

Technische Universität Dresden

Fakultät Elektrotechnik und Informationstechnik

Institut für Regelungs- und Steuerungstheorie

Studienarbeit

Verbesserung der Spurerkennung und -verfolgung autonomer Modellfahrzeuge

vorgelegt von: James Vero Asghar
geboren am: 3. Dezember 1997 in Austin, Texas, **Vereignete Staaten**

Betreuer: Dr.-Ing. Carsten Knoll
M.Sc. Paul Auerbach
Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. habil. Dipl.-Math. K. Röbenack
Tag der Einreichung: 2. Februar 2222



Bitte ersetzen Sie diese Seite vor dem Binden mit der Aufgabenstellung.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tage an der Fakultät Elektrotechnik und Informationstechnik eingereichte Studienarbeit zum Thema

Verbesserung der Spurerkennung und -verfolgung autonomer Modellfahrzeuge

selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Schriften entnommen sind, wurden als solche kenntlich gemacht.

Dresden, 2. Februar 2222

James Vero Asghar

Kurzfassung

An dieser Stelle fügen Sie bitte eine deutsche Kurzfassung ein.

Abstract

At the Connected Robotics Lab (CoRoLa) research group at the Barkhausen Institut, a demonstration using remote controlled 2 axis vehicles was developed. The demonstration is currently in use in order to model communications between vehicles similar to an Internet of Things (IoT) network.

In order to better model the complexities of a real world system a nonlinear control algorithm was introduced to the system in order to more accurately control the vehicles. The Stanley Controller is a nonlinear controller developed for use with the lateral control of 2 axis vehicles in mind. As input to the controller, a fisheye camera was used to create images of road. These images were then digitally processed to find the yellow line for the vehicles to follow.

Inhaltsverzeichnis

1	Einführung	VI
2	Regelalgorithmus	VII
2.1	Stanley-Regler	VII
2.1.1	Kleinsignalverhalten	VIII
2.1.2	Großsignalverhalten	VIII
2.2	Simulation	VIII
3	Bildverarbeitung	XII
3.1	Kamera-Kalibrierung	XII
3.2	„Color Thresholding“	XV
3.3	Perspektivtransformation	XVI
3.4	Histogramm	XVI
3.5	Gleitfenstermethode	XVII
3.6	Berechnung von Ausrichtung und Abstand	XVIII
3.7	Ausreißer	XIX
3.8	Pipeline-Optimierungen	XX
4	Experiment	XXI
4.1	Hardware	XXI
4.2	Experimentkriterien	XXI
4.2.1	Höchstgeschwindigkeit	XXI
4.2.2	Maximal möglicher Fehlerwinkel	XXI
4.2.3	Abstand von der Linie	XXII
4.2.4	Reglerausgang	XXII
4.3	Ergebnisse	XXII
5	Conclusion	XXIII

Kapitel 1

Einführung

Kapitel 2

Regelalgorithmus

2.1 Stanley-Regler

Der Stanley-Regler ist ein nichtlinearer Regelungsalgorithmus, der 2005 von der Stanford University entwickelt wurde, um ein zweiachsiges Fahrzeug nur anhand der Ausrichtung und der Querabweichung der zu verfolgenden Bahn zu steuern. Der Algorithmus hat sich für das kinematische Modell eines zweiachsigen Fahrzeugs als asymptotisch global stabil erwiesen.

Der Stanley-Regler ist ein Bahnfolgeregler anstelle eines Trajektorienfolgereglers. Als Querregler (?) soll er das Fahrzeug auf der Bahn halten, hat aber keinen Einfluss auf die Vorwärtshwindigkeit. Dieser Ansatz ermöglicht eine flexible Wahl der Fahrzeuggeschwindigkeit, die unter Berücksichtigung dynamischer Effekte nach Bedarf für eine bestimmte Anwendung gewählt werden kann. (?)

Der Stanley-Regler wird mathematisch durch die folgende Gleichung dargestellt:

$$u = \theta - \theta_d + \arctan\left(\frac{ke_{fa}}{v}\right). \quad (2.1)$$

Dabei ist u den Reglerausgang, θ die aktuelle Ausrichtung des Fahrzeugs, θ_d die Pfadausrichtung, k ein Skalierungsfaktor, v die Geschwindigkeit des Fahrzeugs und e_{fa} die Querabweichung vom Mittelpunkt der Vorderachse des Fahrzeugs zum Kurs.

Der Stanley-Regler besteht aus zwei Komponenten: einer Komponente, die die Abweichung des Fahrzeugs vom Kurs behandelt, und einer Komponente, die die Differenz zwischen der Ausrichtung des Fahrzeugs und der des Kurses behandelt. Die erste Komponente wird durch $\arctan(\frac{ke_{fa}}{v})$ dargestellt und steuert den Lenkwinkel so, dass er sich auf den Kurs zubewegt, wenn sich das Fahrzeug weiter von ihm entfernt. Die zweite Komponente, dargestellt durch $\theta_d - \theta$, steuert den Lenkwinkel so, dass er parallel zum zu verfolgenden Kurs bleibt. Zusammen bewirken diese beiden Komponenten (?), dass das Fahrzeug auf den gewünschten Kurs gelenkt wird.

2.1.1 Kleinsignalverhalten

Um das Verhalten des Stanley-Reglers besser zu verstehen, wird der Regler um den Punkt $(x_V, \theta_d) = (0, 0)$ linearisiert, was zu folgender Gleichung führt:

$$\bar{u} \approx \bar{\theta} + \frac{k}{v} e_{fa}. \quad (2.2)$$

Die sich daraus ergebende Gleichung (2.2) hat die Form eines PID-Reglers (Proportional-, Integral- und Derivativ-Regler), der nur P- und D-Komponenten hat. Die P-Komponente wird durch $\frac{k e_{fa}}{v}$ dargestellt. Da die Vorwärtsgeschwindigkeit des Fahrzeugs als konstant angenommen wird, besteht eine Proportionalität zwischen den Ableitungen von e_{fa} nach Zeit und Raum. Die Ableitung von e_{fa} nach dem Raum ist θ . Daher ist θ die D-Komponente des linearisierten Stanley-Reglers.

Folglich ist das Verhalten des Stanley-Reglers für kleine Eingangssignale ähnlich wie das eines PD-Reglers mit dem Fehlerterm e_{fa} .

