8. Assignment – Velocity Control and lateral control

by Pascal Müller, Fabian Casares, Friedrich Müller

https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub08/

Exercise 1 – Velocity Control

Unser Ansatz bestand darin das Topic /ticks zu verwenden um die zurückgelegte Distanz pro Zeit zu ermitteln. Messungen ergaben, dass pro (geradeaus) gefahrenen Meter insgesamt ca. 180 ticks zu zählen waren. So war mit 1 Tick pro 5/9cm zu rechnen, bzw mit 36 Ticks pro Radumdrehung.
Dies ermöglichte nach Eingabe einer Zielgeschwindigkeit in Metern pro Sekunde eben genau diese mittels PID-Control anzustreben.
Das Ziel war es einen Subscriber zum Topic /ticks zu bauen, sodass Ticks pro Zeitraum berechnet werden konnten um den momentanen error zu berechnen (angestrebte Geschw. Minus momentane Geschw.) und entsprechend die Geschwindigkeit des Autos mittels P-, I- und D-Gliedern anzupassen.

Hierfür wurde eine Klasse pid_vel angelegt, welche auf dem topic /ticks stets die Ticks zur Ermittlung der gefahrenen Distanz zählt und auf dem topic /manual_control/speed die ermittelten Geschwindigkeitserhöhungen vornimmt. Einziger Instanzierungsparameter ist der Wert der angestrebten Geschwindigkeit in Metern pro Sekunde, welcher in Ticks pro Sekunde umgerechnet wird. Der While loop ist mit einem timer versehen, sodass eine Iteration pro Sekunde geschieht und jeweils die Veränderung des Zählers der Ticks, zur Ermittlung der Geschwindigkeit und Berechnung des Errors, verwendet.

Eine finale Ausführung zur Anpassung der gain-Faktoren und Generierung der Plots steht noch aus.

https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub08/RO-08-Mueller-Mueller-Casares-Lemming%20-%20Ex%201.py

Exercise 2 – Control a car on an oval circuit

Wir haben zwei Varianten implementiert, wovon nur eine Funktioniert hat weil die andere zu inefizientt war.

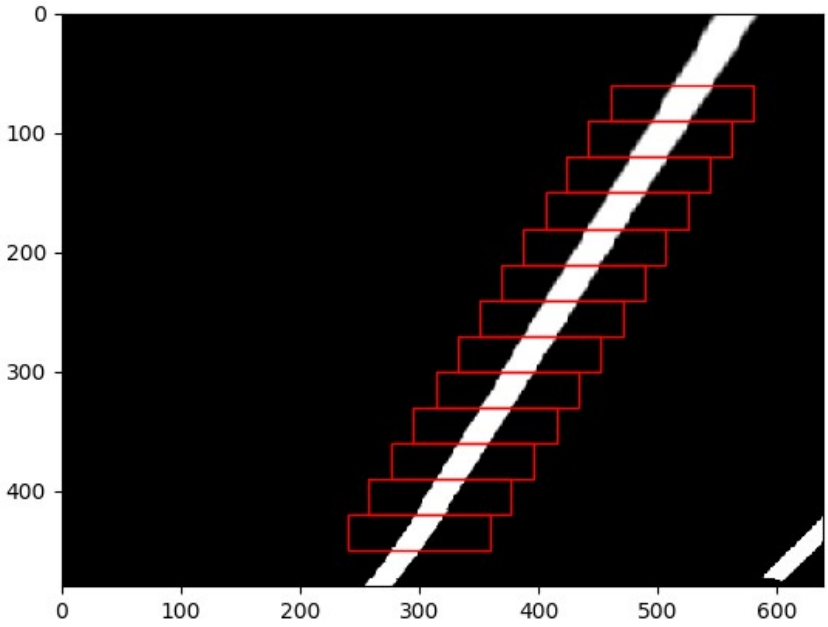
1. Variante
Wir haben den Controller durch den Offset (Abstand Auto zur Linie) trainiert. Und da der Offset bei uns kleiner wird wenn wir auf einer geraden Linie fahren haben wir den durchschnittlichen Ofset über einer Zeit vom speed abgezogen und somit sind wir schneller gefahren auf geraden Strecken als in der Kurve.

