DRAFT - 5. Assignment, Introduction to Robotics WS17/18 - Ver. 0.99

Prof. Daniel Göhring, Martin Dreher, Jakob Krause
Institut für Informatik, Freie Universität Berlin
Submission: online until Tuesday, 28 Nov 2017, 11:55 a.m.

Please summarize your results (images and descriptions) in a pdf-document and name it, e.g., "RO-05-<surnames of the students - group name>.pdf".
Submit your python code, your video
Only one member of the group must submit the necessary files.
Do not copy solutions to other groups.
Every group must contain two people, unless granted differently.
Only submissions via KVV will be accepted.

Since we have more lab time, select free time slots at the new doodle poll:
https://doodle.com/poll/6cx3drrnmfq85cb4
We have 5 cars, so please not more than 5 groups per time slot.

1. Control a car on a straight lane using a heading sensor (3 Points):
Try to calibrate the steering angle of the car at first, i.e., the angle at which the car goes straight.

a. Control a car for three meters on a straight lane with 300 rpm. This means you should control the heading angle to stay constant while driving.
Use a simple proportional controller, which means that if there is an error between the desired heading angle and the current heading angle you apply a steering angle with respect to this error multiplied by a constant:

u = Kp * (desiredHeading - currentHeading) + calibratedZeroSteeringAngle .

Which constant did you chose?
b. Plot the heading angle over time for the the Kp constant that you chose. Calculate the average squared difference between desired and current heading.

Hint: You can subscribe to /model_car/yaw (in degree) or /odom (quaternion) to get the heading angle of the car.

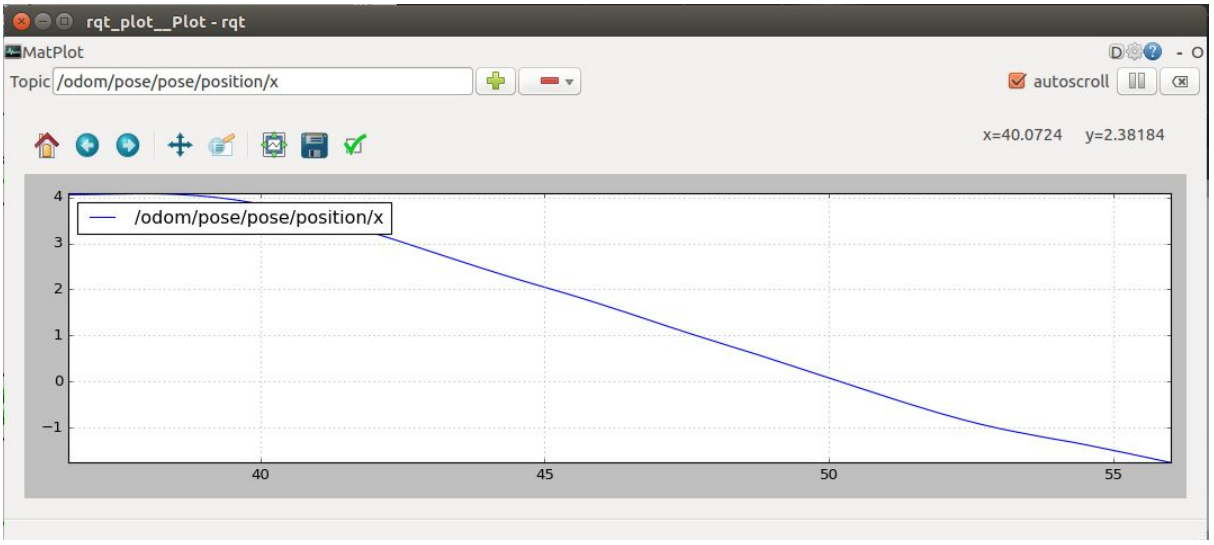
Paste your plot and the average squared difference into your Pdf. Submit your python node, too.

2. Control a car on a trajectory via odometry (4 Points):

a. Assume that your car starts at (0,0). Write a PD controller and tune its weights for steering to follow a straight line at y = 0.2. the car should follow the line for at least 2 meters. Plot /odom/pose/pose/position/y over time.

Hint: Subscribe to /odom topic to find our what your current y-coordinate is.

As an example, your plot for your y-coordinate over time could look like this.



Submit your source code of your python node and paste your plot into your Pdf file.

3. Closing the loop: Control a car using the lidar sensor to keep its distance to a wall (6 Points):

Place your car approx 0.5 meters and not perfectly straight next to a wall, like in Figure 2.

Take two scans of your lidar to the right, e.g., at 240 and 300 degrees, as shown in Figure 2. Now calculate the distance of the vehicle and the angle, as in Assignment 04, also as shown in Figure 1. Remember: When the car is parallel to the wall, this corresponds to an angle \theta of 0 .

