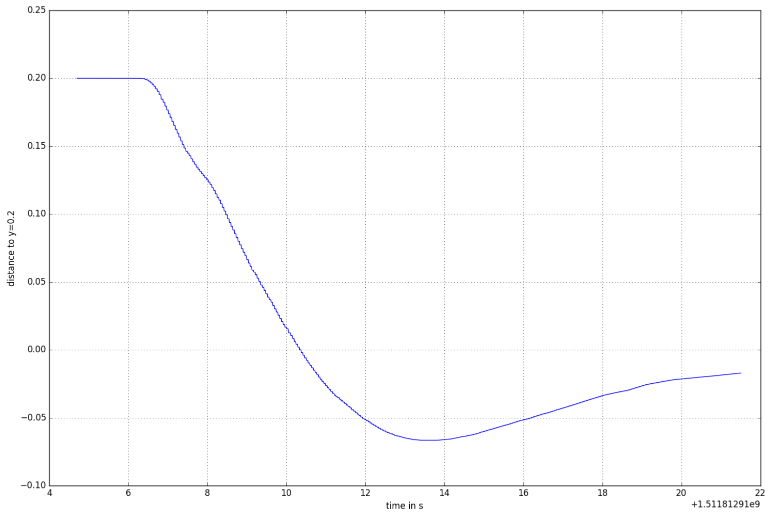
Control a car on a trajectory via odometry

For this task, we subscribed the `/odom` topic. We also recorded the timestamp in `t` and the difference between current y and desired y in `odom_arr` to calculate the derivative:

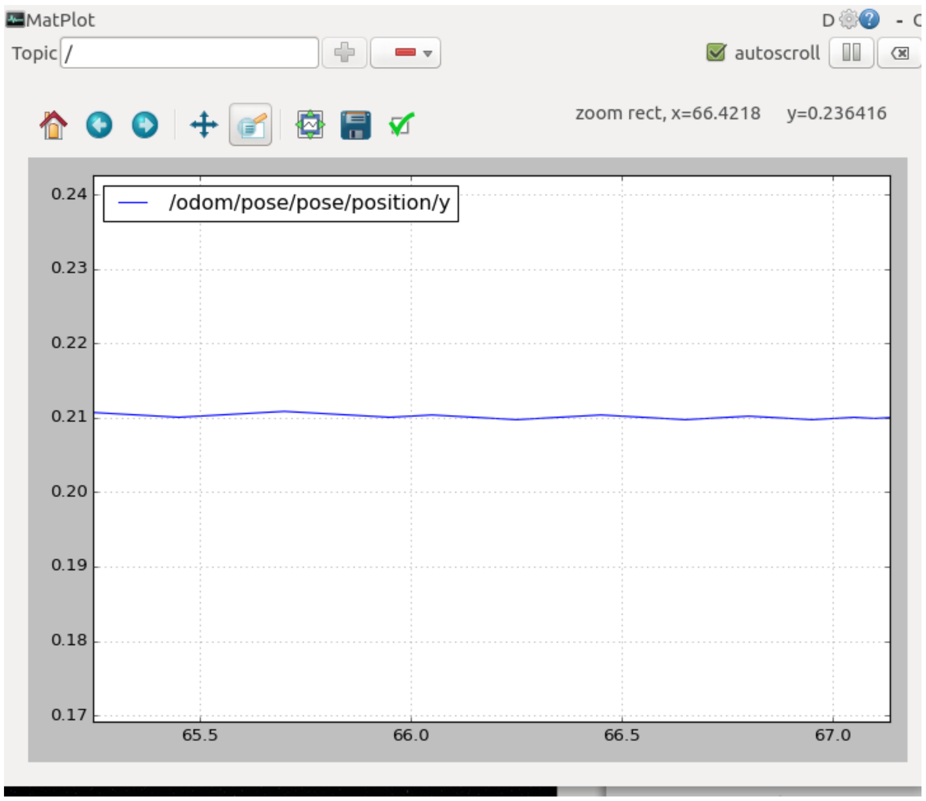
```
def do_PDC(current_y, desired_y):
    derivative = 0
    if len(odomy_arr) > 2:
        derivative = (odomy_arr[-1]-odomy_arr[-2]) / (t[-1]-t[-2])

    u = -KP *(desired_y - current_y) - KD * (derivative) + CALIBRATED_ZERO_ANGLE
    pubSteering(u)
```

