Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Bachelorarbeit

**Bastian Meiners**

## Kamerabasierte Geschwindigkeitsmessung auf autonomen Fahrzeugen

*Fakultät Technik und Informatik*
*Studiendepartment Informatik*

*Faculty of Engineering and Computer Science*
*Department of Computer Science*

Bastian Meiners

# Kamerabasierte Geschwindigkeitsmessung auf autonomen Fahrzeugen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Stephan Pareigis
Zweitgutachter: Prof. Dr. Zhen Ru Dai

Eingereicht am: 12. April 2013

**Bastian Meiners**

**Thema der Arbeit**
Kamerabasierte Geschwindigkeitsmessung auf autonomen Fahrzeugen

**Stichworte**
FAUST, optische Geschwindigkeitsdetektion, Fahrstreifenerkennung, Bildverarbeitung, dynamische ROI, autonomes Fahrzeug, Fahrassistenz, Hallsensor, Carolo Cup

**Kurzzusammenfassung**
Dieses Dokument beschäftigt sich mit der Auswertung der aktuellen Geschwindigkeit eines autonomen Fahrzeugs mittels Bilddaten. Unter Zuhilfenahme einer auf dem Fahrzeug montierten Kamera werden markante Ortsmarken im Bild gespeichert und mit darauf folgenden Aufnahmen verglichen. Eine Software berechnet Weg und Zeit, die zwischen zwei Bildern zurückgelegt wurden, um die Geschwindigkeit des Fahrzeugs festzustellen. Zur Unterstützung wird ein Hall-Sensor verwendet.

**Bastian Meiners**

**Title of the paper**
Camera based speed measurement on autonomous vehicles

**Keywords**
optical speed detection, FAUST, lane recognition, image processung, dynamic ROI, autonomous vehicle, driving assistant, hall sensor Carolo Cup

**Abstract**
This thesis deals with the evaluation of the current speed of an autonomous vehicle using image data. With the aid of a camera that is installed on the car, distinctive spots in the picture will be saved and compared to following recordings. Distance and time covered within two images, will be calculated with a software to determine the speed of the car. A hall sensor is used in support.

# Inhaltsverzeichnis

# Einleitung

In der Automobilbranche nimmt die Automatisierung einen hohen Stellenwert ein. Heutige Fahrzeuge bieten immer ausgereiftere und innovativere technische Hilfsmittel, die das Fahren sicherer und komfortabler machen. Hierzu zählen auch Systeme wie automatische Erkennung von Straßenschildern und sensorische Einparkhilfen.

Die Messung der Geschwindigkeit in Fahrzeugen ist einer der wesentlichen Faktoren für den Straßenverkehr. Um so wichtiger werden die Geschwindigkeitsdaten, wenn das Fahrzeug autonom fährt und somit nicht auf menschliche Intuition bezüglich der Wahl einer angemessenen Geschwindigkeit vertrauen kann.

In aktuellen Fahrzeugen wird die Geschwindigkeit mittels Sensoren an den Rädern oder Achsen gemessen. Hierbei können jedoch Unregelmäßigkeiten auftreten, die fehlerhafte Werte verursachen. Als gutes Beispiel sind hier die Reifen eines Autos zu nennen. Diese unterliegen sowohl dem Verschleiß, als auch einem zuvor definierten Abrollumfang. Werden kleinere oder größere Räder montiert, oder ist der Abrieb der Reifen zu groß, ist eine Tachojustierung erforderlich. Ähnlich verhält es sich auf den Fahrzeugen des Projektes „Fahrassistenz und Autonome Systeme“ (FAUST).

Der Grund für den Bedarf eines Verfahrens zur kamerabasierten Geschwindigkeitsmessung ist, ungenaue Messdaten der Sensorik verhindern und Messfehler ausschließen zu können. Es werden durchdrehende oder blockierende Räder, nachlassende Motorleistung und andere abweichende physikalische Größen außer Acht gelassen und exakt die tatsächliche Geschwindigkeit des Fahrzeugs ermittelt.

Die Funktion des autonomen Fahrens im Projekt FAUST basiert auf der Erkennung der Fahrstreifen über eine am Fahrzeug angebrachte Kamera. Diese Bachelorarbeit beschreibt einen Algorithmus, der die Kameradaten nutzt, um die Geschwindigkeit des Fahrzeug über optische Signale zu errechnen. Das Verfahren hat die Eigenschaft, mittels Bildverarbeitung markante Punkte der Fahrbahn zu detektieren und zu nutzen, um Bewegung zu erkennen. Die räumliche Veränderung dieser Punkte über die Zeit ergibt die gefahrene Distanz und somit die Geschwindigkeit des Fahrzeugs.

Im Gegensatz zu herkömmlichen Verfahren zur Messung der Geschwindigkeit mit Sensoren werden mit dem vorgestellten Algorithmus keine elektrischen Signale zur Verarbeitung herangezogen. Die protokollierten Daten werden ausschließlich aus optischen Aufzeichnungen gewonnen.

