11. Assignment – Obstacle Avoidance

by Pascal Müller, Fabian Casares, Friedrich Müller

https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub11/

Exercise 1 – Calculate Distance to Nearest Obstacle on Lane

1. Transformation der Koordinaten

Die Transformationsmatrix T Sieht wie folgt aus:

```
cos(a) -sin(a) 0  x_cw
sin(a)  cos(a) 0  y_cw
  0       0    1   0
  0       0    0   1
```

, wobei X_cw und Y_cw die Koordinaten des Autos in Respektive zum Welt-Koordinatensystem sind.
Diese wird (in unserem Fall) multipliziert mit einem Vektor v:

```
x_pc
y_pc  = v
 0
 1
```

, wobei x_pc und y_pc die Koordinaten eines Lidar-Punktes in Bezug zum Auto sind. (Die z-Koordinate kann hier vernachlässigt werden.)

2.+3. Hinderniss-Erkennung in der Spur

Hierfür haben wir unseren „Zielpunkt“ des Autos verwendet und dessen Koordinaten (in Respektive zur Welt) transformiert, sodass dieser in Respektive zum Auto vorliegt. Anschließend haben wir stets alle vom Lidar wahrgenommenen Punkte betrachtet. Lagen davon zu irgendeinem Zeitpunkt während der Fahrt eine Treshhold Anzahl von Punkten im Radius von 15cm des angestrebten Punktes in der Trajektorie, wurde dies als Hinderniss erkannt und ein lane-switch initiiert. (Dies setzt die Annahme vorraus, dass während der Fahrt keine Veränderungen der Hindernisse geschieht, also bewegliche Hindernisse ausgeschlossen sind.)

Code:
https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub11/drive_on_line_with_loc.py

Video:
https://github.com/frimuell/catkin_ws_user/blob/master/tasks_wise1819/ub11/Exercise%201.mp4

4. Scannen der zweiten Spur im Falle eines lane-switchs

Wird ein lane-switch initiiert, so wird gleichzeitig die zukünftige lane geprüft. Unsere Idee war zeitgleich mit dem Erkennen eines Hindernisses die zweite Fahrbahn innerhalb des nächsten Meters zu scannen und anzuhalten, sollte dort ebenfalls ein Hinderniss gefunden werden.