Pi-Car-Turbo
Abschlusspräsentation

Wolfsburg | 20. Dezember 2023

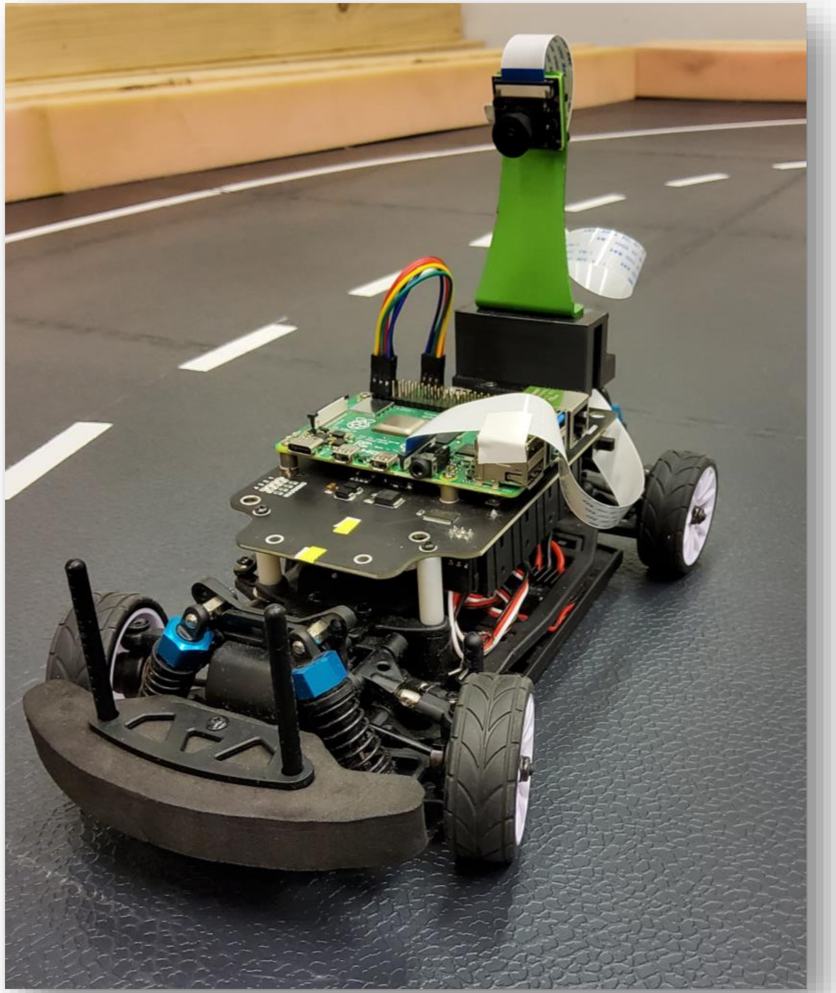
Klassifizierung: Intern

Pi-Car-Turbo
Agenda

1. Projektübersicht
1.1 Motivation
1.2 Unser Ziel

2. Projektverlauf
2.1 Wissensaneignung
2.2 Entwicklung
2.3 Probleme und Lösungen
2.4 Implementierung

3. Aktueller Stand des Projektes
3.1 aktuelle Probleme
3.2 Ergebnis
3.3 Lösungsideen und Ausblicke



1. Projektübersicht

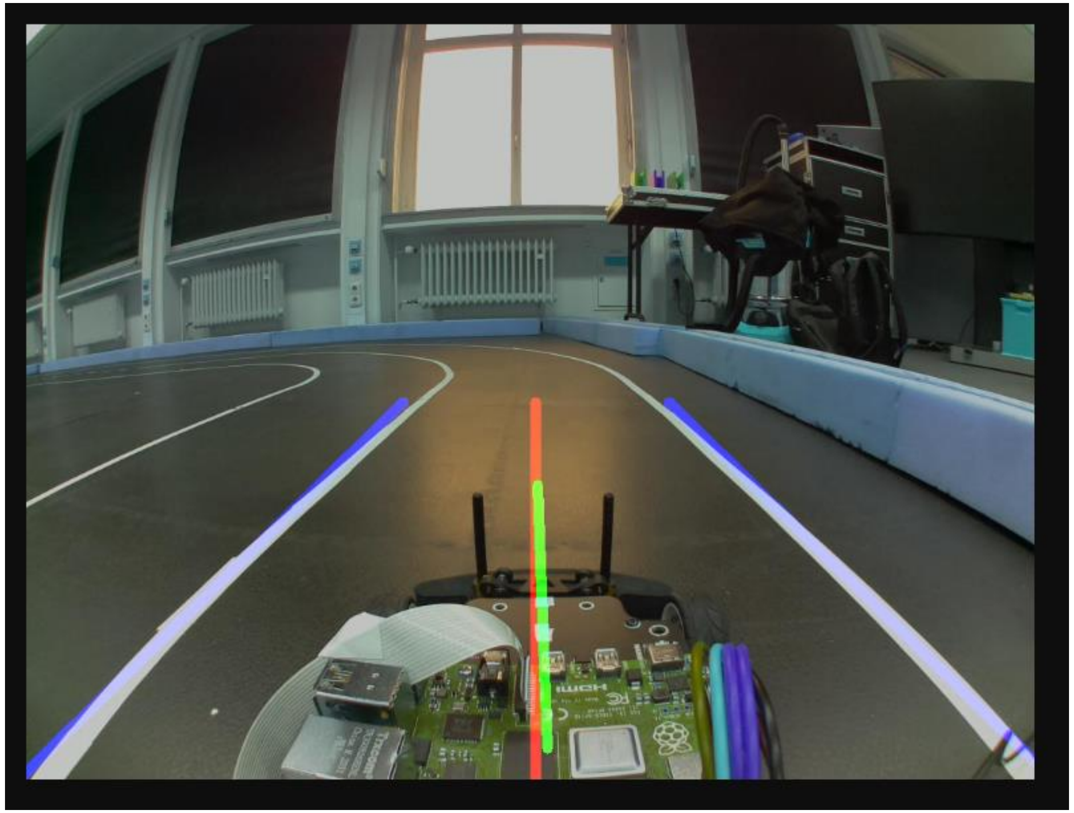
Pi-Car-Turbo
Motivation

- Interesse am autonomen Fahren und dessen Umsetzung
- beweisen das eine Maschine es schafft die Strecke schneller zu bewältigen als ein Mensch
- Interesse an der Entwicklung und Entstehung eines größeren Projektes

Pi-Car-Turbo
Ziele

Ziele der Projektphase

- Auto mittig zwischen zwei Spuren führen
- Kurven erkennen und befahren
- Geschwindigkeit auf Geraden erhöhen



Zukünftige Ziele

- Strecke merken und Geschwindigkeit in den Kurven erhöhen
- Rennlinie befahren

2. Projektverlauf

Pi-Car-Turbo
Wissensaneignung

- Python Kurs (grundlegende Python Kenntnisse)
- YouTube Tutorials als Basis (Sliding Windows, Hough Lines)
- Git Hub (ähnliche Projekte)
- OpenCV Library
- Erfahrung durch Testen und Debugging

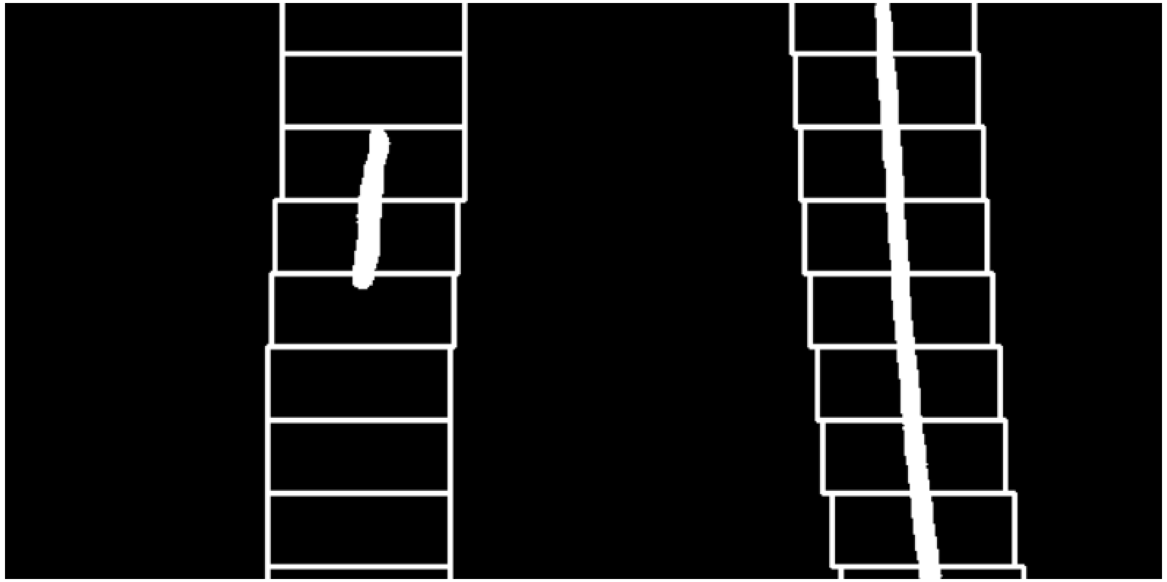
Pi-Car-Turbo
Entwicklung

1. Einrichtung des Autos
2. Linienerkennung (Sliding Windows -> Hough Lines)
3. Ausrichtung des Autos mittig zwischen den Linien
4. Lenkverhalten des Fahrzeuges in Relation zu Linien

Pi-Car-Turbo
Probleme

- Sliding Windows nicht optimal
- Kamerawinkel nicht ausreichend
- Geschwindigkeit abhängig von der Batterieladung (variiert)
- Ausrichtung des Fahrzeuges zur Strecke ist mangelhaft
- Minimalgeschwindigkeit des Autos zu hoch