Im Verlauf dieser Bachelorarbeit wird auf folgende Dinge eingegangen:

- **Kapitel 1 - Das Fahrzeug**
  Das erste Kapitel gibt einen Überblick über das Umfeld, das den Rahmen für diese Bachelor-Thesis schafft. Es wird die Fahrzeugplattform und dessen Spezifikationen sowie dessen Aufgabe vorgestellt. Des Weiteren werden hier relevante Fahrzeugkomponenten analysiert und der Bedarf einer optischen Geschwindigkeitserkennung begründet.
- **Kapitel 2 - Bildverarbeitung**
  In diesem Kapitel wird auf die optischen Komponenten des in Kapitel 1 vorgestellten Fahrzeugs eingegangen. Die für diese Arbeit erforderlichen Bilddaten werden analysiert und deren Verarbeitungsmöglichkeiten vorgestellt.
- **Kapitel 3 - Geschwindigkeitsanalyse**
  In dem Hauptkapitel werden Konzepte vorgestellt, die zur Detektion der Fahrzeugbewegung und zur Feststellung der Geschwindigkeit umgesetzt wurden. Es werden Voraussetzungen sowie hinführende Algorithmen erläutert, die für die Entwicklung unablässig waren. Außerdem werden Verbesserungs- und Einsatzmöglichkeiten vorgestellt.
- **Kapitel 4 - Testergebnisse**
  Alle Ergebnisse der Arbeit sind in diesem Kapitel zu finden. Es wird eine Analyse der Genauigkeit des entwickelten Algorithmus vorgestellt.
- **Kapitel 5 - Zusammenfassung und Aussichten**
  Das Abschlusskapitel stellt den Nutzen und die Einsatzmöglichkeiten des entwickelten Algorithmus vor. Des Weiteren wird hier der Ablauf der Arbeit analysiert.

# 1. Das Fahrzeug

Die Entwicklung der *optischen Geschwindigkeitsmessung auf autonomen Fahrzeugen* findet auf der Plattform „Saphir“ statt. Hierbei handelt es sich um ein elektrisch angetriebenes Fahrzeug im Maßstab 1:10, welches unter der Bezeichnung TT-01 des Herstellers Tamiya (Tamiya, 2013) im regulären Handel erhältlich ist. Es besticht durch eine sehr leichte Bauweise aus Aluminium und Carbon und weist einen tiefen Schwerpunkt auf. Letzteres ist z.B. für schnelle Kurvenfahrten von Vorteil.

Im Rahmen des Projektes FAUST wurde zusätzlich eine eigens konstruierte Ebene aus Polycarbonat angefertigt, die als Plattform für elektronische Bauteile dient. So wurden hier unter anderem ein Mikrocontroller zur „Steuerung der Aktorik und Auswertung der Sensorik“ (FAUST, 2013), ein Pico-PC zur Verarbeitung von Algorithmen, sowie elf Infrarot-Sensoren in selbst entwickelten Halterungen verbaut.

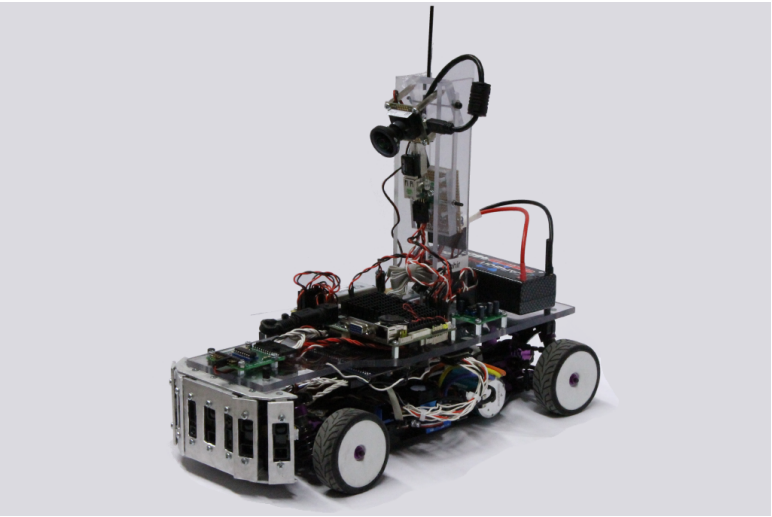

Abbildung 1.1.: Das Versuchsfahrzeug Saphir

Um die entwickelte und installierte Elektronik zu verdecken und zu schützen, wurde die Karosserie eines Ford Shelby GT-350 verwendet. Das Schwester-Fahrzeug „Diamond“, das in allen Aufbauten „Saphir“ gleicht, wurde mit einer Karosserie eines Audi RS8 ausgestattet.

## 1.1. Carolo Cup

Neue Entwicklungen für die Fahrzeuge des FAUST-Projektes können jährlich in einem Wettbewerb auf die Probe gestellt werden:
Der mit einem Preisgeld dotierte „Carolo-Cup“ der TU Braunschweig (Carolo-Cup, 2013) bietet den Studenten verschiedener Hochschulen die Möglichkeit, die Funktion und Präzision des eigenen autonomen Fahrzeugs in verschiedenen Disziplinen zu demonstrieren. Hierbei sind die Fähigkeiten in den Kategorien

- Spurführung und Ausweichen
- Einparken
- Befahren eines Rundkurses mit und ohne Hindernissen

vorzustellen. Die Ergebnisse werden einem Fachpublikum, bestehend aus Mitarbeitern aus Wirtschaft und Vertretern von Automobilherstellern, vorgestellt und von diesen bewertet.

Das Regelwerk des Cups beinhaltet eine Spezifikation für die Fahrbahn:
*„Es gibt lediglich eine Straße. Bei dieser Straße handelt es sich um die Nachbildung einer Landstraße, bestehend aus langen Geraden, scharfen Kurven und Kreuzungen. Die Straße ist konstant 820 mm breit und an den Rändern mit durchgezogenen Linien abgegrenzt. Die zwei Fahrstreifen werden durch eine gestrichelte Mittellinie geteilt. Alle Linien sind weiß und ca. 20 mm breit. Die Mittellinie ist alle 200 mm durch eine 200 mm Lücke unterbrochen.“* ((Carolo-Cup, 2013), Regelwerk).