2.1.2 Großsignalverhalten

Bei großen Signalen wird der Stanley-Regler durch das Verhalten der $\arctan(\dots)$ -Funktion sowie durch die zyklische Natur der Ausrichtung θ dominiert. Die Funktion $\arctan(\dots)$ ist auf einen Wert zwischen -90 und 90 Grad begrenzt und ist glatt. Die Ausrichtung ist ebenfalls auf den vorderen Halbkreis bzw. -90 und 90 Grad begrenzt, da davon ausgegangen wird, dass sich das Fahrzeug in eine Richtung bewegt. (?) Zusammen begrenzen diese Komponenten den Lenkwinkel (?) des Fahrzeugs.

2.2 Simulation

Um ein besseres Verständnis für das Verhalten des Stanley-Reglers in einem ferngesteuerten Fahrzeug zu gewinnen, haben wir eine Simulation des Regelkreises durchgeführt. Die Simulation umfasst drei Hauptkomponenten: den Kurs, dem das Fahrzeug folgen muss, den Stanley-Regler und das Fahrzeugmodell. Zur Untersuchung der Zeitreihendaten des zweiachsigen Fahrzeugs in der Simulation wurde ein Differentialgleichungslöser eingesetzt.

Der Kurs, dem das Fahrzeug folgen muss, ist eine virtuelle Darstellung der realen Strecke (?), die das Fahrzeug durchfahren wird. (?) Eine visuelle Darstellung der Fahrbahn findet sich in Abb. 1. In der Simulation wird der Kurs durch die folgende kontinuierliche statische Funktion definiert:

$$(x, y, h) = f(t, t_f), \quad (2.3)$$

wobei t und t_f die Eingangsparameter Zeit bzw. **Simulationsendzeit** (?) sind und (x, y, h) die kartesischen Koordinaten des Pfadpunkts und die entsprechende Ausrichtung an diesem Punkt darstellt. Diese Funktion wird für eine beliebige Anzahl von t -Werten diskretisiert, um ein **Array** von $(x, y, theta)$ -Werten zu erzeugen. Der Differentialgleichungslöser bezieht den Stanley-Controller und das Fahrzeugmodell ein, um die Zeitreihen des zweiachsigen Fahrzeugs unter Verwendung dieses Arrays als Eingabe zu analysieren. (?)

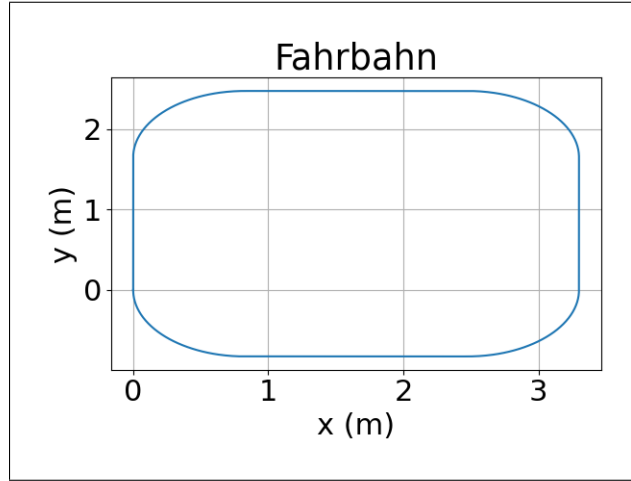


Abbildung 1 – Fahrbahn

Das verwendete Fahrzeugmodell ist das Modell eines Hinterachsfahrzeugs. Dieses Modell wird durch die folgende nichtlineare Zustandsraumdarstellung dargestellt:

$$\dot{x} = v \cos(\theta) \quad (2.4)$$

$$\dot{y} = v \sin(\theta) \quad (2.5)$$

$$\dot{\theta} = \frac{v}{l} \tan(\varphi) \quad (2.6)$$



$$\dot{\varphi} = \frac{(\delta - u)}{T}. \quad (2.7)$$

Die Zustandsvariablen des Modells sind x_H , y_H , ϕ und θ , wobei x_H und y_H die kartesischen Koordinaten des Mittelpunkts der Hinterachse sind und θ die Ausrichtung des Fahrzeugs ist. φ ist der Lenkwinkel des Fahrzeugs. Die Eingangsgrößen des Modells sind v und φ , wobei v die Geschwindigkeit des Fahrzeugs und φ der Lenkwinkel ist. Der Lenkwinkel φ ist selbst ein dynamisches System, das durch eine lineare Differentialgleichung erster Ordnung mit der konfigurierbaren Zeitkonstante T dargestellt wird. Das Verhalten des Stanley-Reglers bei dynamischen Einflüssen auf den Lenkwinkel ist von großer Bedeutung, da die globale asymptotische Stabilität des Reglers nur für das kinematische Zweiachsmodell nachgewiesen wurde.

Da der Stanley-Regler für die Querabweichung den Mittelpunkt der Vorderachse benötigt, muss dieser aus der Hinterachse berechnet werden. Der Mittelpunkt der Vorderachse

wird aus den Zustandsvariablen durch die folgende statische Funktion berechnet:

$$\underline{p}_H := \begin{bmatrix} x_H & y_H \end{bmatrix}^T \quad (2.8)$$



$$\underline{p}_V := \begin{bmatrix} x_V & y_V \end{bmatrix}^T \quad (2.9)$$

$$\underline{p}_V = \underline{p}_H + l \begin{bmatrix} \cos(\theta + \pi/2) \\ \sin(\theta + \pi/2) \end{bmatrix} \quad (2.10)$$

wobei x_V und y_V die kartesischen Koordinaten dieses **Mittelpunkts** sind.

Wie bereits erwähnt, (?) besteht der Stanley-Regler aus mehreren Teilen, die getrennt voneinander berechnet werden **müssen**. Die Ausrichtung θ ist eine Zustandsvariable, daher ist sie immer verfügbar. Um die Querabweichung und die Kursrichtung (?) zu berechnen, muss der richtige Bahnpunkt gewählt werden. Der Stanley-Regler verwendet den nächstliegenden Bahnpunkt vom Vorderachsmittelpunkt des Fahrzeugs, um die Ausrichtung des Kurses θ_d und die aktuelle Querabweichung e_{fa} zu bestimmen. Für die Simulation wird dies bestimmt, indem der Punkt mit dem geringsten Abstand zum Vorderachsmittelpunkt $\delta \underline{x}$ gefunden wird. An diesem Punkt wird dann die Ausrichtung des Kurses ermittelt. Das Skalarprodukt zwischen dem Vektor senkrecht zur Fahrzeugausrichtung \underline{x}_\perp und dem Vektor vom Pfadpunkt zur Fahrzeugvorderachse $\delta \underline{x}$ ist die Querabweichung e_{fa} . Zur Verdeutlichung ist in ABBILDUNG eine visuelle Darstellung dieser beiden Vektoren und ihres Skalarprodukts abgebildet. Dieser Algorithmus wird in folgende Algorithmus dargestellt.