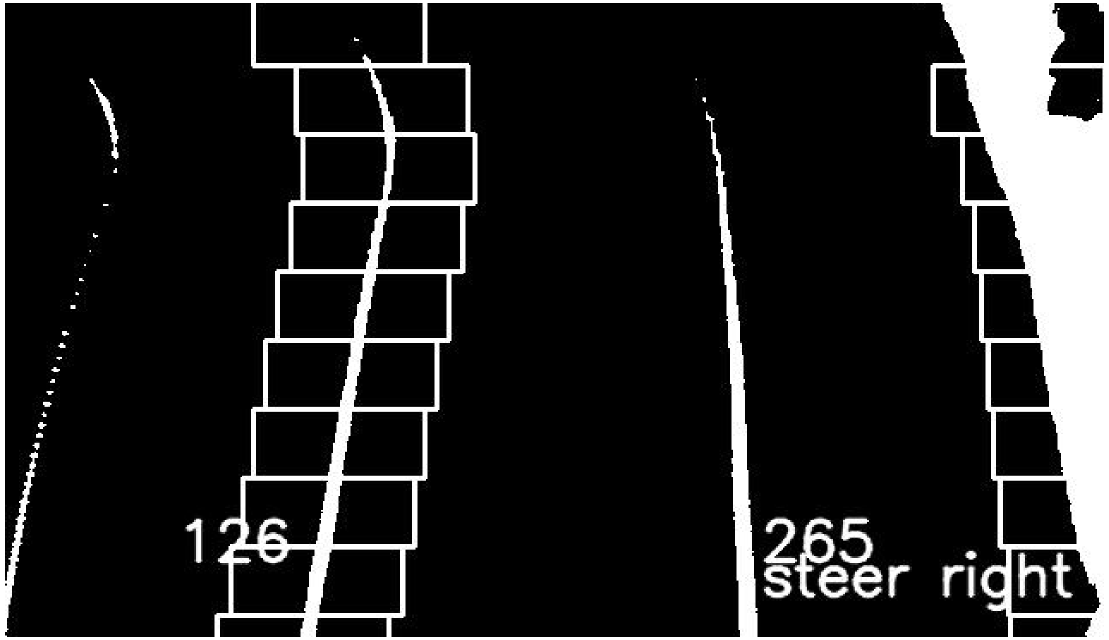
Pi-Car-Turbo
Sliding Windows

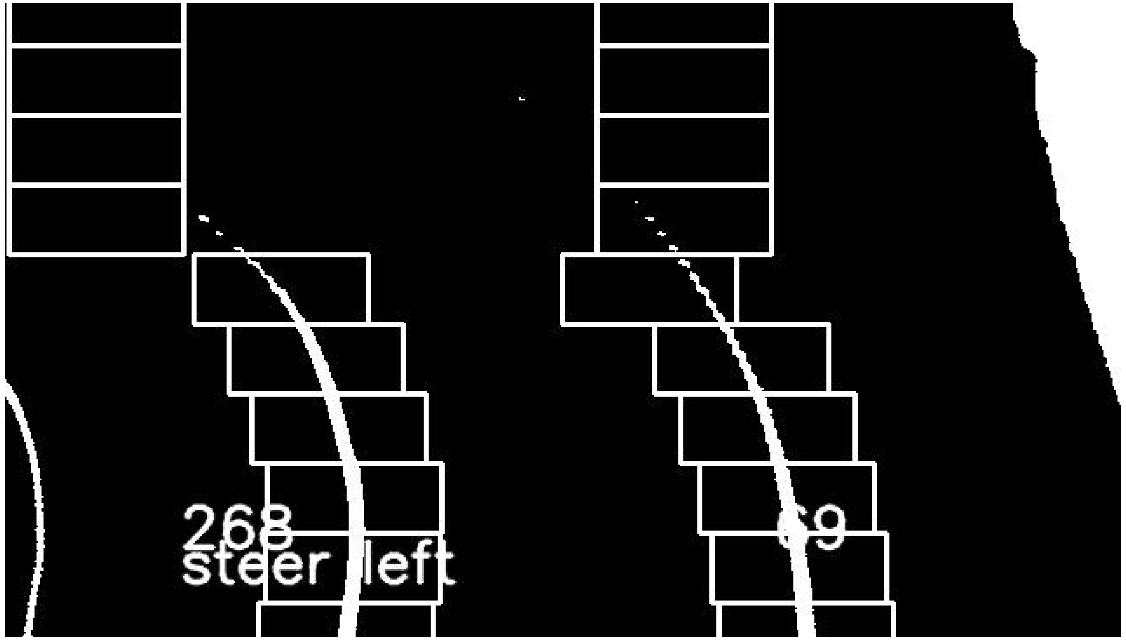


- Birds Eye View
- Zoom auf die Spur
- Perspektivische Verzerrung

- Erkennung durch Intensivität
- 12 Kästchen pro Frame
- Kästchen geben xy Koordinate an

Pi-Car-Turbo
Sliding Windows

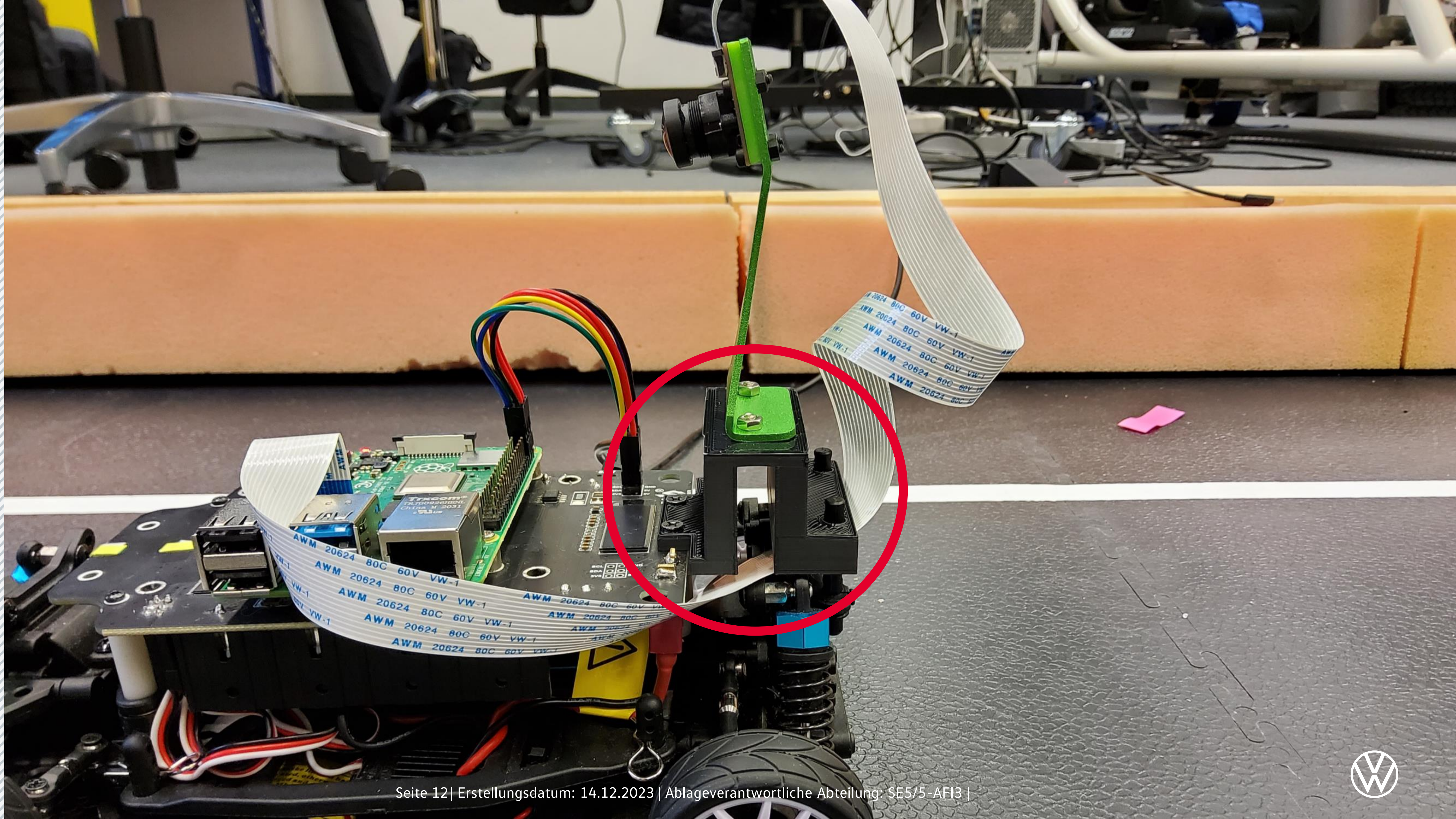




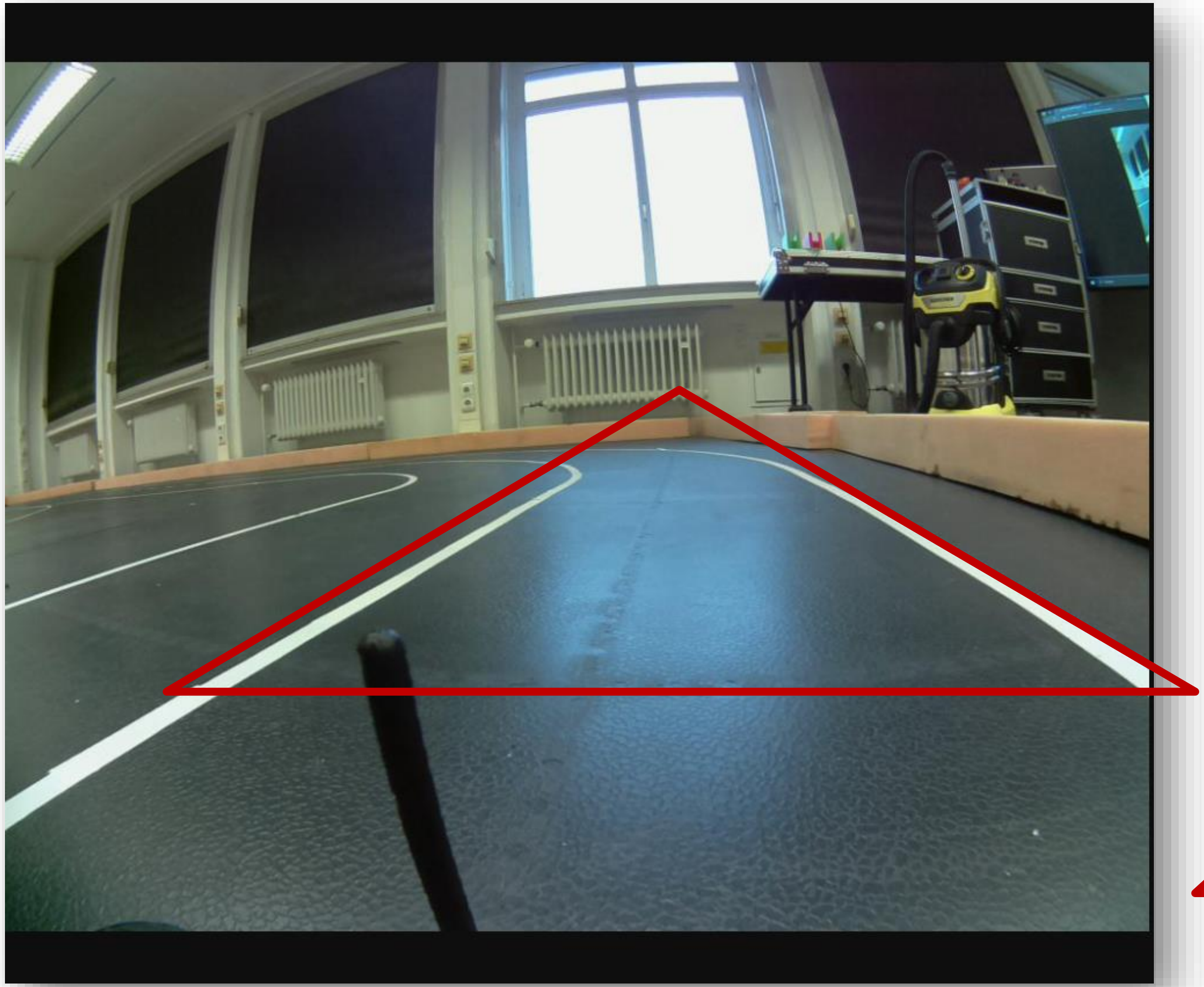
- Linienerkennung unzuverlässig
- Verzerrung durch Perspektive
- Schwer einzuordnen wo sich das Auto befindet