Die Fahrzeuge des FAUST-Projekts, die an dem Carolo-Cup teilnehmen, sind nach den vorgegebenen Spezifikationen konstruiert. Um die Funktionen exakt testen zu können, wurde in den Laboren der HAW Hamburg eine Teststrecke nach oben beschriebenen Regeln nachgebaut.
Im Rahmen dieser Bachelorarbeit sind besonders die Angaben, die die Mittellinie betreffen, von großer Wichtigkeit. Diese bildet die Grundvoraussetzung für eine korrekte optische Messung der Geschwindigkeit.

## 1.2. Technische Besonderheiten

Die Fahrzeugplattform wurde, wie zu Beginn dieses Kapitels beschrieben, aufgrund spezieller technischer Spezifikationen ausgewählt und verfügt über eigens angefertigte Modifikationen. Zu den Besonderheiten, die bereits beim Kauf des Grundgerüstes des Fahrzeugs eine wichtige Rolle spielten, zählt ein Allradantrieb. Der starke Elektromotor treibt ein Ritzel an, durch das die Antriebskraft über einen Zahnriemen gleichmäßig an die Vorder- und Hinterachse verteilt wird.
An diesem Ritzel wurden manuell 20 Neodym-Magnete in identischem Abstand befestigt, welche mittels eines Hall-Sensors detektiert werden (vgl. Abbildung 1.2). Die Anzahl der Ticks, die der Sensor in einer bestimmten Zeitspanne ermittelt, dient als Grundlage für die aktuelle sensorische Geschwindigkeitsmessung.

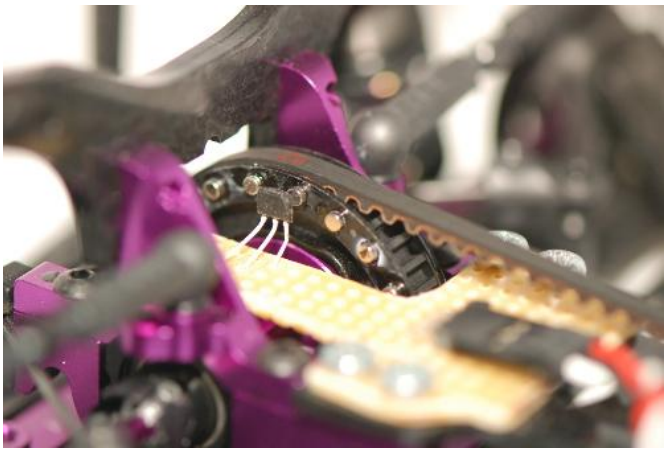

Abbildung 1.2.: Hall-Sensor (FAUST, 2013)

Die Befestigung der Magnete per Hand wirkt sich auf die Genauigkeit der Messwerte aus. Eine Analyse der sensorischen Fehlerabweichung wird im folgenden Kapitel durchgeführt.

## 1.3. Geschwindigkeits- /Fehleranalyse

In diesem Abschnitt wird die Abweichung der tatsächlich gefahrenen Geschwindigkeit gegenüber der theoretischen, mit Sensoren gemessenen bestimmt. Wie bereits bei Hensel (2012) (S. 36), werden die Sensorwerte auf einer vorgegebenen Streckenlänge gemessen, um Abweichungen gegenüber Theoretischen Werten darstellen zu können. Differenzen entstehen

hierbei durch mechanische Einwirkungen, wie z.B. die manuelle Montage der Magnete des Hall-Sensors.

Der Radumfang der Räder beträgt 207 Millimeter. Eine Umdrehung der Antriebsscheibe bei einer Übersetzung von 1:1 (d.h. eine Radumdrehung entspricht genau einer Umdrehung des Antriebsritzels) beträgt 20 Impulse. Somit wird mit einem Impuls eine Wegstrecke von 10,35 mm zurückgelegt.

$$\frac{207\ mm\ Radumfang}{20\ Impulse} = 10,35\ mm\ pro\ Impuls \tag{1.1}$$

An einer Versuchsreihe wird festgestellt, ob die errechneten Werte mit den tatsächlich gemessenen Werten übereinstimmen. Das Fahrzeug wird auf einer zuvor exakt abgemessenen und markierten Strecke gerade über den Boden bewegt. Durch das Wiederholen der Testdurchführung kann ein Mittelwert der festgestellten Ergebnisse errechnet werden.
Der Versuch wurde für die Strecke von 1 Meter und für 5 Meter durchgeführt (vgl. Tabelle 1.1 und 1.2). Hierbei wird für jede Messung der aktuelle Wert des Hall-Sensors in seiner Funktion als Inkrementalgeber notiert und das Fahrzeug entlang der festgelegten Strecke bewegt. Der Endwert des Sensors wird vermerkt und die Differenz der Werte errechnet.
Um die festgestellten Werte analysieren zu können, ist es notwendig, die Anzahl der Impulse des Sensors zu kennen, die rechnerisch auf einer bestimmten Strecke auftreten. Bei einer Wegstrecke von 1 m gilt:

$$1\ m = 1000\ mm \rightarrow \frac{1000\ mm}{10,35\ mm\ pro\ Impuls} = 96,6 \approx 97\ Impulse \tag{1.2}$$