For Paul: this figure will take a bit more time to make.

Algorithm 1 Berechnung von Querabweichung

$$\underline{v}_\perp \leftarrow [-\cos(\theta + \pi/2), -\sin(\theta + \pi/2)]^T$$

$$\Delta \underline{x} \leftarrow \underline{p}_V - \underline{p}_P$$

$$\delta \underline{x} \leftarrow \min ||\Delta \underline{x}||_2$$

$$\theta_d \leftarrow \arctan \left(\frac{\frac{d}{dy} \delta \underline{x}}{\frac{d}{dx} \delta \underline{x}} \right)$$

$$e_{fa} \leftarrow \delta \underline{x} \cdot \underline{v}_\perp$$



Der Kurs und die Querabweichung werden dann in den Stanley-Regler eingegeben, und der Ausgang des Reglers wird in das Fahrzeugmodell eingespeist. Ein Übersichtsdiagramm des Regelkreises ist in ABBILDUNG dargestellt.

For Paul: this figure will take a bit more time to make.

Der Stanley-Regler arbeitet mit zeitkontinuierlichen Signalen, die sich von den **vom Fahrzeug** verwendeten zeitdiskreten Signalen unterscheiden. Das Fahrzeug nimmt Bilder mit einer festen Abtastfrequenz auf, die von **der Pipeline** verarbeitet werden, bevor sie in den Regler eingespeist werden. Um dieses Verhalten zu simulieren, wird ein **Cache** in

den Simulator eingebaut, um die Ausrichtung und die Querabweichung zu speichern. Der Stanley-Regler verwendet dann diese zwischengespeicherten Werte für eine bestimmte Dauer. Dieser Cache-Ansatz ahmt das Verhalten eines Halteglieds nullter Ordnung nach, und die **Ausgabe** wird anschließend in den Stanley-Regler eingespeist. Dieser Simulationsansatz ermöglicht die Untersuchung des Fahrzeugverhaltens für verschiedene Abtastfrequenzen und **erlaubt die Festlegung einer Toleranz für die Rechengeschwindigkeit (?) der Pipeline**. Der Vergleich der Simulationsergebnisse bei verschiedenen Abtastfrequenzen ist in Abb. 2 dargestellt.

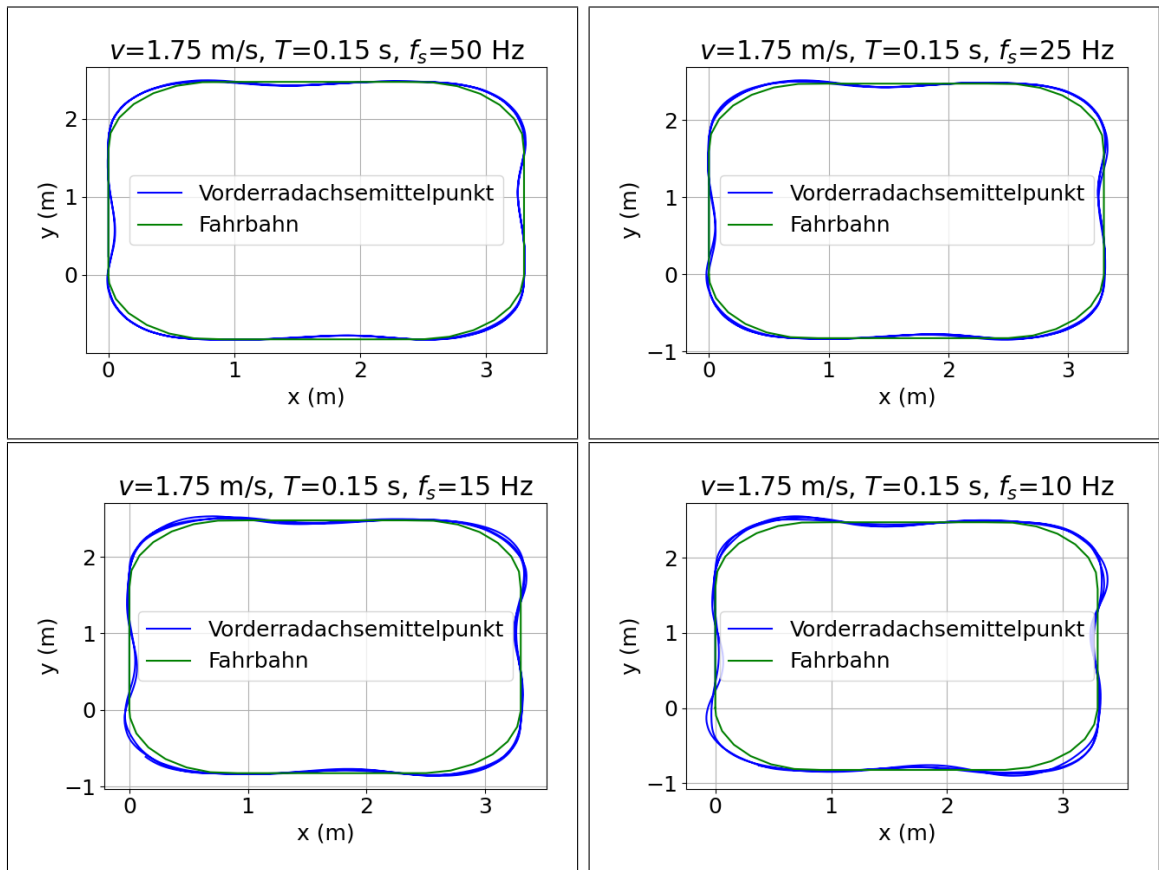


Abbildung 2 – **Abtastfrequenzen**

Wie in Abb. 2 dargestellt, wird der Stanley-Regler durch die Abtastfrequenz beeinflusst, wobei bei niedrigeren Frequenzen ein deutlicher Effekt zu beobachten ist. Oberhalb eines bestimmten Schwellenwerts von 15 Hz scheint das Verhalten des Fahrzeugs jedoch nicht wesentlich durch die Abtastfrequenz beeinflusst zu werden. (?)

Die Simulation **wird** dann für eine ausgewählte Zeitspanne durchgeführt.

