1 Map description

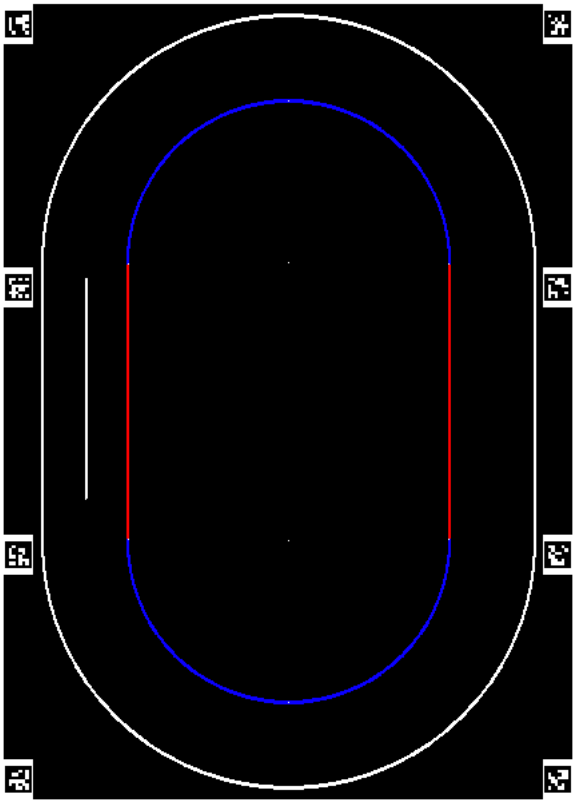

Abbildung 1: Karte mit eingezeichneten Messpunkten.

- Linie links: $P_{1L} = (94,197); P_{2L} = (94,403)$
  $x = 94$
- Linie rechts: $P_{1R} = (336,197); P_{2R} = (336,403)$
  $x = 336$
- Kreis oben: $P_{CO} = (215,196); r = 121$
  $(x - 215)^2 + (y - 196)^2 = 121^2$
- Kreis unten: $P_{CU} = (215,404); r = 121$
  $(x - 215)^2 + (y - 404)^2 = 121^2$

2 Closest point on trajectory

Als erstes haben wir die gegebenen Koordinaten von Metern in Pixel umgerechnet, wobei $1m \equiv 100px \equiv 100cm$.

Nun testen wir ob der angefragte Punkt einer der Mittelpunkte der Kreise ist. Sollte dies der Fall sein geben wir den Mittelpunkt auf dem halben Kreisbogen als Ergebnis zurück.

Als nächsten unterscheiden wir die 4 Quadranten der Karte: oberer Kreis, unterer Kreis, linke Linie, rechte Linie. Abhängig vom Quadranten verwenden folgende Berechnung:

- oberer Kreis:

$$\begin{aligned} V &= P - P_{CO} \\ P_{closest} &= P_{CO} + \frac{V}{|V|} \cdot r \end{aligned}$$

- unterer Kreis:

$$\begin{aligned} V &= P - P_{CU} \\ P_{closest} &= P_{CU} + \frac{V}{|V|} \cdot r \end{aligned}$$

- linke Linie:

$$P_{closest} = (94, P.y)$$

- rechte Linie:

$$P_{closest} = (336, P.y)$$

Hierbei ist $|V|$ die Länge des Vektors. Quellcode: `https://github.com/bigzed/model_car/blob/version-4.0/texinput/src/closest_point_on_trajectory.py`

Ergebnisse in px(cm):

- (0,0): (125.58025288,114.48246309)
- (200,400): (94,400)
- (100,300): (94,300)
- (200,215): (94,215)