Bei einer Strecke von 5 Metern tritt demnach die 5-fache Impulszahl auf:

$$5\ m = 5000\ mm \rightarrow \frac{5000\ mm}{10,35\ mm\ pro\ Impuls} = 483,1 \approx 483\ Impulse \tag{1.3}$$

Messung für die Distanz **1 Meter**:

| **Messung** | **Startwert** | **Endwert** | **Differenz** |
|---|---|---|---|
| 1 | 2 | 103 | **101** |
| 2 | 109 | 210 | **101** |
| 3 | 308 | 409 | **101** |
| 4 | 507 | 608 | **101** |
| Durchschnitt | - | - | **101** |

Tabelle 1.1.: Distanzmessung (1 Meter)

Messung für die Distanz **5 Meter**:

| **Messung** | **Startwert** | **Endwert** | **Differenz** |
|---|---|---|---|
| 1 | 711 | 1222 | **511** |
| 2 | 1876 | 2385 | **509** |
| 3 | 32 | 540 | **508** |
| 4 | 1324 | 1835 | **511** |
| 6 | 2370 | 2882 | **512** |
| 7 | 3417 | 3926 | **509** |
| Durchschnitt | - | - | **510** |

Tabelle 1.2.: Distanzmessung (5 Meter)

Bei einer Wegstrecke von 1 Meter tritt eine Differenz von 4 Impulsen gegenüber der errechneten Impulszahl auf. Dies entspricht einem Fehler von 4,124% (vgl. Formel 1.4) gegenüber der tatsächlich gefahrenen Geschwindigkeit bzw. Wegstrecke. Diese Ungenauigkeit wird von der Messreihe auf 5 Meter bestätigt: Hier beträgt die Impulsdifferenz durchschnittlich 27 Impulse, was im Verhältnis zu der gefahrenen Strecke einem Fehler von 5,176% entspricht (vgl. Formel 1.5).

Fehlerberechnung für 1 Meter:

$$\frac{97}{100\%} = \frac{4}{x} \rightarrow x = 4,124\% \tag{1.4}$$

Fehlerberechnung für 5 Meter:

$$\frac{483}{100\%} = \frac{27}{x} \rightarrow x = 5,590\% \tag{1.5}$$

Wird diese Fehleranalyse auf die gefahrene Geschwindigkeit angewendet so ergibt sich ein Unterschied von circa 5 % zu der tatsächlich gefahrenen Geschwindigkeit.

# 2. Bildverarbeitung

Die Nutzung optischer Systeme im Rahmen dieser Bachelorarbeit impliziert die Verwendung von Bildverarbeitungsalgorithmen. Hierbei werden die zur Verfügung gestellten Bilddaten so modifiziert, dass nutzbare Informationen zur weiteren Verarbeitung extrahiert werden können.

Die freie Programmbibliothek „Open Source Computer Vision Library“ (openCV) (OpenCV, 2013) ist hierbei von großem Nutzen. Sie „verfügt über mehr als 2500 optimierte Algorithmen“, mit denen unter Anderem optische Merkmale aus Bildern gefiltert werden können. Außerdem ist es mit dieser Bibliothek möglich Merkmale durch Einzeichnung in das Bild kenntlich zu machen.

Bereits in vorhergehenden Bachelor- und Master-Thesen wurde diese Bibliothek genutzt, um Bildmerkmale zu analysieren. So wurde auch in der Bachelor-Thesis von Andreas Liedtke die „Realisierung eines robusten Verfahrens zur Identifikation von Positionsmarken unter Verwendung von Bildverarbeitung in Videobildern“ (Liedtke, 2011) mittles in openCV enthaltenen SURF- (Speeded Up Robust Features) und SIFT- (Scale-Invariant Feature Transform) Algorithmen realisiert.

Eine weitere Entwicklung für die autonomen Fahrzeuge des FAUST-Projektes unter dem Aspekt der Bildverarbeitung, geht aus der Master-Thesis von Eike Jenning (Jenning, 2008) hervor. Diese beschäftigt sich mit der „Systemidentifikation eines autonomen Fahrzeugs mit einer robusten, kamerabasierten Fahrspurerkennung in Echtzeit“. Dabei wird „die Fahrspur durch ein kubisches Polynom approximiert und anschließend die Fahrzeuglage in der Fahrspur, der tangentiale Steuerkurswinkel des Fahrzeugs zur Fahrspur und der Kurvenradius der Fahrspur identifiziert“. Die daraus entstandene Fahrspurerkennung POLARIS wird im Rahmen des FAUST-Projektes weiterhin genutzt und weiterentwickelt.

## 2.1. Umwandlung von Bilddaten

Aufgrund kamera- und linsenspezifischer Gegebenheiten ist es notwendig, die Bilddaten, vor Weitergabe an einen Algorithmus, zu korrigieren. Hierbei ist es nötig, Kenntnis über den genutzten Bildsensor, das verwendete Objektiv und deren Spezifikationen zu haben. Auch der Montage-Winkel der Kamera auf dem Fahrzeug spielt eine große Rolle.
All diese Einflüsse müssen bei der Einrichtung und Kalibrierung der Kamera beachtet werden. Zwei hierbei äußerst relevante Faktoren werden im Folgenden erläutert.