Kapitel 3

Bildverarbeitung



3.1 Kamera-Kalibrierung

Am Anfang der Bildverarbeitungspipeline steht die Kamerakalibrierung. Für die Co-RoLa Car Platform wurde eine Fischaugenkamera im Gegensatz zu einer geradlinigen Kamera gewählt. Der Vorteil einer Fischaugenkamera ergibt sich aus ihrem größeren Blickwinkel im Vergleich zu einer geradlinigen Kamera. Eine Fischaugenkamera hat jedoch von Natur (?) aus eine tonnenförmige Verzeichnung, wodurch gerade Linien gekrümmt werden. Die Krümmung dieser Linien hängt von ihrem radialen Abstand vom Bildmittelpunkt ab. Ein Beispiel ist in Abb. 3 zu sehen. Diese Verzeichnung kann mit Hilfe der Software durch die sogenannte Kamerakalibrierung kompensiert werden.

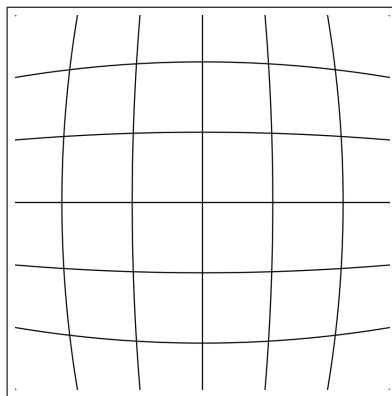


Abbildung 3 – Verzeichnung

Die Kamerakalibrierung wird verwendet, um die extrinsischen und intrinsischen Parameter einer Kamera anzunähern. Eine kalibrierte Kamera ermöglicht es, 3-D-Informationen aus einem 2-D-Bild zu gewinnen. Die intrinsischen Parameter einer Kamera werden

häufig durch die folgende 3x3-Matrix dargestellt:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.1)$$

wobei f_x und f_y die Brennweiten der Kamera in Form von Pixeln (?) in den Richtungen x und y und c_x und c_y die Koordinaten des Hauptpunkts sind. Die extrinsischen Parameter werden häufig durch die folgende Matrix dargestellt:

$$\begin{bmatrix} R & T \end{bmatrix}. \quad (3.2)$$

Dabei ist R die Rotationsmatrix der Kamera in Bezug auf das Laborsystem und T der Positionsspaltenvektor des Ursprungs des Laborsystems, ausgedrückt (?) in den Koordinaten des Kamerabildes. Die Kameramatrix M ist die Abbildung von den Weltkoordinaten auf Pixelkoordinaten, die durch die Matrix-Matrix-Multiplikation dargestellt wird,

$$M = K \begin{bmatrix} R & T \end{bmatrix}. \quad (3.3)$$

Durch Rekonstruieren der M -Matrix mit Hilfe von Software kann das verzerrte Bild wieder in das Weltbild eingefügt werden, wodurch gekrümmte Linien gerade werden.

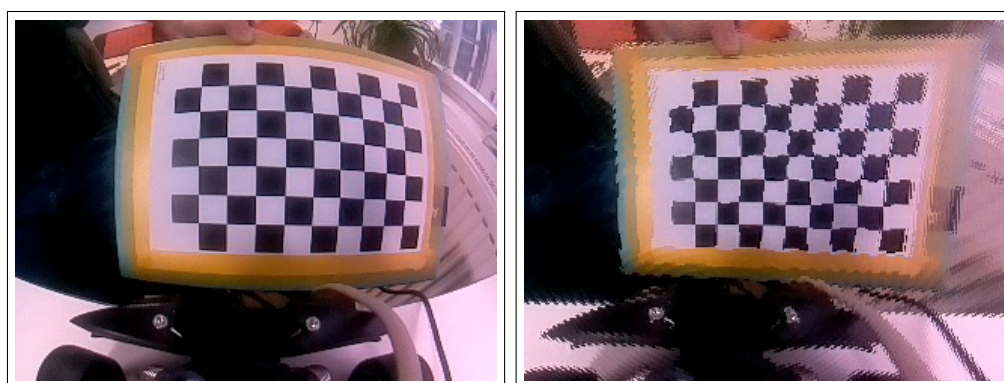


Abbildung 4 – Calibrated Images.

Eine Kamera wird anhand einer Sammlung von Fotos mit bekannten geraden Linien kalibriert. In der Regel wird eine Reihe von Schachbrettbildern mit bekannten Dimensionen verwendet. Danach werden Fotos des Schachbretts in verschiedenen Winkeln und an verschiedenen Stellen der Szene aufgenommen. Diese Bilderserie wird dann in den Algorithmus zur Kamerakalibrierung eingespeist, der zunächst die Positionen der Schachbrettfelder und der sie verbindenden Linien bestimmt. Anschließend gleicht der Algorithmus anhand des Modells einer Fischaugenkamera die Verzerrung aus. Wie in Abb. 4 zu sehen, werden beispielsweise die gekrümmten Linien des Schachbretts nach der Kalibrierung wieder gerade gemacht. Die Anzahl der benötigten Bilder hängt von der jeweiligen Kamera ab, eine große Sammlung von Fotos führt jedoch zu einer genaueren

Annäherung an die Parameter. (Zitat OPENCV) Die Verwendung von Bildern mit einer höheren Auflösung führt ebenfalls zu genaueren Parametern. (Zitat OPENCV/BLOG) Die angenäherten Parameter sind jedoch nur für die Kalibrierung von Bildern **genau**, die mit der gleichen Auflösung aufgenommen wurden wie die für die Kalibrierung verwendeten Bilder. Um die Kameramatrix für Bilder mit einer niedrigeren Auflösung zu verwenden, muss die intrinsische Kameramatrix K mit der folgenden Formel skaliert werden:

$$K_n = kK, \quad (3.4)$$

wobei k ein skalarer Wert ist, der den **Verkleinerungsfaktor** darstellt. Da es sich bei der intrinsischen Kameramatrix um eine affine Abbildung handelt, muss der Wert bei $K_{n3,3}$ auf 1 gesetzt werden. Die resultierende intrinsische Kameramatrix funktioniert bei kleineren Auflösungen, die dem **Aspektverhältnis** der für die Kalibrierung verwendeten Bilder gleich sind.