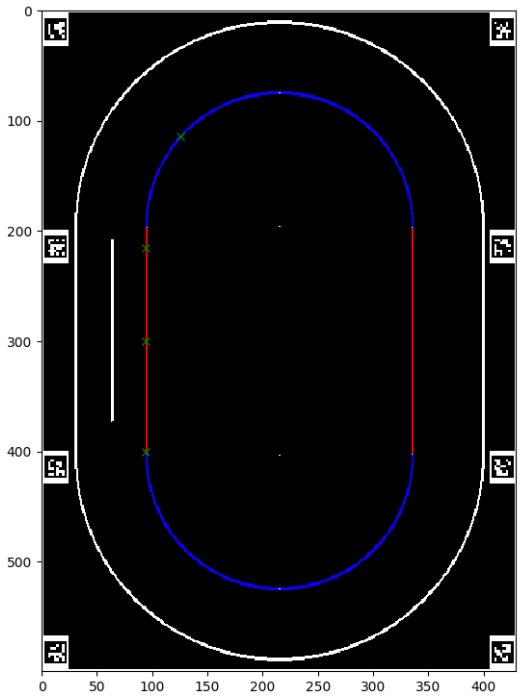

Abbildung 2: Karte mit eingezeichneten nächsten Punkten.

3 Ceiling camera GPS

Quellcode; `https://github.com/bigzed/model_car/blob/version-4.0/catkin_ws/src/assignment9_ceiling_camera_gps/src/line_detection.py`

- mean absolute error: $6.30822263109cm$
- mean squared error: $75.0087521354cm^2$

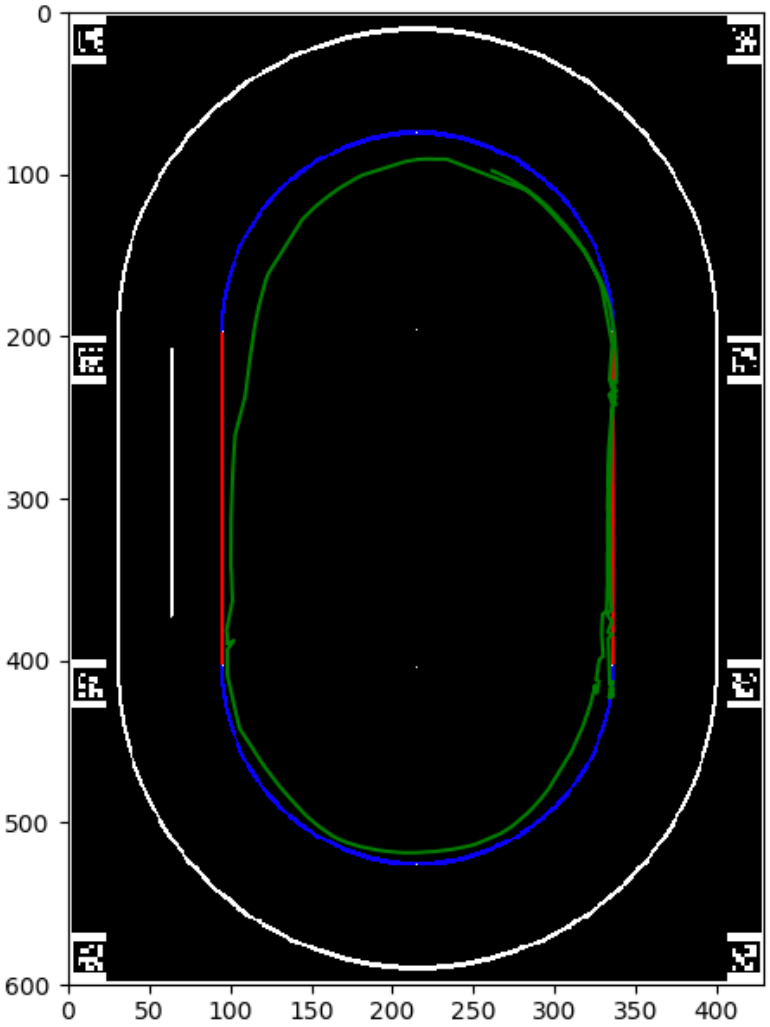

Abbildung 3: Lokalisierungs Daten in Grün