### 2.1.1. Korrektur der Linsenverzeichnung

Die Linsenverzeichnung ist die Verzerrung eines aufgenommenen Bildes. „Bei weit-winkligen Objektiven“, so wie sie auf den autonomen Fahrzeugen des FAUST-Projektes genutzt werden, „ist die radiale Linsenverzeichnung oftmals schon mit bloßem Auge zu erkennen”(Jacobi, 2009). Hierbei sind „kissen-“ oder „tonnenförmige“ Verzerrungen des Bildes zu sehen und Geraden werden als Kurven oder krumme Linien dargestellt. Dabei nimmt die Krümmung mit dem Abstand von der Bildmitte zu. Abbildung 2.1 zeigt den Unterschied zwischen einem verzeichneten und einem korrigierten Bild.

Die Master-Thesis von E. Jenning (Jenning (2008) (S. 23)) beschäftigt sich ebenfalls mit dieser Problematik. In dieser wird ein Weg vorgestellt, wie die Verzeichnung zu korrigieren ist, um korrekte Bilddaten zu erhalten. Hierbei werden „durch eine einmalige Kalibrierung der Kamera die, zur Verzeichnungskorrektur benötigten, internen Kameraparameter bestimmt“.

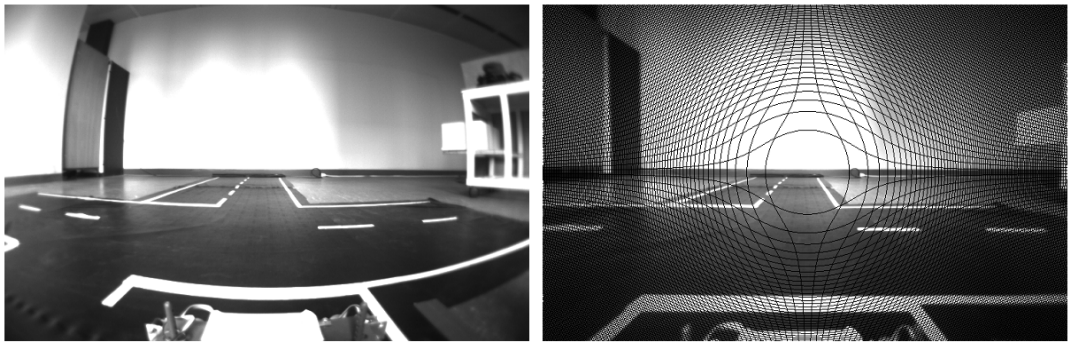

Abbildung 2.1.: Verzeichniskorrektur (Jenning, 2008)

Für die optische Geschwindigkeitserkennung ist die Korrektur von Bilddaten von hohem Stellenwert. Bestimmte Pixelwerte werden dem Bild entnommen und zur weiteren Verarbeitung gespeichert. Verzeichnete Bilddaten können hier also zu Fehlern führen.

### 2.1.2. Bildtransformation

Da die Kamera in einem bestimmten Winkel auf dem Fahrzeug montiert ist, liegt die 2-dimensionale Sensor-Ebene der Kamera anders im Raum, als die Ebene der Fahrbahn (siehe (Jenning, 2008) S. 26). Die Abbildung 2.2 skizziert diese Problematik.

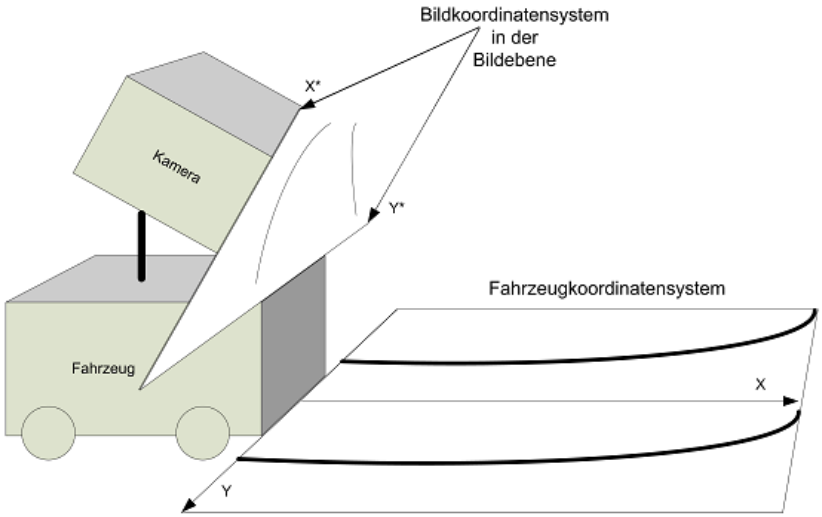


Abbildung 2.2.: Kameramontage (Jenning, 2008)

Für die korrekte Weitergabe der Bilddaten ist es notwendig, dass diese veränderte Projektion auf den Sensor so transformiert wird, dass Entfernungs- und Größenangaben, die aus den Bilddaten gewonnen werden, korrekt vorliegen. Auch die projektive Transformation wurde im Rahmen der Master-Thesis von E. Jenning (Jenning, 2008) behandelt und gehört zur Basis der Bildverarbeitung auf den autonomen Fahrzeugen des FAUST-Projektes.

Konkret bedeutet das für diese Bachelorarbeit, dass Objekte, die im „hinteren“ Teil des Bildes liegen, kleiner erscheinen, als Objekte mit der selben Größe im „vorderen“ Abschnitt. In der Abbildung 2.3 erkennt man, dass ein Mittelliniensegment nahe dem Fahrzeug länger erscheint, als eines, dass weiter vom Fahrzeug entfernt liegt. Die projektive Transformation der Bilddaten

sorgt dafür, dass die Längenmessung des selben Segmentes an unterschiedlichen Bildpositionen den selben Wert ergibt.