Ein Nachteil der Kamerakalibrierung ist, dass jedes kalibrierte Bild eine geringere Auflösung hat als das Originalbild. Dies ist eine Folge des Kalibrierungsprozesses, da dieser **eine Teilmenge des Bildes** verzerrt, insbesondere die Pixel in den Ecken des Bildes. Daher werden diese Pixel beim Kalibrierungsprozess aus dem resultierenden Bild entfernt. Um das Bild in seiner ursprünglichen Auflösung zu rekonstruieren, wird **ein Interpolator** verwendet. **Der Interpolator** liefert jedoch ein unscharfes Bild als das Original. Um dies zu kompensieren, empfiehlt es sich, eine Kamera mit hochauflösenden Bildern zu kalibrieren und dann Bilder mit dieser Auflösung aufzunehmen. Anstatt den Interpolator zu verwenden, verkleinern Sie die Bilder dann auf eine Auflösung, die für die jeweilige Anwendung erforderlich ist. Das Ergebnis ist ein genaueres Bild ohne Unschärfe. Leider ist dieser Prozess sehr rechenintensiv, und es wurde beschlossen, nur die Bilder aus dem Interpolator zu verwenden, um die Rechenleistung der Pipeline zu erhöhen.

Ein weiterer Nachteil der Kamerakalibrierung ist, dass sich der Mittelpunkt der Kamera **verschiebt**. Ein Beispiel dafür ist in Abb. 5 zu sehen. Durch manuelles Ändern von c_x kann der Bildmittelpunkt wieder an seine ursprüngliche Position verschoben werden.

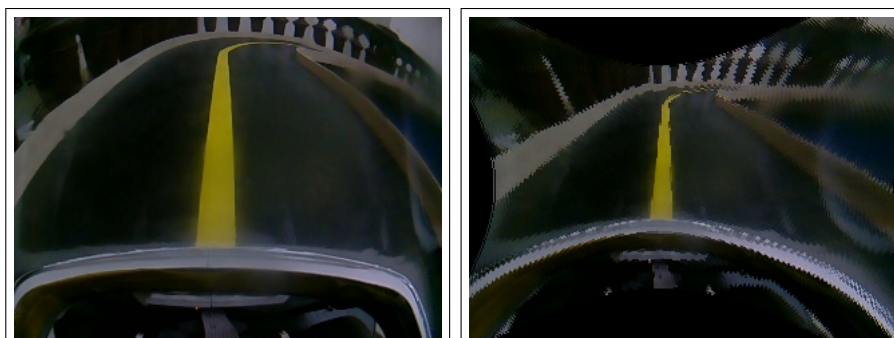


Abbildung 5 – **Shifted Images**.

3.2 „Color Thresholding“


Als Nächstes folgt die **Stufe** des sogenannten "Color Thresholding". In dieser Stufe werden alle Pixel aus dem Bild entfernt, **die nicht hellgelb sind**.

Zunächst wird das Bild von allen Pixeln ohne hohen Rotanteil gefiltert, da Hellgelb im Rot-Grün-Blau-Farbraum (RGB) einen hohen Rotanteil hat. Anschließend wird das Bild in den HSV-Farbraum (**Farbwert, Sättigung und Hellwert**) umgewandelt.



Im RGB-Farbraum ist reines Gelb so definiert, dass sowohl der rote als auch der grüne Kanal gleich sind und der blaue Kanal auf Null gesetzt ist. Dies lässt nur einen Freiheitsgrad für die Justierung der Pipeline zu. Ein einziger Freiheitsgrad führt zu Problemen bei der Abstimmung, da die Pipeline dadurch weniger in der Lage ist, Störungen zu berücksichtigen. Unterschiedliche Lichtverhältnisse oder reflektierende Oberflächen sind Beispiele für solche Störungen.

Im HSV-Farbraum **wählt** der Farbwertkanal die Farbe aus, der Sättigungskanal die Reinheit des Farbtons und der Wertekanal die Helligkeit des Farbtons. (MS ANNO) Nach der Auswahl des Farbwerts, in diesem Fall Gelb, werden der Helligkeits- und der Sättigungskanal verwendet, um den spezifischen Gelbton auszuwählen. Die Verwendung dieser beiden Kanäle ermöglicht eine zuverlässigere Erkennung der Farbe.

Um die **g**  Spur zu erkennen, filtert die Pipeline Farben außerhalb des gelben Farbtonbereichs heraus und schneidet dann den unteren Teil des Helligkeits- und Sättigungskanals ab. Das Ergebnis ist, dass nur reines Gelb im Bild übrig bleibt. Die in der Pipeline verwendeten Werte für Gelb werden durch den folgenden Bereich, TABLE, dargestellt.

Das Ergebnis dieser Stufe der Pipeline ist beispielsweise in Abb. 6 dargestellt.

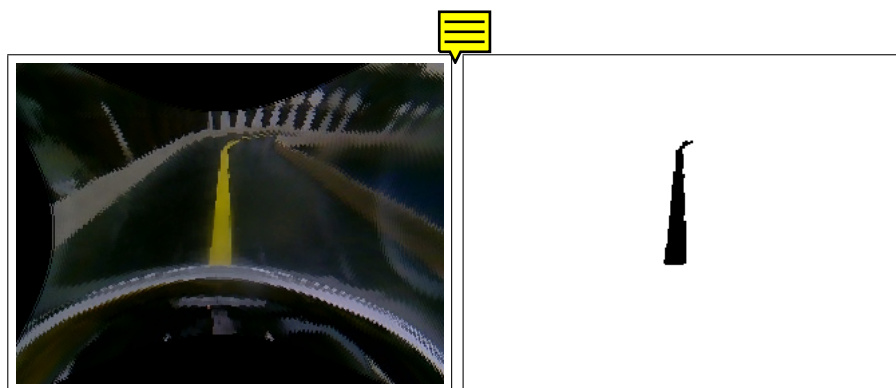


Abbildung 6 – **Thresholded image.**

3.3 Perspektivtransformation

Der dritte Teil der Bildverarbeitungspipeline ist die Perspektivtransformation.

Wenn ein Bild mit einer am Fahrzeug montierten Kamera aufgenommen wird, ist die **Fahrspurlinie** nicht rechteckig, sondern trapezförmig. Diese Perspektive erfordert, dass alle Berechnungen bezüglich der Fahrspurlinie **deren abnehmende** Breite kompensieren müssen. Um dies zu vermeiden, wird eine Perspektivtransformation durchgeführt.

Bei der Perspektivtransformation wird eine Teilmenge eines Bildes beschnitten (?) und so angepasst, dass sie das gesamte Bild umfasst. Ein Beispiel ist in Abb. 7 dargestellt.

Bei diesem Projekt ist die Form der Teilmenge des Bildes ein Trapez. Mithilfe dieser Perspektivtransformation wird die trapezförmige Form der Fahrbahnlinie in eine gerade Form korrigiert, wie in Abb. 7 dargestellt.