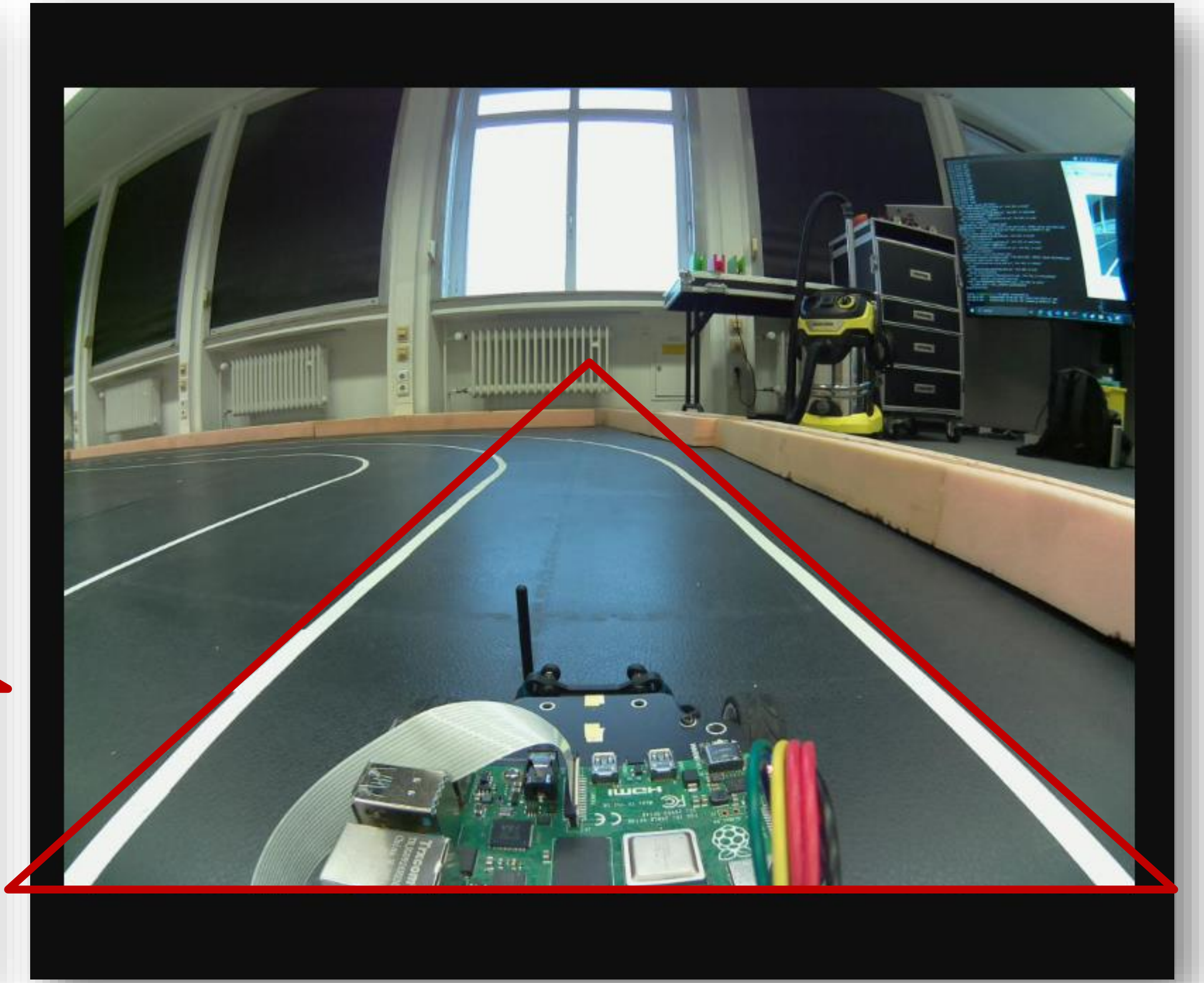
- Kurven verzehrt und schwer durchfahrbar
- Lenkwinkel muss vorrauschauend eingestellt werden
- Hough Lines Algorithmus entdeckt