Abbildung 2.3.: verzeichnetes Kamerabild

# 3. Geschwindigkeitsanalyse

Um die Bewegung eines Objektes optisch feststellen zu können, ist es notwendig, Veränderungen der Umgebung zu erfassen und auszuwerten. Die Geschwindigkeit an sich besteht aus der Änderung der Ortsposition über die Zeit. Diese Änderung kann auf unterschiedliche Weise festgestellt werden:

- mechanisch
- sensorisch
- optisch

Während die Fahrzeugindustrie die ersten beiden Punkte bereits nutzt (beispielsweise mit Wellen zur Drehzahlmessung oder Sensoren zur Geschwindigkeitsmessung), beschäftigt sich diese Bachelorarbeit mir der optischen Geschwindigkeitserkennung. Hierbei wird die Bewegung mittels optischer Merkmale ausgewertet.
Die Kamera des Versuchsfahrzeugs liefert Bilddaten, in denen per Bildverarbeitung brauchbare von unbrauchbaren Informationen getrennt werden. Die nutzbaren Daten werden dann zur weiteren Analyse an Verarbeitungsalgorithmen weitergegeben und zur Auswertung der Geschwindigkeit und Steuerung des Fahrzeuges genutzt.

## 3.1. Optische Merkmalsextraktion

Die Konzeption und Entwicklung eines Algorithmus zur optischen Kantendetektion basiert auf den vom Fahrzeug bereitgestellten Bilddaten.

Es werden die Grauwerte nebeneinander liegender Pixel miteinander verglichen. In welcher Form genau hierbei gesucht wird, ist in Kapitel 3.1.2 beschrieben.
In den aufgenommenen Kamerabildern haben die Bildpunkte der weißen Linien der genormten Fahrbahn einen hohen, die des schwarzen Fahrbahnuntergrunds jedoch einen niedrigen Grauwert. Eine große Differenz zweier Grauwerte wird somit als Linienkante gedeutet und die Pixelkoordinaten der gefundenen Kante gespeichert.

### 3.1.1. Position des Fahrzeugs auf der Fahrbahn

Bei der Auswertung der Bilddaten werden folgende Voraussetzungen angenommen:

1. das Fahrzeug befindet sich auf der genormten Fahrbahn (vgl. Kapitel 1.1 - Carolo Cup)
2. das Fahrzeug hält sich zu jedem Zeitpunkt auf der rechten Fahrspur auf
3. die Mittelliniensegmente sind in der Mitte der Bildes positioniert
4. die Mittelliniensegmente laufen dicht neben der linken Fahrzeugseite vorbei

Nachfolgend ist eine Abbildung dargestellt, die die notwendige Position verdeutlicht:

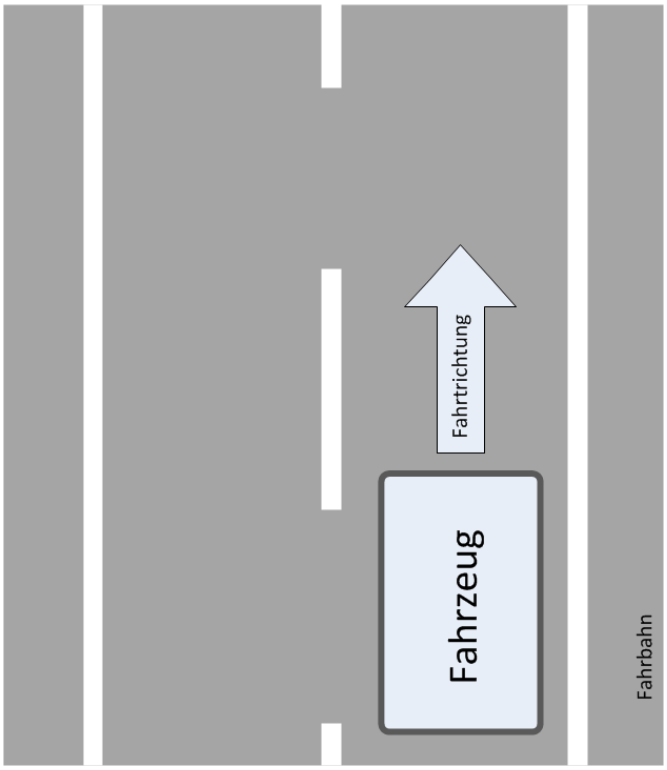


Abbildung 3.1.: optimale Fahrzeugposition

Ist einer der oben genannten Punkte nicht erfüllt, so kann im Rahmen dieser Bachelorarbeit nicht gewährleistet werden, dass die Geschwindigkeit des Fahrzeugs richtig bestimmt wird.

### 3.1.2. ROIs nutzen

*Regions of Interest* (ROI) sind ein wichtiges Hilfsmittel für die Extraktion relevanter Bilddaten zur Erstellung des Algorithmus zur optischen Geschwindigkeitsfeststellung. Mit ihnen wird ein bestimmter Bereich im Bild festgelegt, in dem nach definierten Strukturen gesucht wird. Im Rahmen dieser Bachelorarbeit werden die Übergänge schwarzer und weißer Bildbereiche gesucht. Es werden demnach ROIs an die Positionen im Bild gelegt, in denen sich weiße Mittelliniensegmente befinden - also im vertikalen Zentrum des Bildes. Die Abstände der ROI ergeben sich aus der späteren Parametrisierung des Algorithmus.