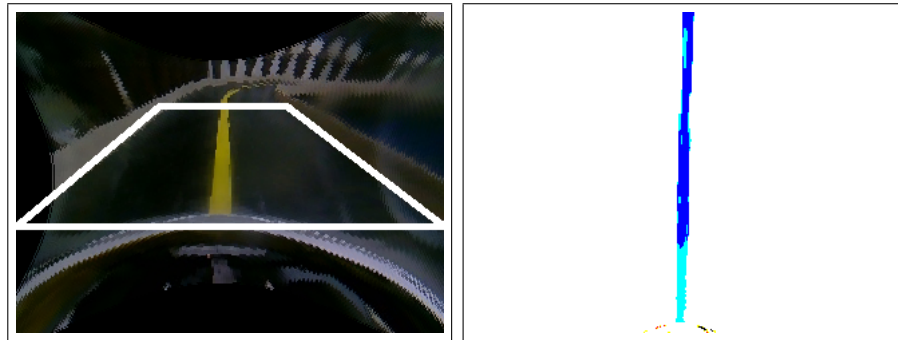


Abbildung 7 – ROI, Inverted Perspective Trans.

Eine Folge dieser neuen Perspektive ist, dass das resultierende Bild einer 2-D-Ebene der Strecke ähnelt. Die Perspektivtransformation vereinfacht auch die weitere Bildverarbeitung, da alle Objekte außerhalb des Trapezes abgeschnitten werden und nur die Spur übrig bleibt. Wie in Abb. 7 zu sehen ist, führt die Perspektivtransformation jedoch zu zusätzlichem Rauschen im Bild. Die durch die Perspektivtransformation verursachte **Scherung** (?) führt dazu, dass die Pixel des Originalbildes gestreckt werden, wodurch das Bild mit Rauschen verunreinigt wird. Daher ist es erforderlich, dass diese Stufe nach der „Color-Thresholding“-Stufe erfolgt.

3.4 Histogramm

Nach der Perspektivtransformation wird ein **Histogramm der Pixelanzahl** erstellt, um den Startpunkt für die Gleitfenstermethode zu bestimmen. Die Anzahl der weißen Pixel wird für jede Spalte des Bildes gezählt, und diese Zahlen werden in einer Liste gespeichert. Es wird davon ausgegangen, dass die Spalten mit den höchsten Zahlen

Fahrspurinformationen enthalten, und der Index der Spalte mit der höchsten Zahl wird als Startpunkt für die Gleitfenstermethode gewählt. Dieser Schritt trägt dazu bei, die Berechnungszeit zu verkürzen, indem Teile des Bildes herausgefiltert werden, die keine Fahrspurinformationen enthalten. Eine visuelle Darstellung des Histogramms ist in Abb. 8 zu sehen.

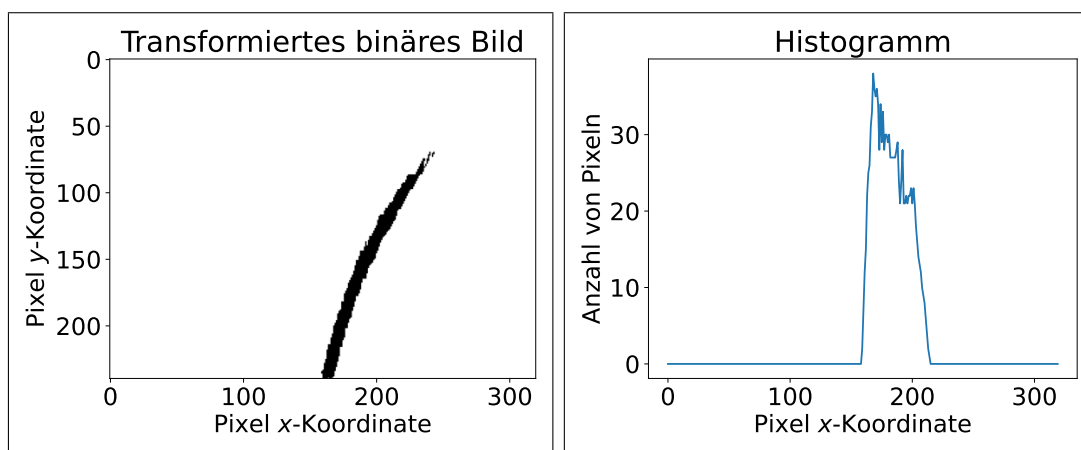


Abbildung 8 – Histogramm

3.5 Gleitfenstermethode

In der fünften Stufe der Pipeline werden die Richtung und der Versatz (?) der Fahrspur mit Hilfe der Gleitfenstermethode berechnet.

Zunächst wird eine ausgewählte Anzahl von Rechtecken (Fenstern) erstellt. Die Fenster werden in einer Spalte ausgerichtet und sind gleich groß. Ihre Höhe wird so gewählt, dass die Spalte die gesamte Höhe des Bildes überspannt, während ihre Breite um einen Faktor der Bildbreite gewählt wird. Die Anzahl der Fenster wird so gewählt, dass die Spalte die schwarzen Pixel umfasst, die die Fahrspur darstellen, (?) wie in Abb. 9 zu sehen ist. Wenn die Anzahl der Fenster zu groß ist, besteht die Gefahr, dass die Methode weiße Pixel, die tief im Bild liegen, fälschlicherweise als Teil der Fahrspur erkennt.

Zweitens wird die x-Koordinate des Mittelpunkts des ersten Fensters auf den Spitzenwert des Histogramms aus der vorherigen Stufe gesetzt. Dann wird die Anzahl der weißen Pixel innerhalb des Fensters gezählt. Wenn die Anzahl über einem gewählten Schwellenwert liegt, werden die Pixel zu einem Array hinzugefügt und der Mittelwert der x-Koordinaten der enthaltenen Pixel berechnet. (?) Die x-Koordinate des nächsten Fensters wird auf diesen berechneten Mittelwert gesetzt und über das erste Fenster gelegt. Dieser Vorgang wird dann für alle verbleibenden Fenster wiederholt. Wenn in einem Fenster keine Pixel enthalten sind, wird das Fenster ignoriert und das folgende Fenster darüber gelegt. Im

Bildkoordinatensystem liegt die linke obere Ecke des Bildes bei $(0, 0)$, wobei die positive x-Achse nach rechts und die positive y-Achse nach unten im Bild ansteigt. Die untere Kante des Bildes wäre also jede Koordinate, deren y-Komponente gleich der Höhe des Bildes ist. Aus der Sicht des Ingenieurs wären die Fenster von unten nach oben gestapelt, im Bildkoordinatenraum ist es jedoch umgekehrt.