Now use a PD controller which gets the desired angle of your vehicle \theta* with respect to a lookahead point and the current angle \theta as an input. The controller must then output a steer command. The car shall follow a line, parallel to the wall and with 0.4 meters distance to the wall. Feel free to try out different distances if this work better for you. See Figure 2 for an example and take a look at the controller paper (lecture-05-controller-paper.pdf, Figure 8) in the resources folder at KVV.

Your car should drive at least for two meters but no more than 5 meters. Use a starting position not exactly on the desired trajectory, i.e., at least 10 cm away from it and with an angle theta not perfectly 0, i.e., at least 10 degrees away, so that one can see how well your controller converges.

Take a video (i.e. with your smartphone) of not more than 20 seconds (and not larger than 5 MB, in ogv or mp4 format, no other formats are allowed) showing how your car is moving from that starting position and traveling between 2 or 5 meters.

Plot your measured distance, measured angle \theta, and steer output over time (can be 3 plots with the same time reference or one plot showing them altogether).

Submit the video, your python code and put the plots in your Pdf. Provide a link to your repository where your code can be found, too.

Hint: Remember, PD control for this case:

*deltaHeading_t = theta*_t - theta_t*

*u = Kp * (deltaHeading_t) + Kd * ((deltaHeading_t - deltaHeading_(t-1)) / controlFrequency + calibratedZeroSteeringAngle .*

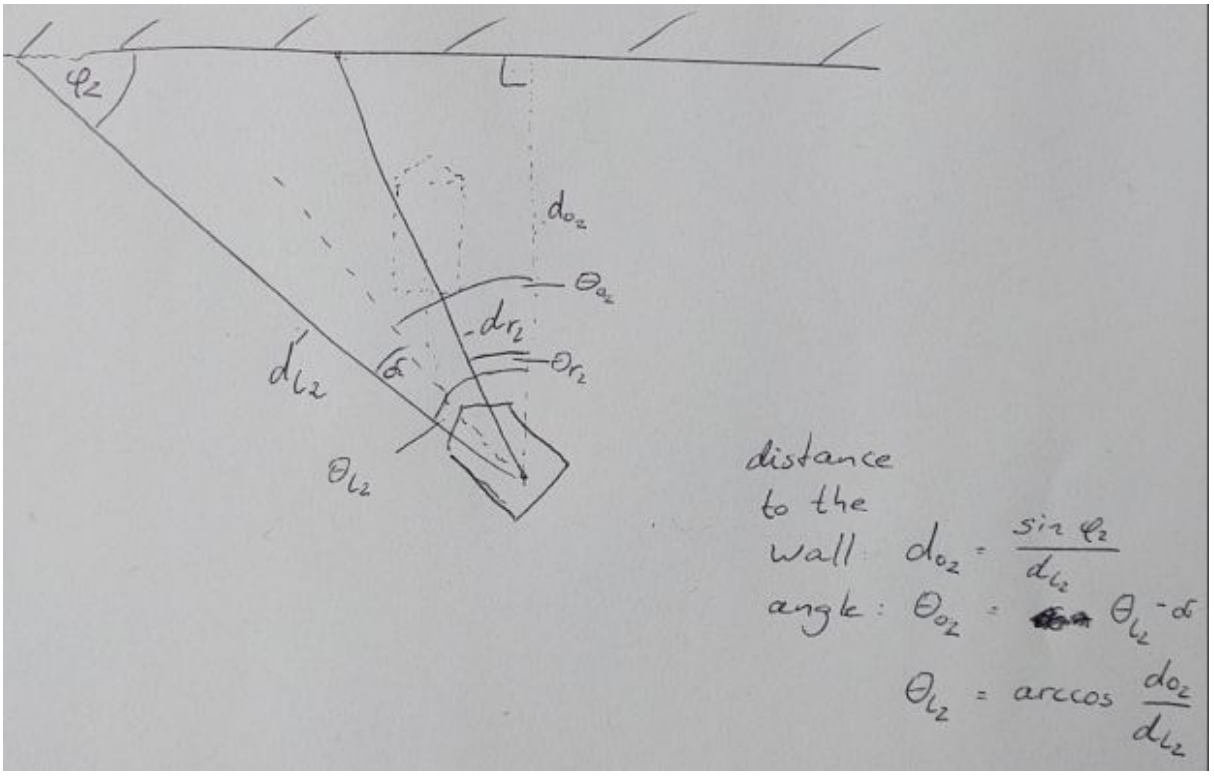

φ_2
d_{o_2}
Θ_{o_2}
dr_2
δ
d_{l_2}
Θ_{l_2}
distance to the wall $d_{o_2} = \frac{\sin \varphi_2}{d_{l_2}}$
angle: $\Theta_{o_2} = \Theta_{l_2} - \delta$
$\Theta_{l_2} = \arccos \frac{d_{o_2}}{d_{l_2}}$