For the real cat we had to use high weight values `KP = 300, KD = 100` to see acceptable results over a few meters.
Here are two plots we generated with the real car. They show the distance between `0.2 - current_y` against the time:





finally a plot of the same programm using the gazebo:



Output:

```
...
u= 2.9 *( 0.2 - 0.201403176746 )  + 2.6 * - 0.0153677153998 = 80.9558835035
u= 2.9 *( 0.2 - 0.201406393693 )  + 2.6 * - 0.0157934304967 = 80.9528238898
u= 2.9 *( 0.2 - 0.20140961064 )  + 2.6 * - 0.0153677153998 = 80.9587496072
u= 2.9 *( 0.2 - 0.201412827587 )  + 2.6 * - 0.0168380043108 = 80.9415971952
u= 2.9 *( 0.2 - 0.201416044534 )  + 2.6 * - 0.000204371124299 = 80.9957149158
u= 2.9 *( 0.2 - 0.201419261482 )  + 2.6 * - 0.000204653206481 = 80.9436018456
u= 2.9 *( 0.2 - 0.201422478429 )  + 2.6 * - 0.000204927768309 = 80.9037975928
u= 2.9 *( 0.2 - 0.201425695376 )  + 2.6 * - 0.0291632293682 = 80.8914564932
u= 2.9 *( 0.2 - 0.201428912323 )  + 2.6 * - 0.0389216945799 = 80.8872449446
u= 2.9 *( 0.2 - 0.20143212927 )  + 2.6 * - 0.0413891230706 = 80.8858737194
u= 2.9 *( 0.2 - 0.201435346217 )  + 2.6 * - 0.0400381427924 = 80.9007658951
```

Closing the loop: Control a car using the lidar sensor to keep its distance to a wall

We subscribed to the `/scan` topic and wrote a simple mapping function to get the distance for 80° and 100° for `dr2,dl2`.

From there values and the given constants we calculated `do2`:

```
alpha = alpha * math.pi / 180.0 #deg to rad
t = math.sqrt(dr2**2 + dl2**2 - 2 * dl2 * dr2 * math.cos(alpha))
phi2 = math.asin(dr2* math.sin(alpha) / t)
do2 = - math.sin(phi2) * dl2
```

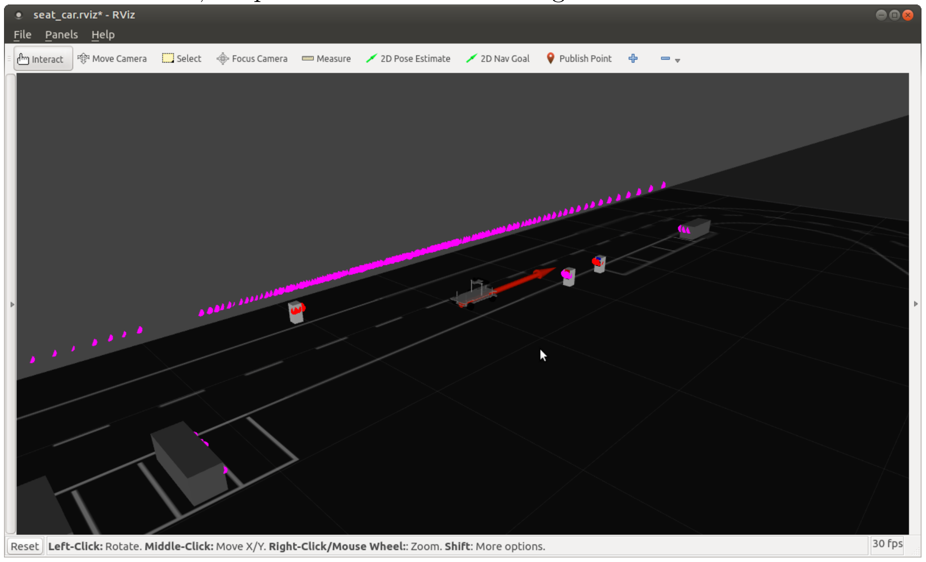
With these three values we tried to calculate the angled `theta` and `theta_stern`:

```
thetal2 = math.asin(do2 / dl2)
theta = thetal2 - alpha
cy = do2 + math.sin(theta) * s
thetaStar = math.atan2(p - cy, l) * 180 / math.pi
theta = theta * 180 / math.pi - 90
```