  -> besseres Ergebnis erwartet


AWM 20624 80C 60V VW-1

Pi-Car-Turbo
Umbau der Kameraperspektive

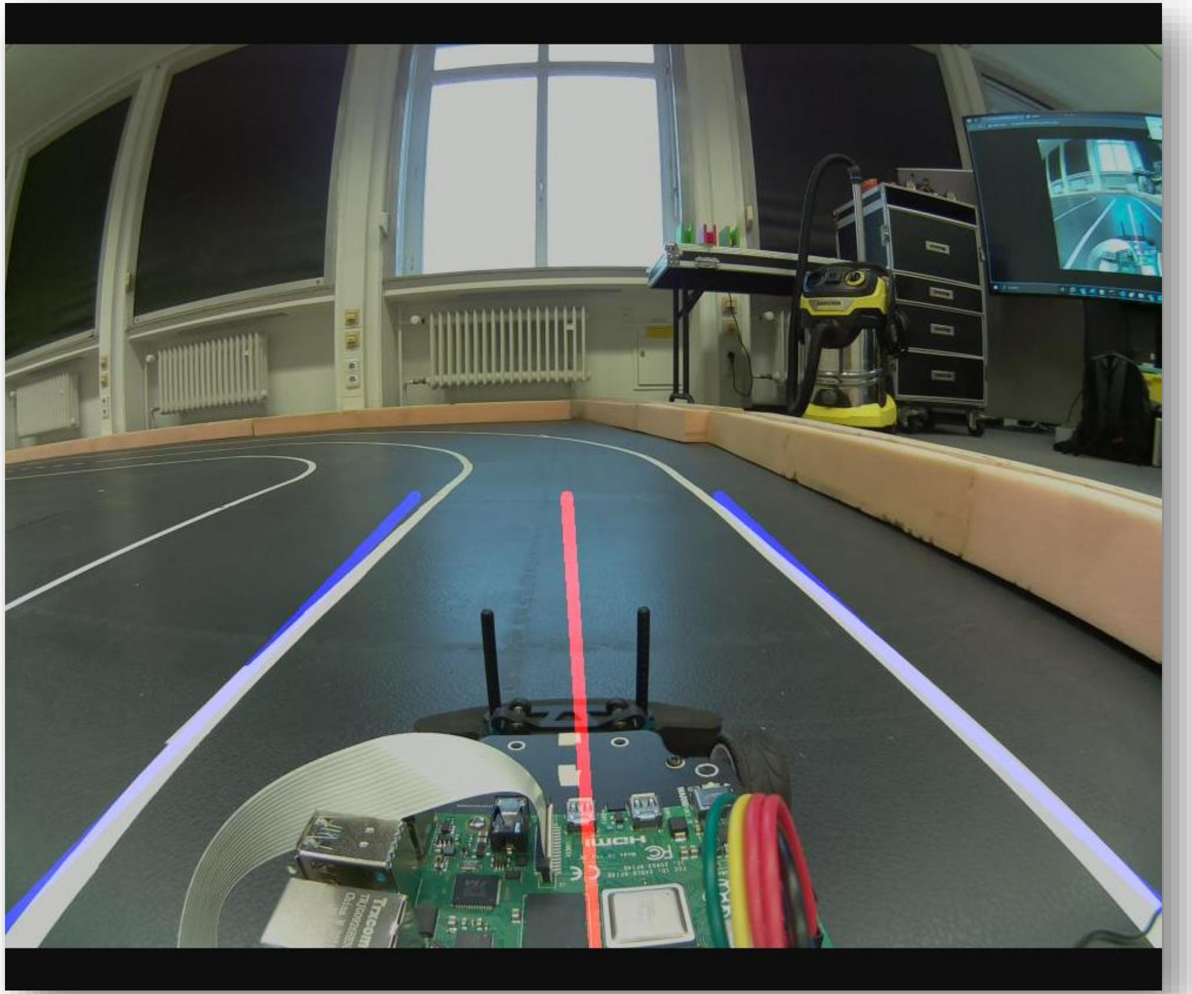


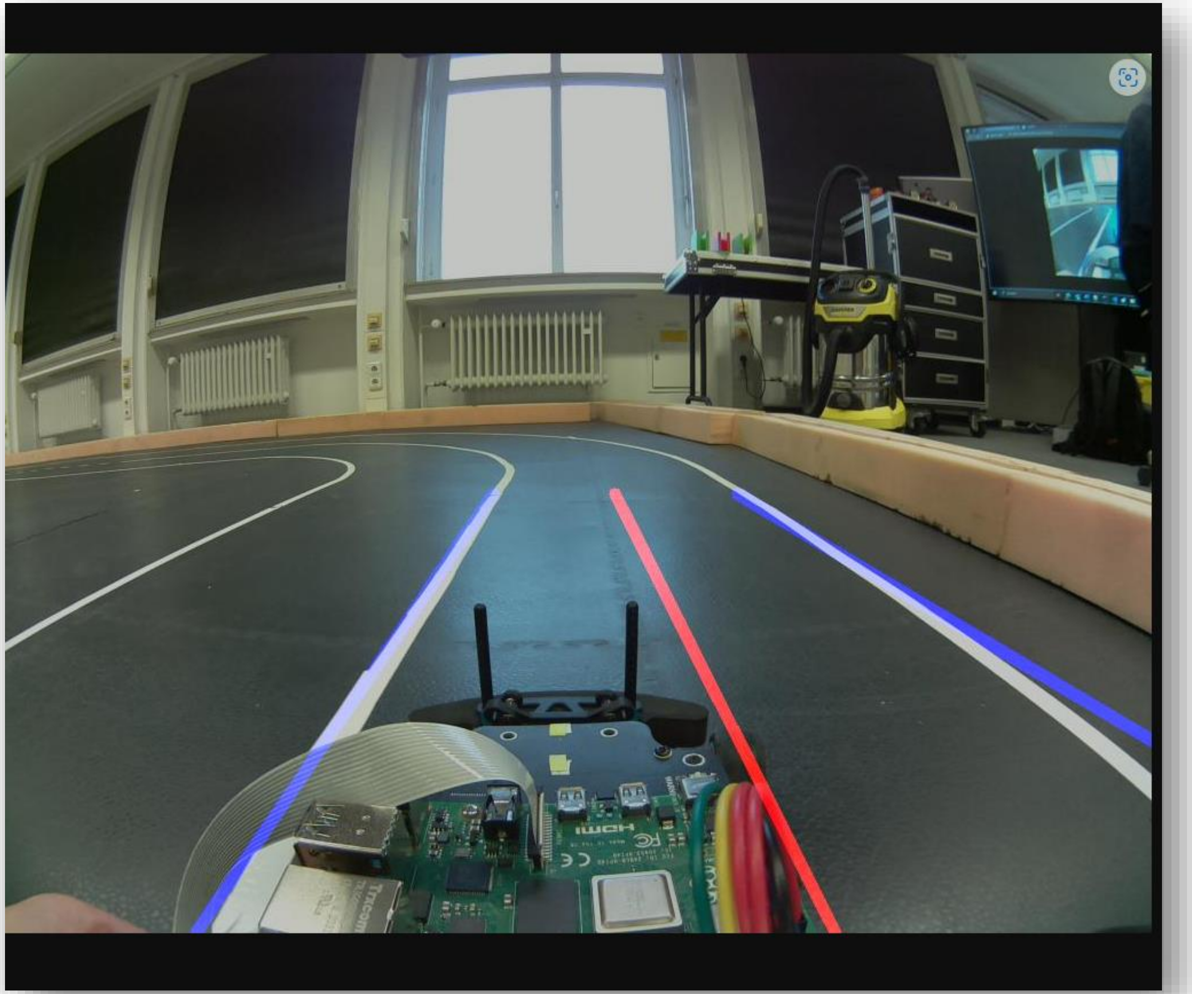


Pi-Car-Turbo
Implementierung

1. Linien ermitteln
2. Ausrichtung des Autos erkennen
3. Lenkung abhängig zur Entfernung der Außenlinien
4. Stream der Kamera über HTML-Seite

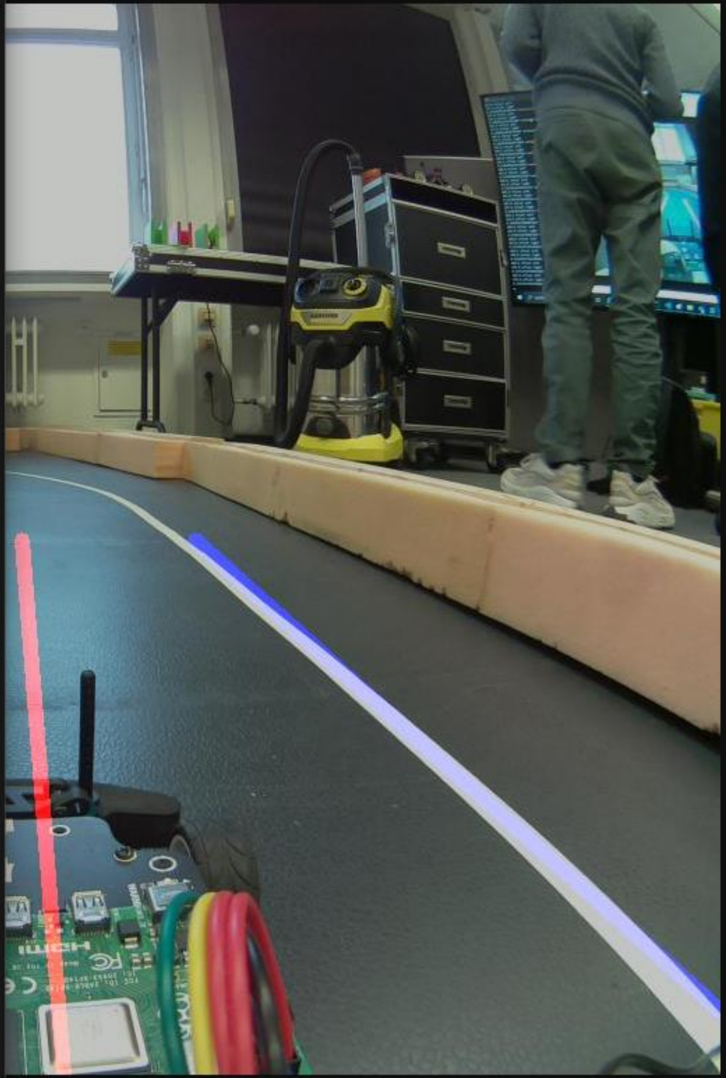
Pi-Car-Turbo
Implementierung





itlab@piracer: ~/cam

```
492.0 offset left
571.0 offset right
0.13835376532399302 variance left
486.0 offset left
561.0 offset right
0.1336898395721925 variance left
480.0 offset left
566.0 offset right
0.15194346289752647 variance left
489.0 offset left
567.0 offset right
0.13756613756613756 variance left
489.0 offset left
566.0 offset right
0.1360424028268551 variance left
479.0 offset left
563.0 offset right
0.14920071047957373 variance left
484.0 offset left
566.0 offset right
0.14487632508833925 variance left
488.0 offset left
562.0 offset right
0.1316725978647687 variance left
484.0 offset left
564.0 offset right
0.14184397163120566 variance left
480.0 offset left
568.0 offset right
0.15492957746478875 variance left
488.0 offset left
554.0 offset right
0.11913357400722024 variance left
485.0 offset left
565.0 offset right
0.1415929203539823 variance left
490.0 offset left
551.0 offset right
0.1107078039927405 variance left
485.0 offset left
562.0 offset right
0.13701067615658358 variance left
480.0 offset left
563.0 offset right
0.14742451154529312 variance left
490.0 offset left
563.0 offset right
0.12966252220248664 variance left
```



Pi-Car-Turbo
Schwarz-weiß Umwandlung

```python
def canny(image):
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    canny = cv2.Canny(blur, 50, 150)
    return canny
```

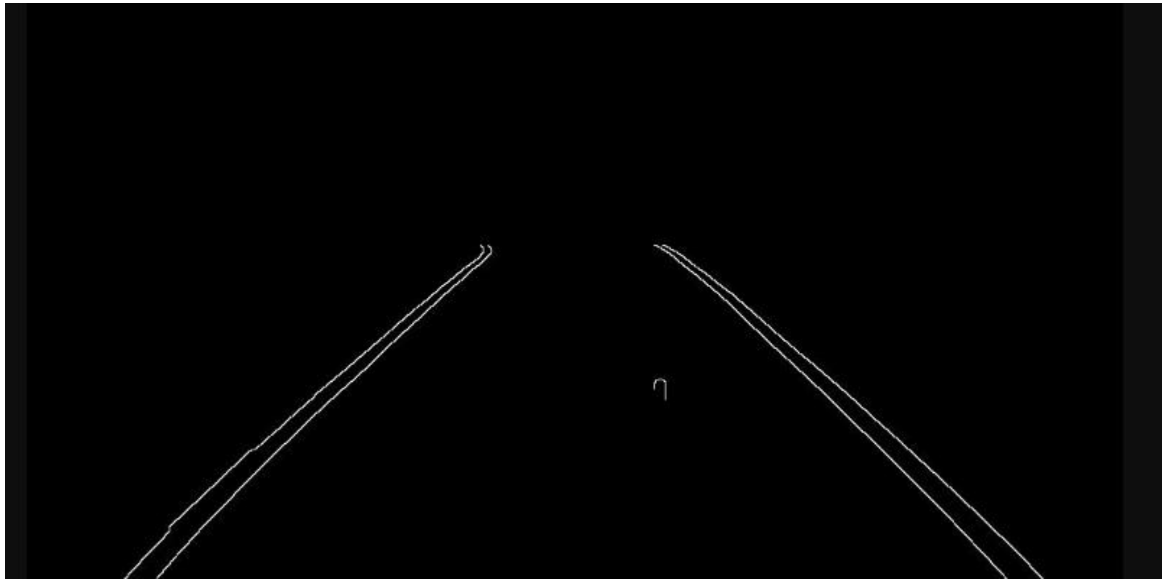
- Änderung der Farbe auf Schwarz-weiß,
- Blur entfernen
- Noise verringern
- Bildglättung anwenden

Grund:

- Edge detection
- Verbraucht weniger Ressourcen/ funktioniert schneller

Pi-Car-Turbo
Region of Interest

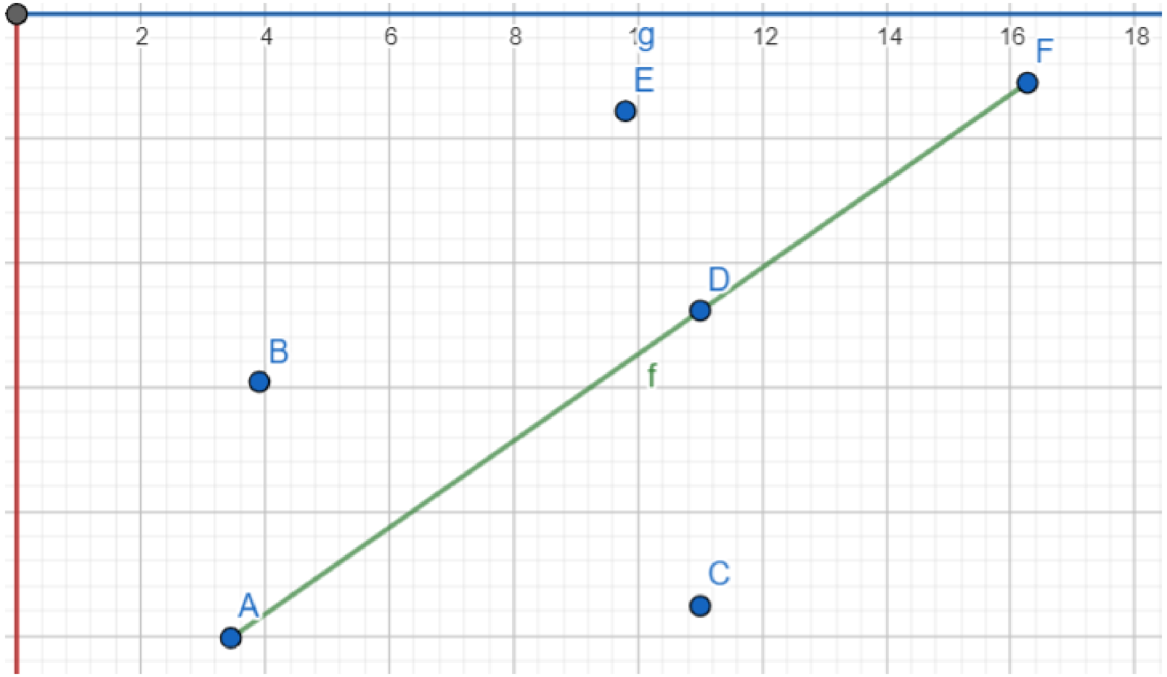
```python
def region_of_interest(image):
    height = image.shape[0]
    width = image.shape[1]

    polygons = np.array([
        [(0, height-50), (1100, height-50), (500, 250)]
        ])
    mask = np.zeros_like(image)
    cv2.fillPoly(mask, polygons, 255)
    masked_image = cv2.bitwise_and( image, mask)
    return masked_image
```



- Im Bild ein Dreieck definieren und weiß ausfüllen
- Maske (schwarzes Bild) definieren
- Dreieck in Maske einsetzen
- Farbdefinition im Dreieck weiß auf weiß -> weiß, schwarz auf weiß -> schwarz (bitwise_and)

Pi-Car-Turbo
Hough Lines

```python
        # Define the vertices of the triangle
        pts = np.array([[100, 800], [900, 800], [500, 400]], dtype=np.int32)
        pts2 = np.array([[100, 350], [900, 350], [500, 250]], dtype=np.int32)

        # Reshape the array into a 3D array with one row
        pts = pts.reshape((-1, 1, 2))
        pts2 = pts2.reshape((-1, 1, 2))

        # Draw the filled triangle on the black image
        cv2.fillPoly(canny_image, [pts, pts2], color=(0, 0, 0))


        cropped_image = region_of_interest(canny_image)
        lines = cv2.HoughLinesP(cropped_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=10)

        if lines is not None:
            averaged_lines = average_slope_intercept(frame, lines)
            line_image = display_lines(frame, averaged_lines)
            #line_image = display_lines(frame, lines)
```



- erkennt Linien
- Koordinaten von Linien in Array speichern

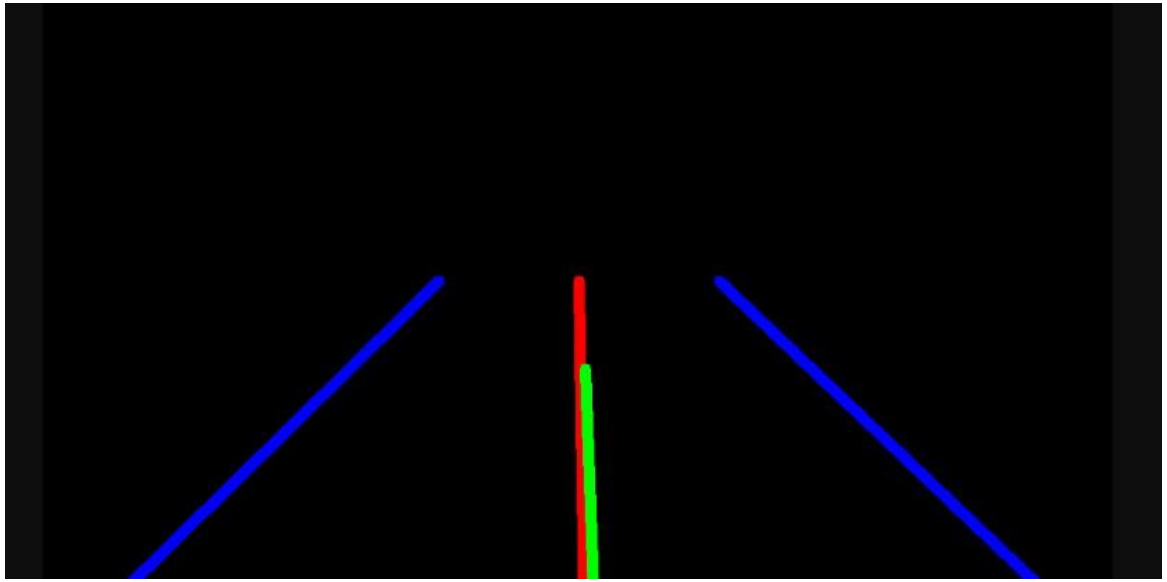
Pi-Car-Turbo
Average Slope Intercept

```python
def average_slope_intercept(image, lines):
    global backup_left_line, backup_right_line

    left_fit = []
    right_fit = []

    for line in lines:
        x1, y1, x2, y2 = line.reshape(4)
        parameters = np.polyfit((x1, x2), (y1, y2), 1)
        slope = parameters[0]
        intercept = parameters[1]

        if slope < 0:
            left_fit.append((slope, intercept))
        else:
            right_fit.append((slope, intercept))

    left_fit_average = np.average(left_fit, axis=0)
    right_fit_average = np.average(right_fit, axis=0)

    left_line = make_coordinates(image, left_fit_average, "left")
    right_line = make_coordinates(image, right_fit_average, "right")

    backup_left_line = left_line
    backup_right_line = right_line

    return np.array([left_line, right_line])
```

- Erstellung von Arrays
- prüfen ob positive oder negative Steigung vorliegt
- Definieren der linken und rechten Linie mit den Koordinaten
- Speichern der Linien als Backup falls kurz keine Linie erkannt wird

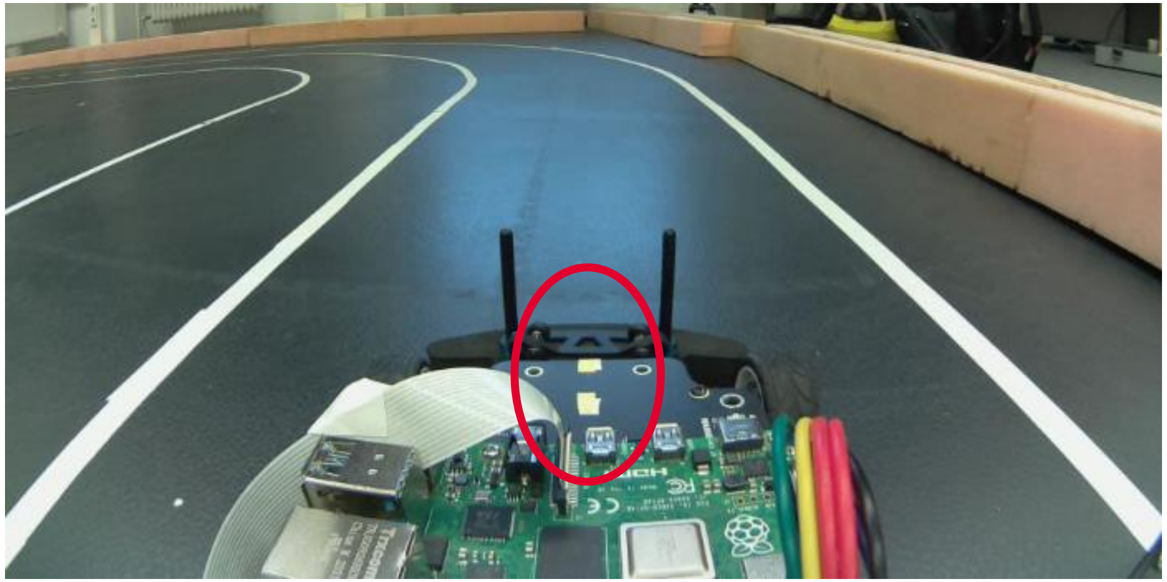
Pi-Car-Turbo
Display Lines

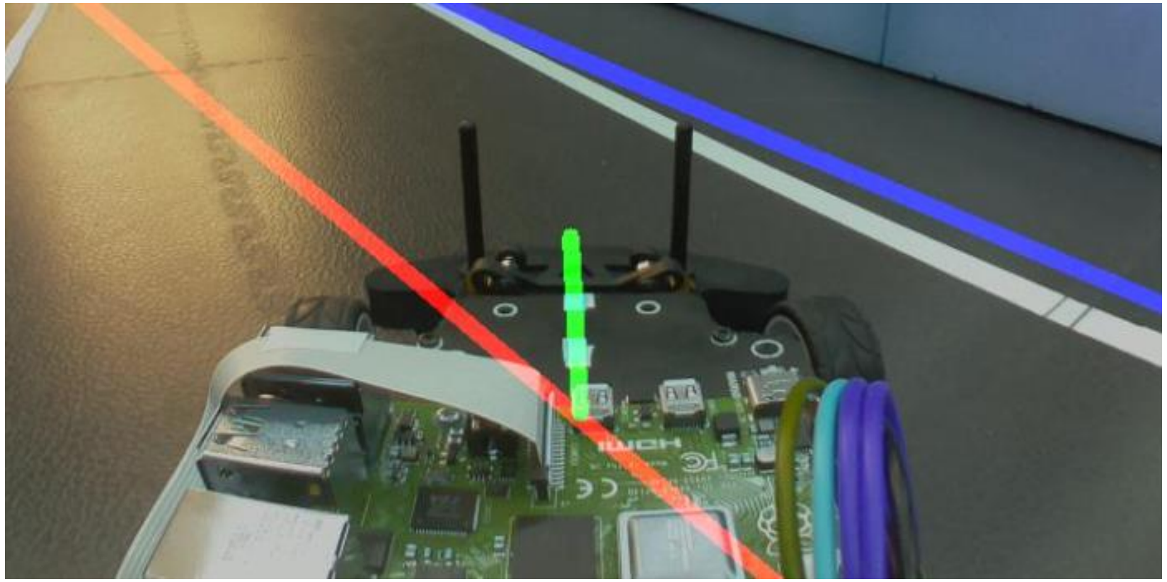
```python
def display_lines(image, lines, centroids, distance = 150):
    line_image = np.zeros_like(image)
    points_on_lines = []

    if lines is not None:
        for line in lines:
            #x1, y1 = line.reshape(2)
            #print(type(line_image))
            x1,y1,x2,y2 = line.reshape(4)
            cv2.line(line_image, (x1, y1), (x2, y2), (0, 0, 255), 10)

        x2 = coords_right[0]
        x1 = coords_left[0]
        x4 = coords_right[2]
        x3 = coords_left[2]
        y1 = coords_right[1]
        y2 = coords_right[3]

        # Calculate midpoints xu, yu, xo, yo
        xu = int(x1 + (x2 - x1) / 2)
        yu = int(y1)
        xo = int(x3 + (x4 - x3) / 2)
        yo = int(y2)

        # Draw a line connecting midpoints
        cv2.line(line_image, (xu, yu), (xo, yo), (255, 0, 0), 10)
```



