Die erste Aufgabe ist uns nicht gelungen und daher konnten wir für die zweite Aufgabe nur eine Idee Implementieren und nicht austesten.

1 Closest Point with Prediction

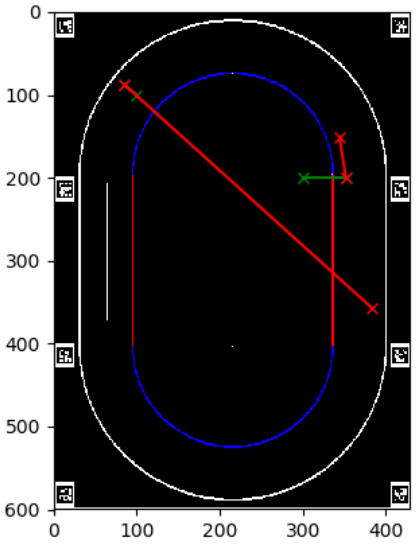

Abbildung 1: Karte mit nächsten Punkten (grün) und errechneten Distanzen (rot)

`https://github.com/bigzed/model_car/blob/version-4.0/texinput/src/closest_distant_point.py`
Das Konzept, dass wir in dieser Aufgabe versucht haben umzusetzen basiert auf der Aufgabenstellung der vorangegangen Woche. Dazu wird zunächst ein closest point zu einem Punkt berechnet. Dieser ist nun zusätzlich abhängig von der laneID, indem die innere Spur eine Vergrößerung des Ovals um 16 cm und die äußere um 48 cm erhält.

Nun wird die Distanz zusätzlich hinzugefügt, für eine Gerade ist dies trivial, sollte jedoch der Bereich zuende sein und es ist noch etwas von der Distanz übrig, dann wird der Rest auf den anschließenden Kreis abgebildet.

Beim Abbilden auf den Kreis hatten wir keinen Erfolg. Unser Vorgehen:

1. Den gefunden Closest Point um den Kreismittelpunkt in Richtung des Koordinatenursprung verschieben
2. Herausfinden des Winkels des Closest Points
3. Errechnen des Winkels der Distanz auf dem Kreis
4. Rotieren des Vektors des Closest Points mittels Drehmatrix um die Summe der beiden Winkel.
5. Abziehen und merken der möglichen Distanz, die den Halbkreis überschreitet.
6. Verschieben des Vektors des Closest Points (bereits rotiert um den Winkel der Distanz) zurück in Richtung des ursprünglichen Kreises.

Zu beobachten ist, dass umso größer der Winkel, umso größer wird die Distanz abgebildet. Dies ist gut zu vergleichen an den beiden Punkten im Bild.
Der Punkt rechts oben ist 0°, während der Punkt links oben zwischen 90° und 180° eine unwahrscheinlich hohe Distanz bekommt. Dies konnten wir uns nicht erklären, da es augenscheinlich nicht mehr von dem Distanzwert, sondern von dem Winkel abhängig wurde.

2 Follow lane with ceiling camera GPS

`https://github.com/bigzed/model_car/blob/version-4.0/catkin_ws/src/assignment10_look_ahead_point_and_gps/src/line_detection.py`
Da wir nur Konzepte implementieren konnten, wird folgend die Grundidee am Code erklärt.

- In Zeile 86 wurde die Fehlerübergabe für die Lenkung auf die GPS-Lokalisierung umgestellt.
- In den Zeilen 245-251 wurde das Quaternion in die Ausrichtung (yaw) des Fahrzeugs transformiert.
- In den Zeilen 252-258 wurde der Fehler aus dem Winkel zwischen Blickrichtung des Fahrzeugs und Winkel des Fahrzeugs zum entfernten Punkt ermittelt.