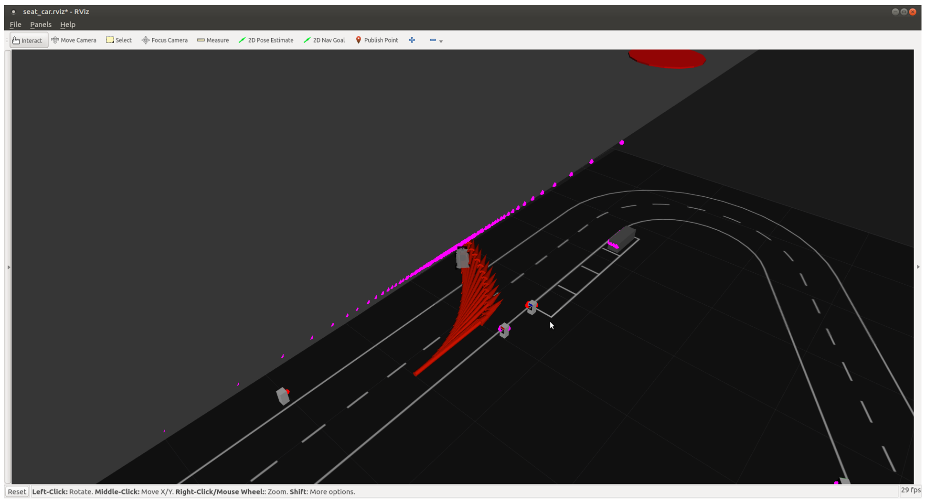
We got stable values for our `dl2,dr2,do2` but crazy values for `theta,theta_stern`. We tried to calculate the PD controller anyway:

```
deltaHeading = thetaStar - theta
derivative = 0
if len(heading_arr) > 1:
  derivative = (heading_arr[-1] - heading_arr[-2]) / (t[-1] - t[-2])
u = - KP * deltaHeading - KD * derivative + CALIBRATED_ZERO_ANGLE
```

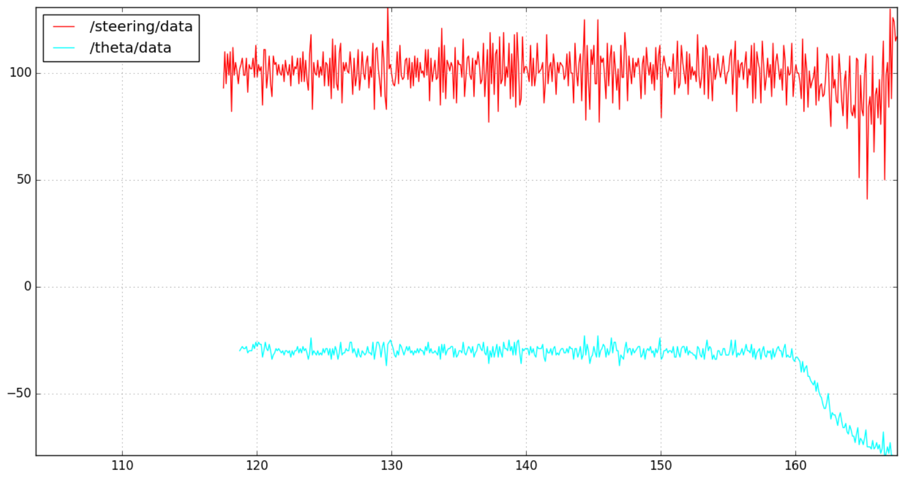
To test this code, we placed an obstecla in the gazebo world:



We get valid values for `dl2, dr2, do2` but the calculation for the steering fails:



Here is the plot of the steering angle and the theta angle. You can see the car starting af 160:


/steering/data
/theta/data