Die Pixel, die nicht innerhalb der einzelnen Fenster liegen, werden dann aus dem Bild herausgefiltert. Das Bild vor und nach dem Filterungsprozess ist in Abb. 9 zu sehen. Mit Hilfe der Methode der kleinsten Quadrate wird ein Polynom durch die verbleibenden Pixel approximiert.

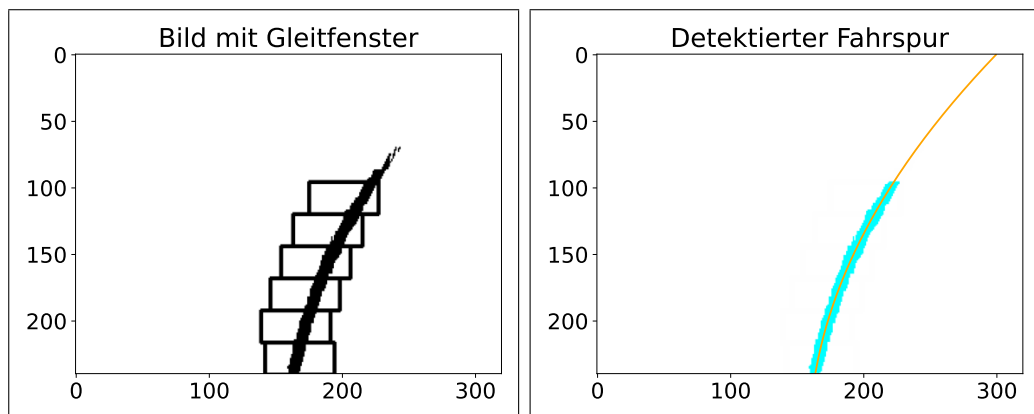


Abbildung 9 – Vor und Nach dem Filterungsprozess

3.6 Berechnung von Ausrichtung und Abstand

In der letzten Stufe der Pipeline werden die Ausrichtung und die Abweichung von der Fahrspurlinie berechnet.

Aus dem in der vorangegangenen Stufe berechneten angepassten Polynom wird eine analytische Ausrichtung an einem ausgewählten Punkt entlang der Bahn mit der folgenden Gleichung ermittelt,

$$\theta = \arctan(f'(y)). \quad (3.5)$$



Der inverse Tangens der Ableitung des Polynoms liefert die Ausrichtung an einem bestimmten Punkt. Für den Stanley-Regler wird nur die Differenz zwischen der Bahnausrichtung und der Ausrichtung des Fahrzeugs benötigt, da es kein globales Koordinatensystem für das Fahrzeug gibt; die Ausrichtung des Fahrzeugs wird so gewählt, dass sie zu jeder Zeit null Grad beträgt. In Abb. 10 ist das Koordinatensystem des Fahrzeugs dargestellt.

For Paul: I removed the paragraph here as I realized it was no longer necessary, the results of looking ahead into the future i can talk about in the results chapter

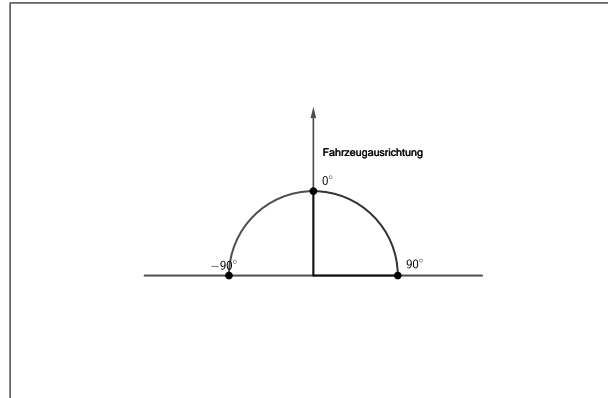


Abbildung 10 – For Paul: I know this looks strange, do you have any recommendations for programs where I can draw vectors?

Die Abweichung von der Fahrspurlinie wird aus dem angepassten Polynom mit folgender Funktion berechnet:

$$e_{fa} = \left(\frac{w}{2} - f(h)\right) \cdot x_{mpp}. \quad (3.6)$$

Dabei ist w die Breite des Bildes, $f(h)$ die Ausgabe des angepassten Polynoms am unteren Rand des Bildes, x_{mpp} eine Skalierungskonstante zur Umrechnung von Pixeln in Meter und e_{fa} der Abstand des Fahrzeugs zur Fahrspur. Zusammenfassend wird die Differenz aus der x-Komponente des Mittelpunkts und der x-Komponente des Polynompunkts am unteren Bildrand in Meter umgerechnet.

3.7 Ausreißer

Die in Algorithmen zur Fahrspurerkennung verwendete Gleitfenstermethode kann zu einem Phänomen führen, das als „Ausreißer“ bekannt ist. Diese treten auf, wenn der Algorithmus die Fahrspurlinie in einem bestimmten Bild falsch bestimmt. Abb. 11 zeigt eine Sequenz von drei Einzelbildern, wobei das ganz linke Einzelbild den Ausgangspunkt bildet. Im ersten Bild wird die Fahrspurlinie richtig erkannt, aber im zweiten Bild tritt ein Ausreißer auf, bei dem die Fahrspurlinie falsch bestimmt wird. Im letzten Bild wird die Fahrspurlinie wieder korrekt erkannt.

Eine der Hauptursachen für Ausreißer ist die falsche Auswahl der HSV-Werte. Wenn die im Algorithmus verwendeten HSV-Werte nicht restriktiv genug sind, werden Farben außer Gelb möglicherweise nicht herausgefiltert, was zu einer ungenauen Bestimmung der Fahrspurlinie führt.