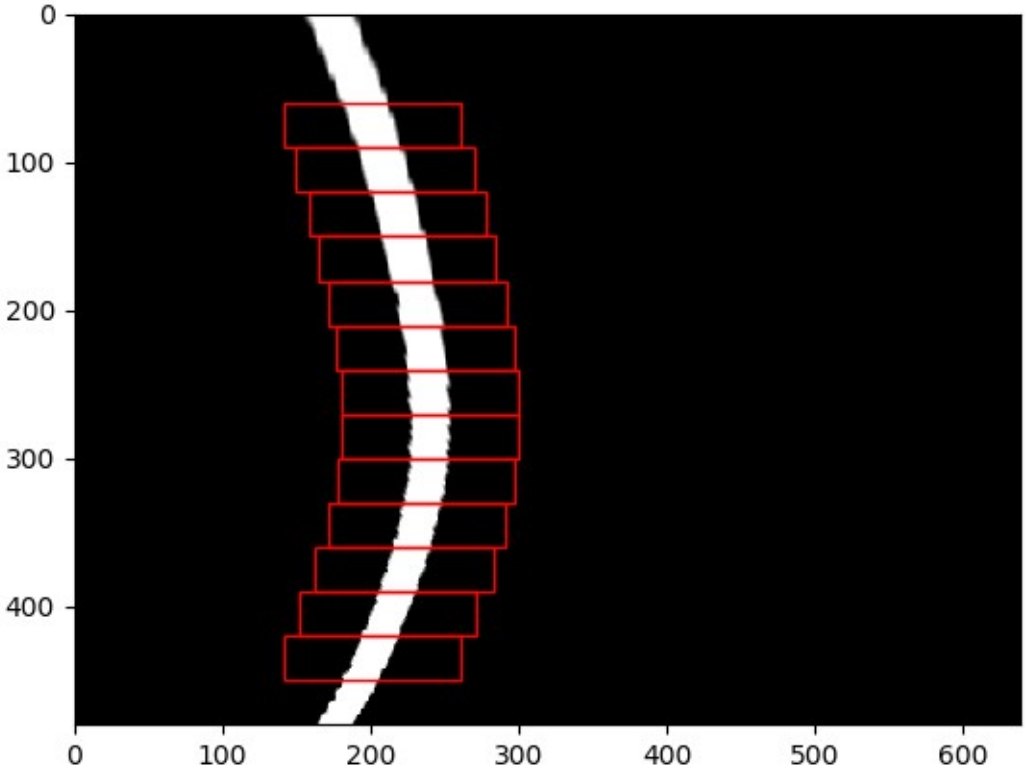
Code:
https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub08/line_detection3.py
https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub08/drive_one_line3.py

2. Variante
Bei der Zweiten Variante haben wir den “Sliding Window Algorithmus” angewendet um gerade Linien von Kurven zu unterscheiden. Als Parameter dazu haben wir den Radius verwendet, dessen Kreis einen gemeinsamen Linienabschnitt mit der erkannten Linie hat. Wenn die Linie eine Gerade ist, ist der darauf projizierte Kreis groß und wenn die Linie gekrümmt ist ist der Radius klein.

Wir haben dann den Radius und den Offset verwendet um den Contrller zu trainieren.
Leider haben wir den Algorithmus aber etwas zu uneffizient implementiert und deshalb, hat es nicht sehr gut funktioniert.
Code:
https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub08/line_detection4.py
https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub08/drive_one_line5.py

Ein Video der Fahrt befindet sich im Repository
https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub08/RO-08-Mueller-Mueller-Casares-Lemming%20-%20Ex%202.mp4




0
100
200
300
400
0
100
200
300
400
500
600