In dem Bild wird abhängig von der Fahrzeugposition ein eindimensionaler ROI in das Bild gelegt (vgl. Abbildung 3.2). Dieser besteht aus einem vertikalen Bereich, der nur einen Pixel hoch, und eine definierte Anzahl von Pixeln breit ist. Auf ihm wird ein Algorithmus zum Detektieren von schwarz-weiß-Übergängen angewendet. Diese „Scan-Linie“ wird pixelweise von links nach rechts durchlaufen, wobei der Grauwert jedes Pixels mit dem Grauwert des folgenden Pixels verglichen wird. Sollte der Algorithmus einen Unterschied der Werte feststellen, so wird weiterhin geprüft, ob in kurzer Distanz eine weitere Differenz festzustellen ist. Ist dies der Fall, wird die Position zwischen den festgestellten Pixeln als Linienmittelpunkt gespeichert.

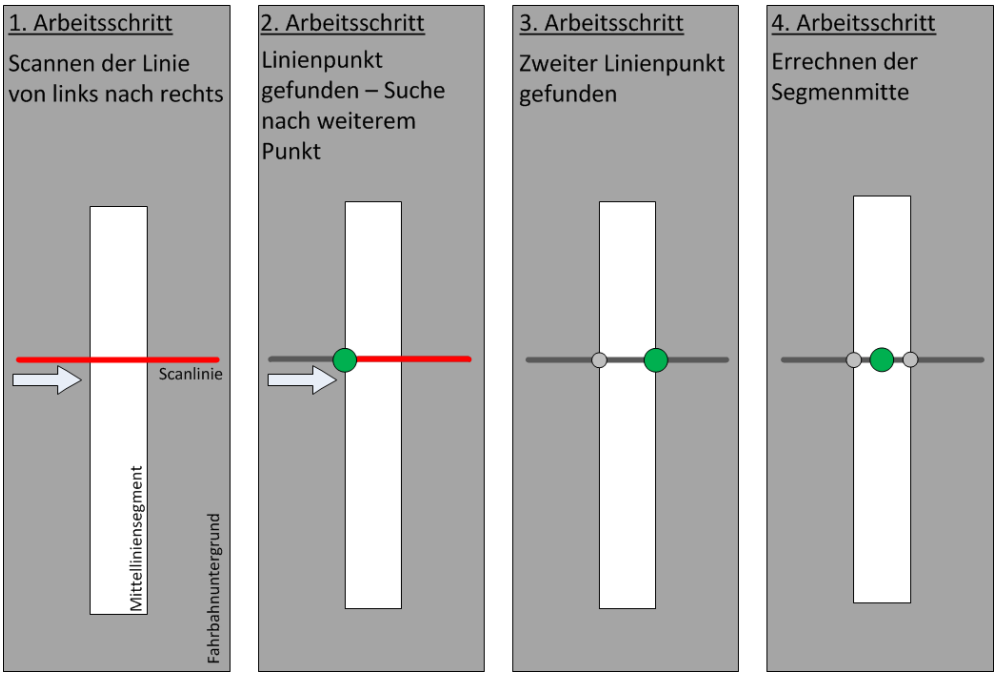


Abbildung 3.2.: Scan-Algorithmus

## 3.2. Algorithmus zur Kantenfindung

Der Algorithmus soll mittels markanter und sich in gleicher Weise wiederholender Punkte die Bewegung des Fahrzeugs ermitteln. Die einzigen Merkmale, die hierfür auf der genormten Fahrbahn in Frage kommen, sind die kurzen Enden der Mittelliniensegmente, da diese stets einen Abstand von 200 mm haben (vgl. Kapitel 1.1, Carolo-Cup Regelwerk) und immer im selben Bildbereich liegen (siehe Markierungen in Abbildung 3.3).

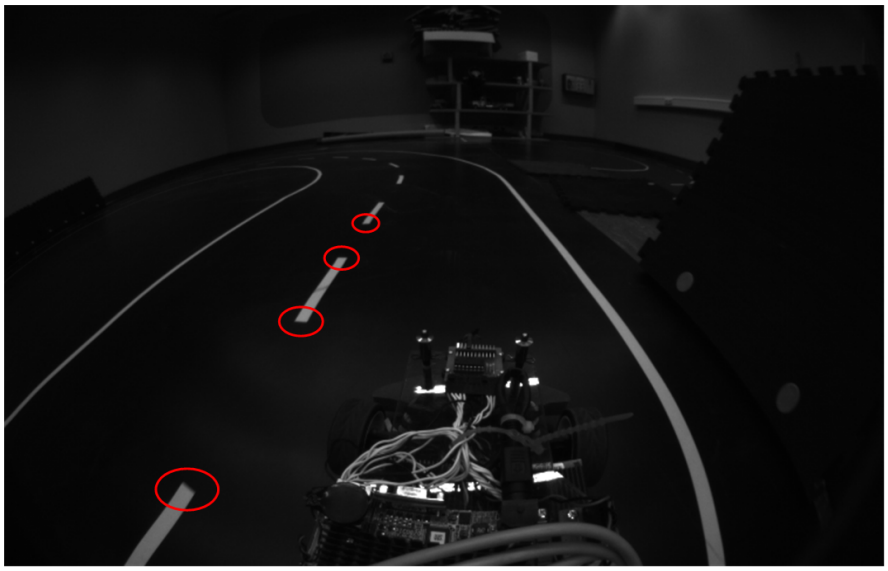

Abbildung 3.3.: kurze Linienkanten

Im Folgenden wird die Entwicklung des Algorithmus erläutert, der zur Detektion der o.g. kurzen Linienkanten und der daraus resultierenden Geschwindigkeitsfeststellung führt. Hierbei wurde der Ansatz von E. Hensel aus Kapitel 2.4 („Ermittlung der Geschwindigkeit mittels Bildverarbeitung“) seiner Master-Thesis „Kamerabasierte Fahrspuridentifikation und -repräsentation unter Berücksichtigung eines kinematischen Bewegungsmodells“ (Hensel, 2012) berücksichtigt.