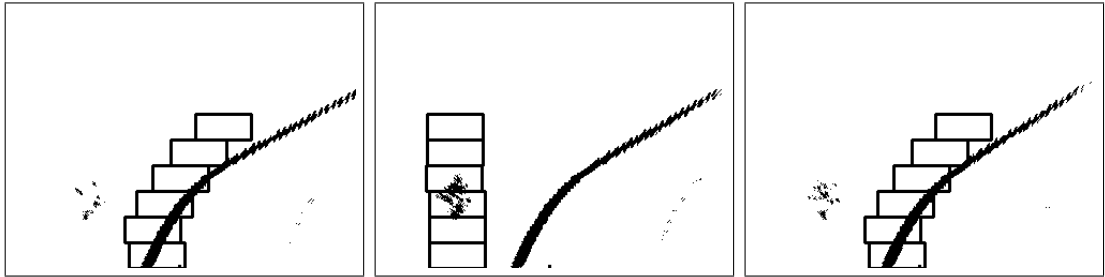


Abbildung 11 – Ausreißer

3.8 Pipeline-Optimierungen

Bei der Entwicklung der Bildverarbeitungspipeline wurde festgestellt, dass die Laufzeit der Pipeline **zu langsam** war. Um die Rechenzeit der Pipeline auf ein akzeptables Niveau zu verbessern, wurde eine Optimierung eingeführt. Weitere Optimierungen sind zwar möglich, würden aber den Rahmen der Arbeit sprengen und werden daher hier nicht behandelt.

Das Python-Programm cProfiler wurde verwendet, um die Funktionsaufrufe der Pipeline mit ihren jeweiligen Laufzeiten zu **tabellieren**. Da es sich bei der Pipeline um ein **deterministisches Programm handelt**, (?) verbessert die Zwischenspeicherung der Ausgabe der teuren Funktionsaufrufe im Speicher die Laufzeit der Pipeline. Daher wird bei allen nachfolgenden Pipeline-Ausführungen das zwischengespeicherte Ergebnis zurückgegeben, anstatt die Berechnung erneut auszuführen. Beispielsweise ist die Berechnung der perspektivischen Transformationsmatrizen kostspielig, was bei der Zwischenspeicherung zu einer Leistungssteigerung der Pipeline führt.

FOR PAUL: This next paragraph exists in the english version, but I removed it here as it no longer accurate due to our use of the corola brain. Do you think I should still have it?

In order to prove the performance improvement of the pipeline, an experiment was implemented. First, the pipeline is executed once and the runtime ignored as the performance improvement is only relevant to subsequent executions of the pipeline. Therefore, on the second execution of the pipeline, the runtime is measured for both pipeline variants, with and without caching the expensive function calls from the first execution. The experiment is then run 100 times and the result tabulated. The arithmetic mean of the runtimes for both variants are then compared to one another to measure the speed improvement. The values are in TABLE, showing that by caching the expensive function calls, the pipeline processing speed improved by VALUE

Kapitel 4

Experiment

4.1 Hardware

Die CoRoLa-Car-Plattform besteht aus einem Raspberry Pi, einer Fisheye-Kamera, einem Gleichstrommotor und einem Motorregler zur Steuerung der Fahrzeuggeschwindigkeit und einem Servomotor um den Lenkeinschlag des Fahrzeugs zu steuern. Der Raspberry Pi sendet PWM Signale an den Motorregler und den Servoregler, um das Fahrzeug zu steuern. das Fahrzeug zu steuern. Mit dem Robot Operating System 2 (ros2) wird der Raspberry Pi über das Netzwerk mit einem zentralen Computer verbunden. Der Stanley-Controller läuft auf dem Zentralrechner, der die Daten von der Kamera verarbeitet und die Ausgabe des Stanley-Reglers an das Fahrzeug sendet.

4.2 Experimentkriterien

4.2.1 Höchstgeschwindigkeit

Um die Höchstgeschwindigkeit zu messen, fährt das Fahrzeug mit einer bestimmten Vorwärtsgeschwindigkeit auf der Strecke. Wenn das Verhalten des Fahrzeugs als „akzeptabel“ angesehen wird, wird das Fahrzeug angehalten und die Geschwindigkeit erhöht. Dieser Vorgang wird so lange wiederholt, bis das Verhalten des Fahrzeugs nicht mehr „akzeptabel“ ist. Das „akzeptable“ Verhalten des Fahrzeugs wird empirisch anhand der folgenden beiden Kriterien definiert: Fähigkeit, der Fahrspur zu folgen, und Fehlen von beobachteten Schwingungen.

4.2.2 Maximal möglicher Fehlerwinkel

Die maximalen Fehlerwinkel sind definiert als die maximalen Winkel, bei denen das Fahrzeug in der Lage ist, den Weg links bzw. rechts der Fahrspurlinie wiederzufinden.

Um dies zu messen, wird das Fahrzeug bei deaktiviertem Motorregler kollinear mit der Fahrspurlinie platziert. Dann wird das Fahrzeug im Uhrzeigersinn gedreht, wobei der Ausgang des jeweiligen Reglers in Echtzeit angezeigt wird. Das Fahrzeug wird so lange gedreht, bis der Ausgang des Reglers entweder chaotisch wird oder konstant Null ist. Das Experiment wird dann für den Stanley-Regler und den PID-Regler durchgeführt. Das Auftreten eines chaotischen Ausgangs ist nicht deterministisch, empirisch gesehen ist dies jedoch der Fall, wenn das Fahrzeug die Fahrspurlinie nicht zuverlässig erkennen kann. Ein Ausgang von Null bedeutet auch, dass das Fahrzeug die Fahrspurlinie überhaupt nicht erkennen kann.

4.2.3 Abstand von der Linie

Die Messung des Fahrzeugversatzes erfolgt durch die folgenden zwei Versuche. Zunächst fährt das Fahrzeug mit einer bestimmten Vorwärtsgeschwindigkeit auf der Strecke. Dann wird mit einer Überwachungskamera ein Video von der Kurvenfahrt des Fahrzeugs aus der Vogelperspektive aufgenommen. Das Fahrzeug wird mit unterschiedlichen Vorwärtsgeschwindigkeiten aufgezeichnet. Bei der ersten Aufnahmerunde wird das Fahrzeug mit dem Stanley-Regler gesteuert, bei der zweiten mit dem PID-Regler. Dieses Experiment wird für die gewählten Geschwindigkeiten von 1,0, 1,25, 1,5, 1,75 und 2,0 Metern pro Sekunde durchgeführt.

4.2.4 Reglerausgang

Um den Reglerausgang der beiden Regler zu messen, wird das Fahrzeug angewiesen (?), mindestens dreimal um die Strecke zu fahren. Während der Fahrt wird der Ausgang des Reglers aufgenommen. Nach der Fahrt werden die Daten im Zeitverlauf grafisch dargestellt und eine Teilmenge der Daten ausgewählt. Die Teilmenge der Daten wird nach den folgenden Kriterien ausgewählt: Periodizität und Fehlen von Ausreißern.

4.3 Ergebnisse

Kapitel 5

Conclusion