- Zeichnen der blauen Linien
- Berechnung der Koordinaten für die rote Linie
- Zeichen der roten Linie

Pi-Car-Turbo
Ausrichtung des Autos



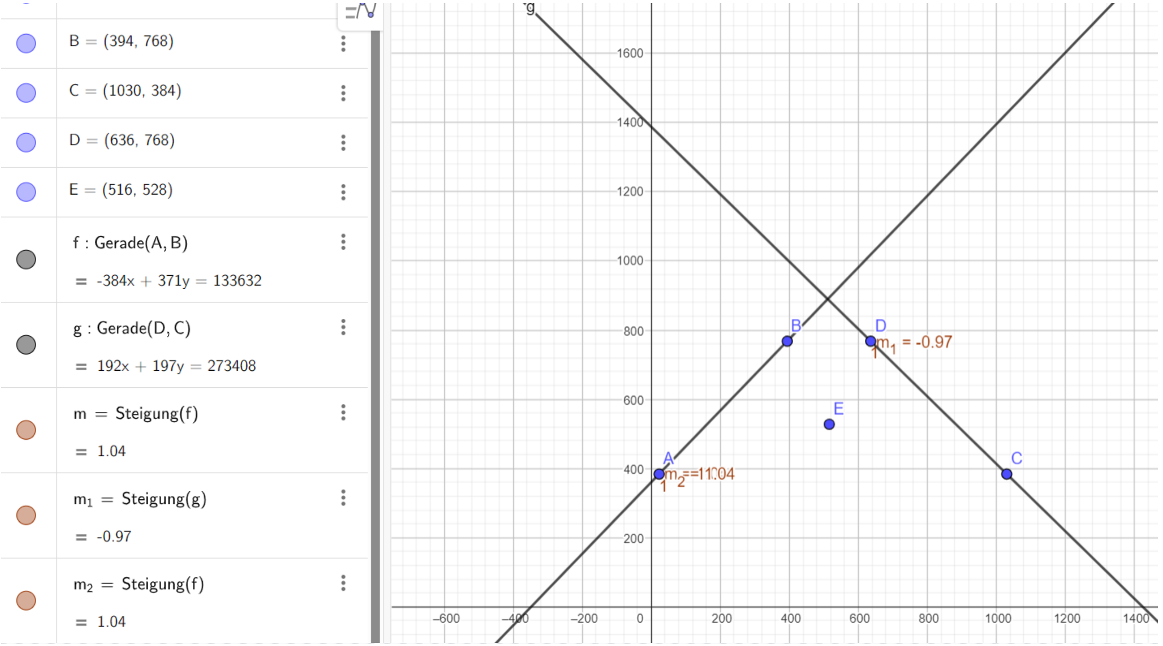


- Tracking Points auf dem Auto
- Bild in schwarz-weiß
- weiße Punkte und deren Koordinaten erkennen

- Tracking Points erkennen
- Linie durch Tracking Points ziehen
- Ausrichtung des Autos erkannt

Pi-Car-Turbo
Abstandsberechnung

```python
import numpy as np
import math


target_y = 528

px = 516

# Left line points
x1_left, y1_left = 394, 768
x2_left, y2_left = 23, 384

# Calculate slope and intercept for the left line
slope_left = (y2_left - y1_left) / (x2_left - x1_left)
intercept_left = y1_left - slope_left * x1_left

print("Left Line:")
print("Slope:", slope_left)
print("Intercept:", intercept_left)

# Calculate the perpendicular distance between the left line and the target point
distance_left = abs(slope_left * px - target_y + intercept_left) / math.sqrt(slope_left**2 + 1)

print(f"Distance left: {distance_left}")


# Right line points (corrected order)
x1_right, y1_right = 636,768
x2_right, y2_right = 1030,384

# Calculate slope and intercept for the right line
slope_right = (y2_right - y1_right) / (x2_right - x1_right)
intercept_right = y1_right - slope_right * x1_right

print("\nRight Line:")
print("Slope:", slope_right)
print("Intercept:", intercept_right)

# Calculate the perpendicular distance between the right line and the target point
distance_right = abs(slope_right * px - target_y + intercept_right) / math.sqrt(slope_right**2 + 1)

print(f"Distance right: {distance_right}")
```



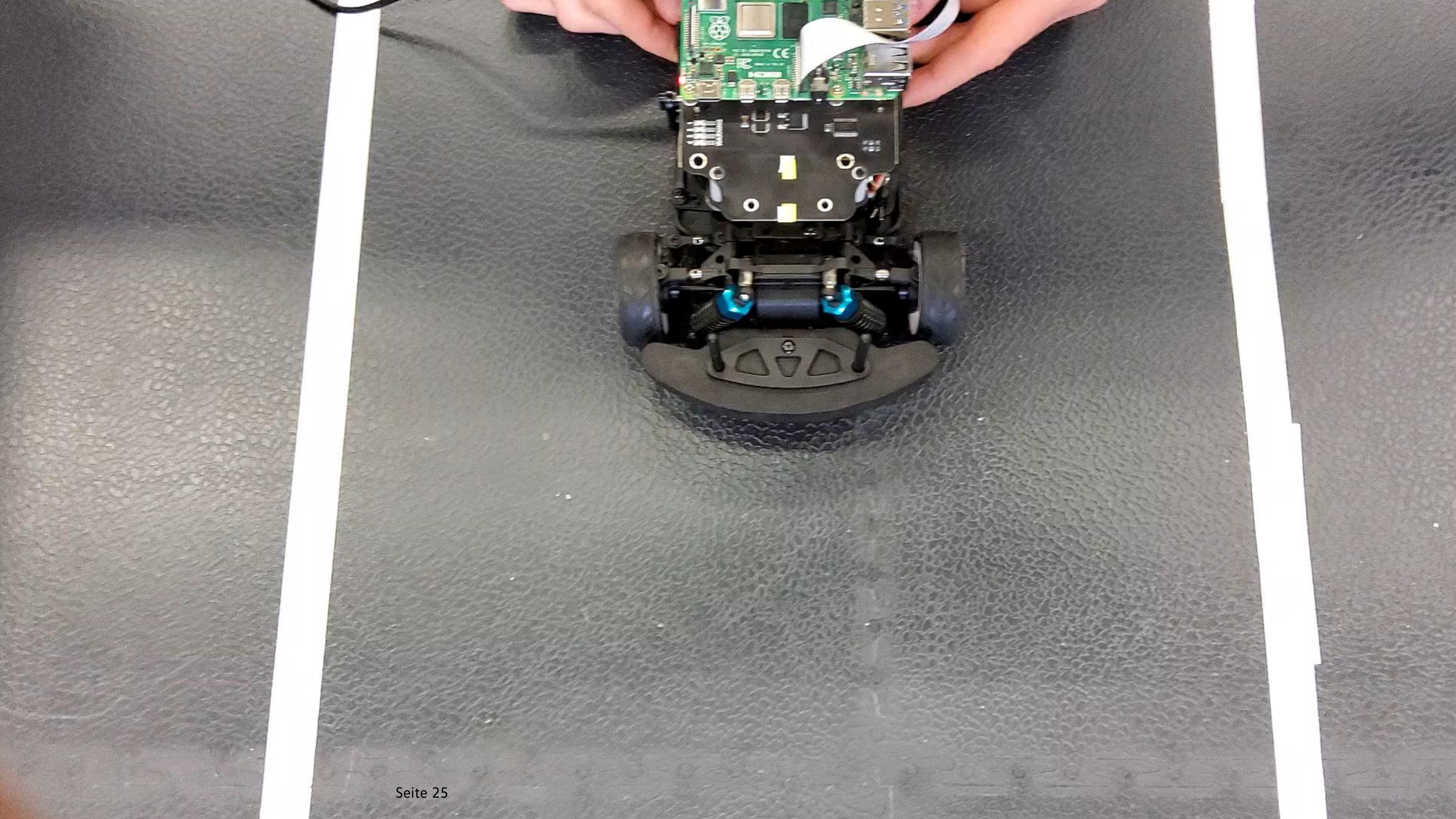
- Koordinaten der Linien und der Position des Autos
- Abstand des Punktes zu den Linien auf gleicher y-Koordinate

Pi-Car-Turbo
Lenkung

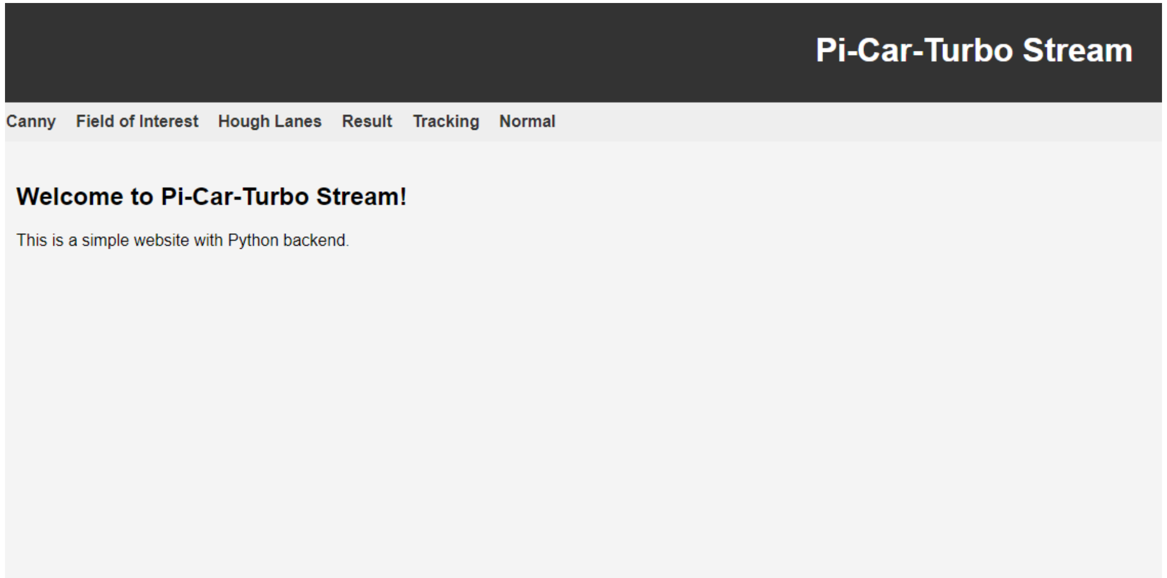
```python
def steer_based_on_distance_difference(self, distance_left, distance_right, threshold=5.0):
    #check the difference between left and right distances
    distance_left = distance_left + 40
    distance_difference = abs(distance_left - distance_right)
    max_difference = 30

    if distance_difference < threshold:
        #if the difference is small, move straight
        self.driving_instance.neutral_steering()
        steering_direction = "straight"
    elif distance_left < distance_right:
        #if the left distance is smaller, steer right
        variance = distance_difference / max_difference
        if variance <= 1:
            self.driving_instance.right_steering(variance)
        else:
            self.driving_instance.right_steering(1)

        steering_direction = "right"
```