Figure 1

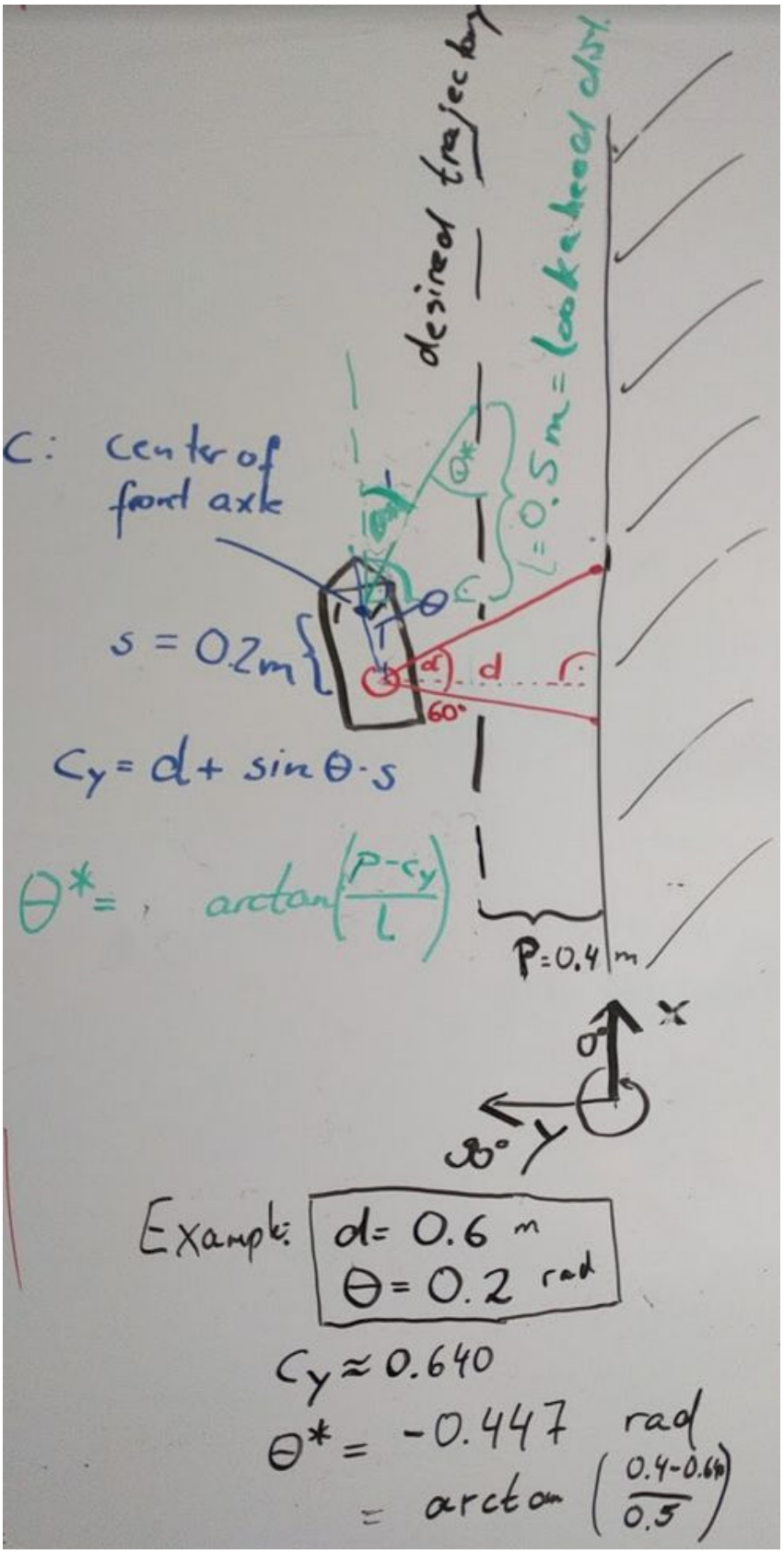

desired trajectory
C: center of front axle
L = 0.5 m = lookahead dist.
s = 0.2m
d
60°
$C_y = d + \sin\theta \cdot s$
$\theta^* = \arctan\left(\frac{P - C_y}{L}\right)$
P = 0.4 m
x
y
Example:
d = 0.6 m
θ = 0.2 rad
$C_y \approx 0.640$
$\theta^* = -0.447$ rad
$= \arctan\left(\frac{0.4 - 0.64}{0.5}\right)$

Figure 2