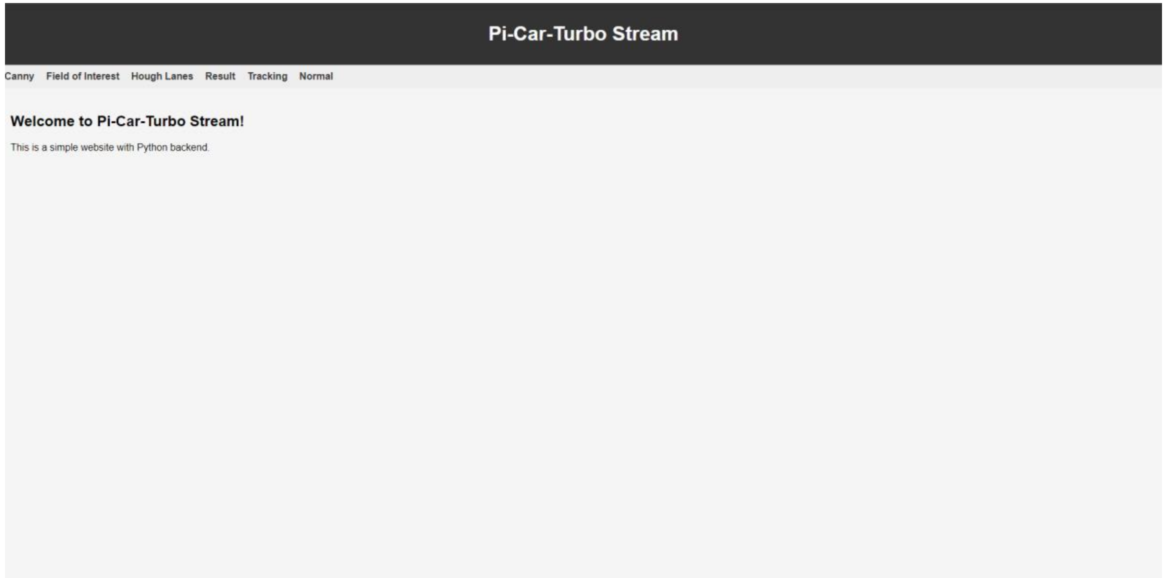
```python
    else:
        #if the right distance is smaller, steer left
        variance = distance_difference / max_difference
        if variance <= 1:
            self.driving_instance.left_steering(variance)
        else:
            self.driving_instance.left_steering(1)
        steering_direction = "left"

    return steering_direction
```

- anhand des errechneten Abstandes lenken
- Lenkungswinkel abhängig von der Entfernung


CE

Pi-Car-Turbo
Stream der Kamera über die HTML-Seite





verschiedene Kameraansichten:

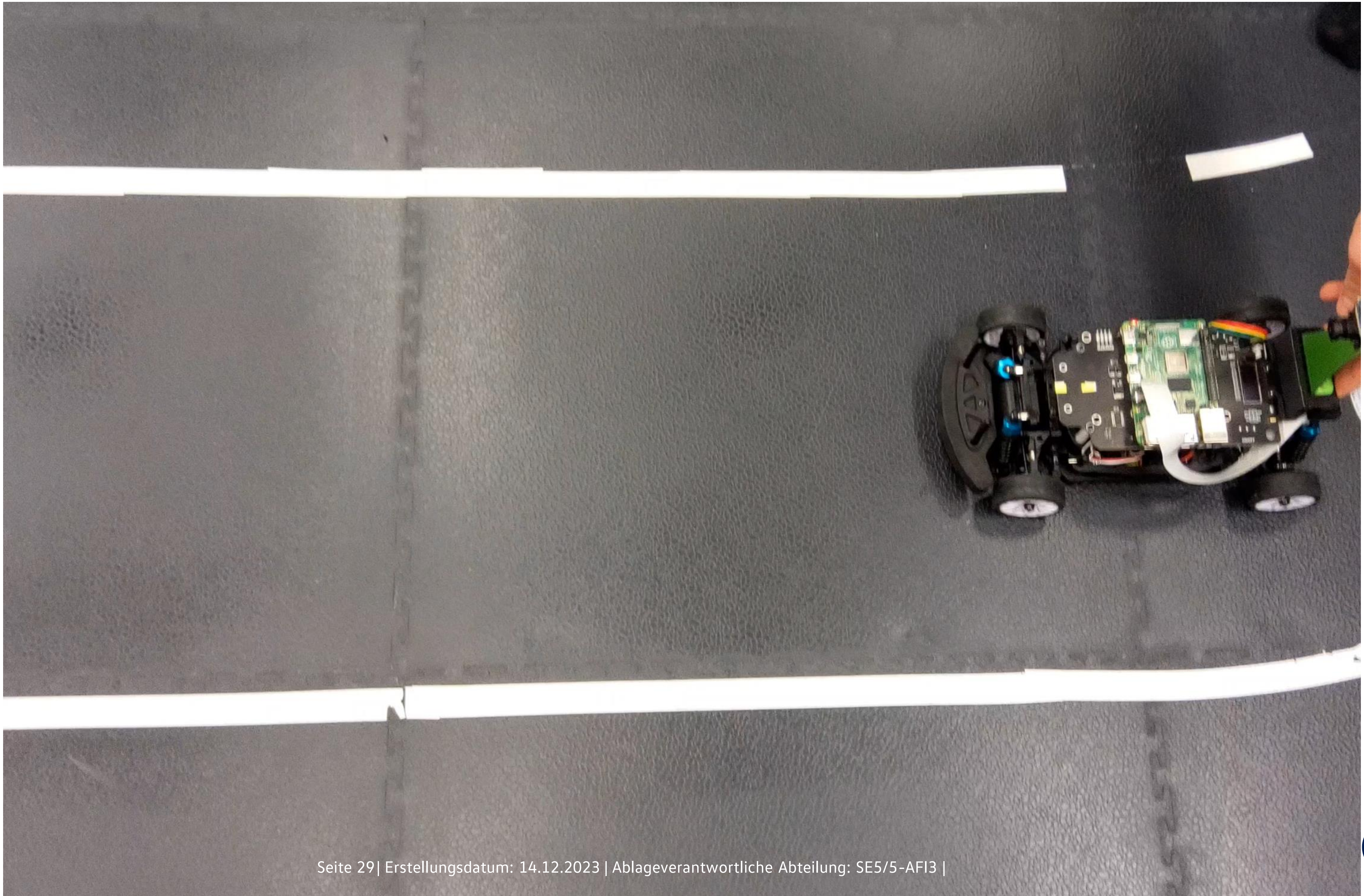
1. Kontraste schwarz-weiß dargestellt
2. Region of interest
3. Hough Lines
4. Kamerabild mit Hough Lines
5. Tracking points auf dem Auto
6. Kamerabild

3. Aktueller Stand des Projektes

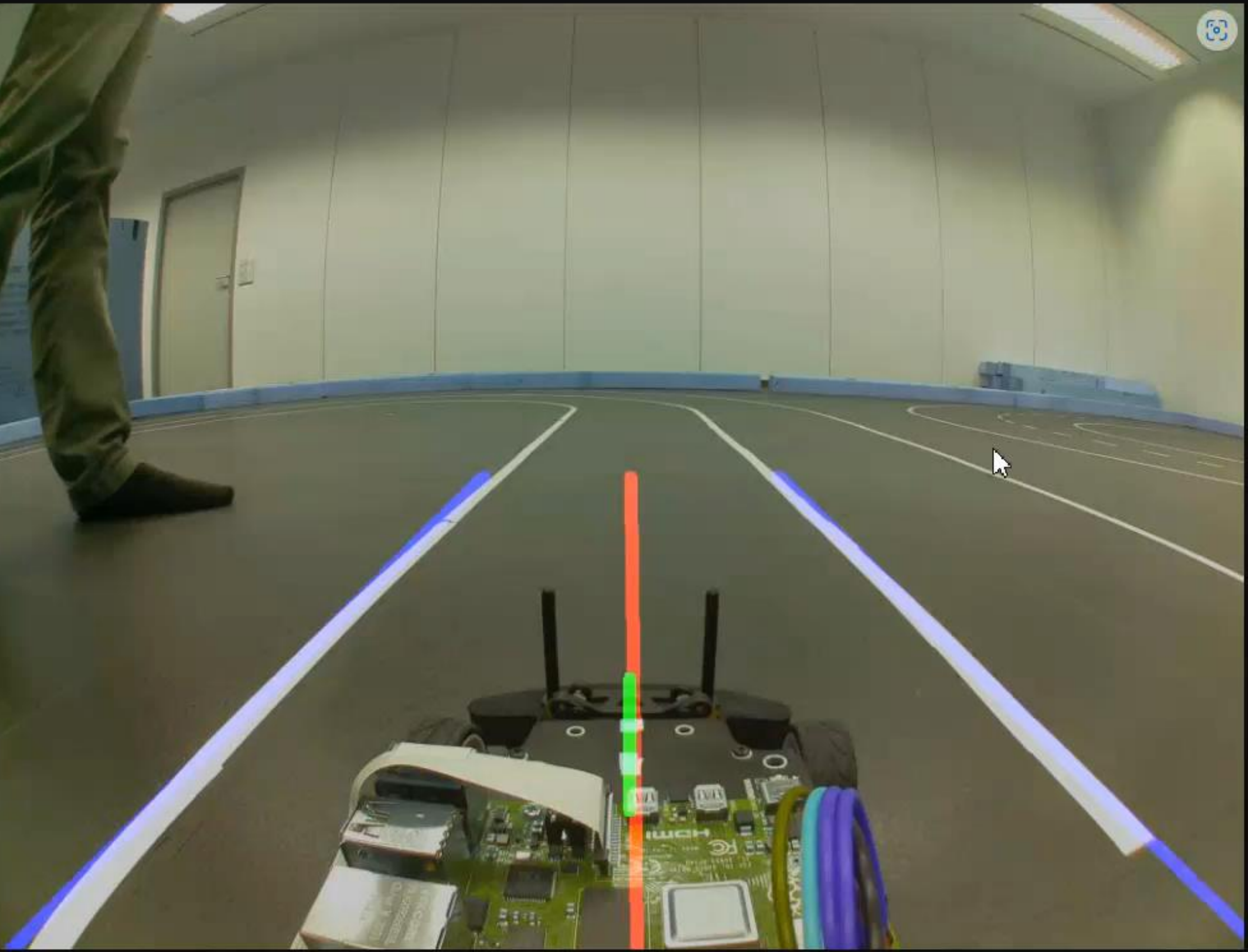
Pi-Car-Turbo
aktuelle Probleme

- Geschwindigkeit des Autos kann nicht weiter gedrosselt werden
- vorausschauendes Fahren (Kurvenerkennung)
- zu aufwendige Verarbeitung des Bildes

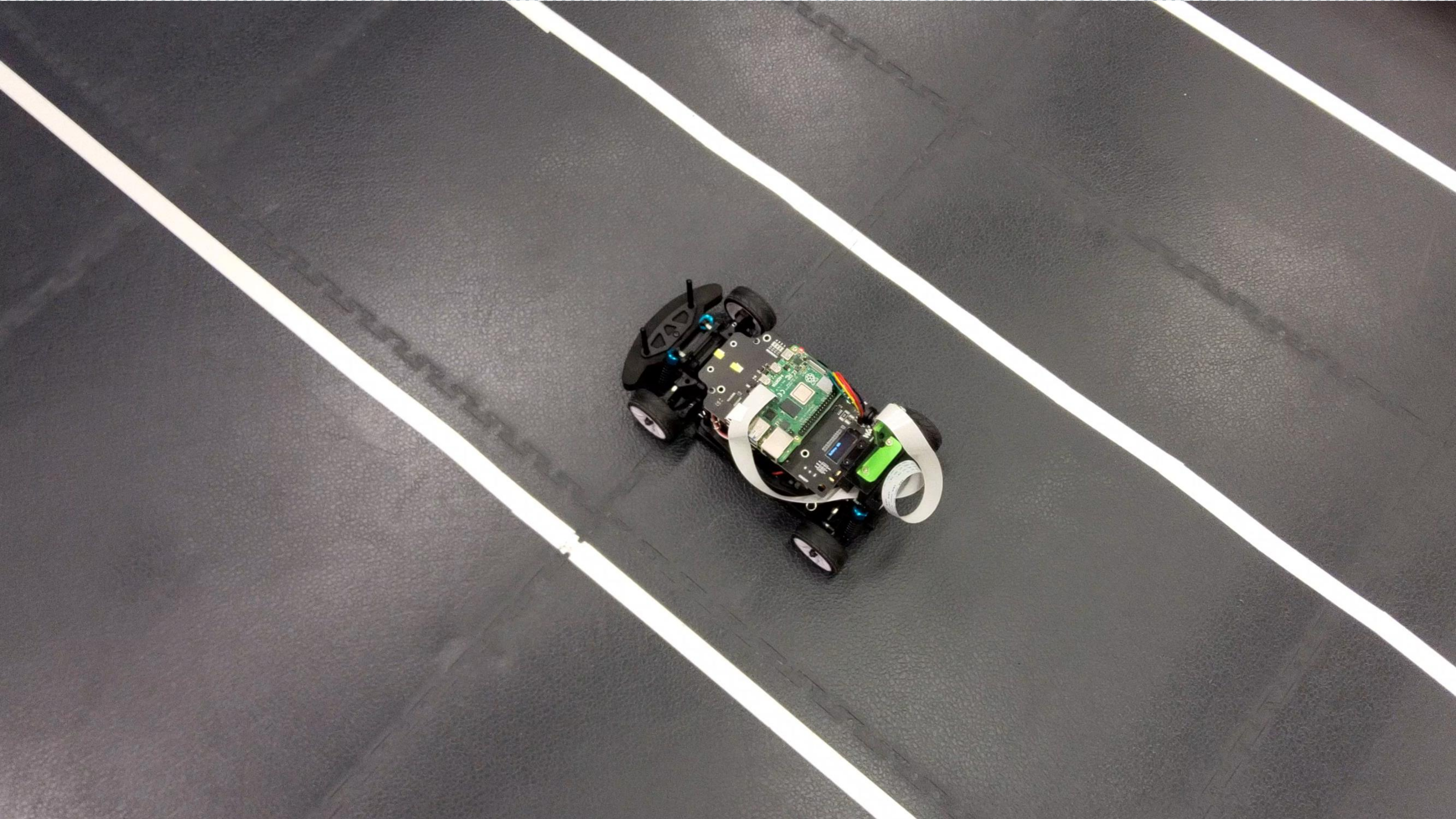
-> Stream zu leistungsintensiv



Nicht sicher | 192.168.0.118:8443/result_html.py



```
[1029  768  642  384]
test lines
Distance left: 177.73779205462952
Target point on the left line: (387.2067567567568, 655)
Closest point on the left line: (528.9056298498268, -4338.19838
5184383)
Distance right: 179.71656509290892
Target point on the right line: (272.29885057471284, 655)
Closest point on the right line: (435.14772201636185, 819.74246
29700402)
Steering direction: straight
[ 13 768 407 384]
[1035  768  645  384]
test lines
Distance left: 177.49383152508807
Target point on the left line: (387.44238410596006, 655)
Closest point on the left line: (529.1846703106685, -3997.84461
23719714)
Distance right: 179.81850113530786
Target point on the right line: (273.53932584269677, 655)
Closest point on the right line: (435.1139667985353, 820.289000
5180417)
Steering direction: straight
[ 23 768 407 384]
[1039  768  646  384]
test lines
Distance left: 177.44592443819516
Target point on the left line: (387.4885906040266, 655)
Closest point on the left line: (529.2392565173654, -3936.48896
11059934)
Distance right: 179.91379246893376
Target point on the right line: (274.7527675276754, 655)
Closest point on the right line: (435.0943865089261, 820.849537
1905303)
Steering direction: straight
[ 28 768 409 384]
[1029  768  642  384]
test lines
Distance left: 177.10421320606633
Target point on the left line: (387.81756756756755, 655)
Closest point on the left line: (529.6266185312822, -3542.54790
8525974)
Distance right: 179.71656509290892
Target point on the right line: (272.29885057471284, 655)
Closest point on the right line: (435.14772201636185, 819.74246
29700402)
Steering direction: straight
```



Nicht sicher | 192.168.0.118:8443/index...

Pi-Car-Turbo Stream

Canny Field of Interest Hough Lanes Result Tracking Normal

Welcome to Pi-Car-Turbo Stream!

This is a simple website with Python backend.

192.168.0.118:8443/result_html.py

```
  File "/home/itlab/cam/hough_lanes.py", line 181, in VideoCapture
    line_image = display_lines(frame, averaged_lines, centroids, backup_centroids)
  File "/home/itlab/cam/hough_lanes.py", line 97, in display_lines
    cv2.line(line_image, (x1, y1), (x2, y2), (0, 0, 255), 10)
cv2.error: OpenCV(4.8.1) :-1: error: (-5:Bad argument) in function 'line'
> Overload resolution failed:
>  - Can't parse 'pt1'. Sequence item with index 0 has a wrong type
>  - Can't parse 'pt1'. Sequence item with index 0 has a wrong type

WARNING:root:Removed streaming client ('192.168.0.189', 51634): OpenCV(4.8.1) :-1: erro
nt) in function 'line'
> Overload resolution failed:
>  - Can't parse 'pt1'. Sequence item with index 0 has a wrong type
>  - Can't parse 'pt1'. Sequence item with index 0 has a wrong type

^CTraceback (most recent call last):
  File "/home/itlab/cam/stream_html.py", line 376, in <module>
    init()
  File "/home/itlab/cam/stream_html.py", line 371, in init
    server.serve_forever()
  File "/usr/lib/python3.10/socketserver.py", line 232, in serve_forever
    ready = selector.select(poll_interval)
  File "/usr/lib/python3.10/selectors.py", line 416, in select
    fd_event_list = self._selector.poll(timeout)
KeyboardInterrupt

(venv) itlab@piracer:~/cam$ python stream_html.py
192.168.0.189 - - [16/Nov/2023 07:20:20] "GET /normal.py HTTP/1.1" 200 -
Traceback (most recent call last):
  File "/home/itlab/cam/stream_html.py", line 260, in do_GET
    self.send_frame(frame)
  File "/home/itlab/cam/stream_html.py", line 353, in send_frame
    self.wfile.write(b'--FRAME\r\n')
  File "/usr/lib/python3.10/socketserver.py", line 826, in write
    self._sock.sendall(b)
BrokenPipeError: [Errno 32] Broken pipe
WARNING:root:Removed streaming client ('192.168.0.189', 51763): [Errno 32] Broken pipe
192.168.0.189 - - [16/Nov/2023 07:20:23] "GET /normal.py HTTP/1.1" 200 -
Traceback (most recent call last):
  File "/home/itlab/cam/stream_html.py", line 260, in do_GET
    self.send_frame(frame)
  File "/home/itlab/cam/stream_html.py", line 353, in send_frame
    self.wfile.write(b'--FRAME\r\n')
  File "/usr/lib/python3.10/socketserver.py", line 826, in write
    self._sock.sendall(b)
BrokenPipeError: [Errno 32] Broken pipe
WARNING:root:Removed streaming client ('192.168.0.189', 51766): [Errno 32] Broken pipe
```

Pi-Car-Turbo
Lösungsideen und Ausblicke

- schnellere Verarbeitung der Bilder
- Geschwindigkeit erhöhen
- Bessere Kurvenerkennung (Blick ins Weite)
- Rennlinie auf der Strecke fahren

Vielen Dank.

Auszubildene Fachinformatiker im Bereich der Anwendungsentwicklung
SE 5-5/AFI3

Volkswagen Aktiengesellschaft
Brieffach 1594
Berliner Ring 2
D-38436 Wolfsburg

Stand: 20. Dezember 2023 I Version 1.6