Da die Mittelliniensegmente aus perspektivischen Gründen nicht senkrecht, sondern in einem bestimmten Winkel im Bild liegen, kommt folgender Ansatz zum Tragen: Unter Zuhilfenahme vertikaler ROI (vgl. Abbildung 3.2) kann ein Punkt auf einem Segment detektiert werden. Führt

man diesen Algorithmus ein weiteres Mal auf dem selben Mittelliniensegment, aber an einer leicht veränderten Position aus, so erhält man zwei Punkte, die das Segment markieren. Die beiden Punkte können nun miteinander verbunden werden; die entstehende Gerade liegt direkt auf der Mittellinie und wird als neuer ROI betrachtet. Da dieser neue ROI jedoch nur die beiden Punkte verbindet, wird nachfolgend erläutert, wie er verlängert wird, um die kurzen Kanten des Segments zu schneiden.

Die folgende Nomenklatur stellt die genutzten Variablen für die Berechnung der o.g. Verlängerung des neuen ROI dar.

| **Variable** | **Erläuterung** |
|---|---|
| FirstPoint ($x_f$, $y_f$) | durch vertikalen ROI ermittelter Linienpunkt, der vorne auf dem Mittelliniensegment liegt; $x_f$ und $y_f$ stellen die Punkt-Koordinaten dar |
| SecondPoint ($x_s$, $y_s$) | durch vertikalen ROI ermittelter Linienpunkt, der hinten auf dem Mittelliniensegment liegt; $x_s$ und $y_s$ stellen die Punkt-Koordinaten dar |
| straight$_a$ | Länge der Ankathete des Punktes *FirstPoint* im Steigungsdreieck |
| straight$_b$ | Länge der Ankathete des Punktes *SecondPoint* im Steigungsdreieck |
| $y$ | Gerade im Steigungsdreieck, die *FirstPoint* und *SecondPoint* verbindet |
| $\beta$ | Winkel der Geraden $y$ zu *straight_a* |
| m | Steigung der Geraden $y$ |
| b | Schnittpunkt der Geraden $y$ mit der y-Achse |
| add_a | Wert, um den *straight_a* verlängert wird (horizontale Erweiterung) |
| add_b | Wert, um den *straight_b* verlängert wird (vertikale Erweiterung) |
| newFirstPoint ($x_{nf}$, $y_{nf}$) | neue Position des Punktes *FirstPoint*, die durch Erweiterung von *straight_a* und *straight_b* entsteht; $x_{nf}$ und $y_{nf}$ stellen die Punkt-Koordinaten dar |
| newSecondPoint ($x_{ns}$, $y_{ns}$ | neue Position des Punktes *SecondPoint*, die durch Erweiterung von *straight_a* und *straight_b* entsteht; $x_{ns}$ und $y_{ns}$ stellen die Punk-Koordinaten dar |
| edge1, edge2 | Positionen der gefundenen kurzen Kanten eines Mittelliniensegments |

Tabelle 3.1.: Nomenklatur

In der nachfolgenden Vergrößerung des Bildausschnitts, der die horizontalen Scanlinien und detektierten Punkte auf dem Mittelliniensegment zeigt, wurde ein Steigungsdreieck eingezeichnet, mit dem verdeutlicht wird, wie die Formel zum Erstellen des neuen ROI entsteht.

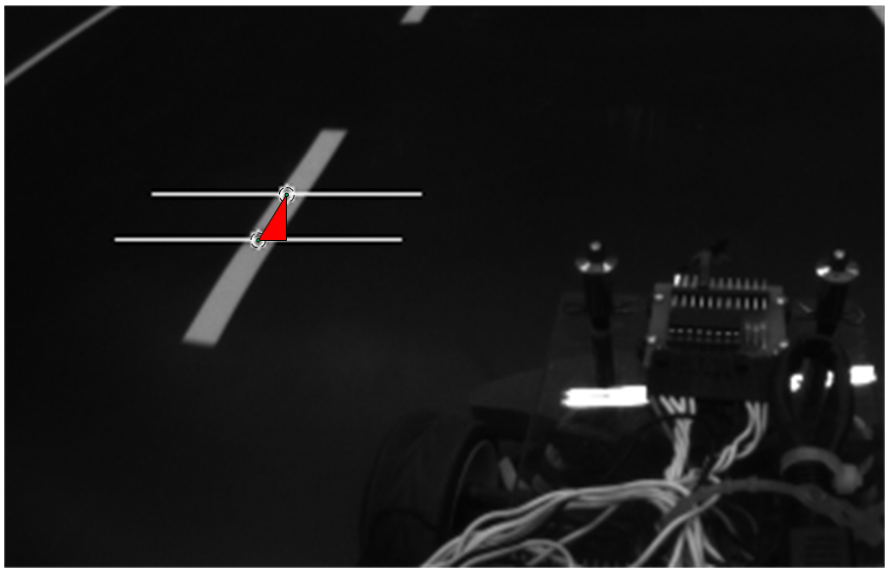

Abbildung 3.4.: Steigungsdreieck im Bild

In der folgenden Abbildung ist die schematische Darstellung dieses Dreiecks skizziert.

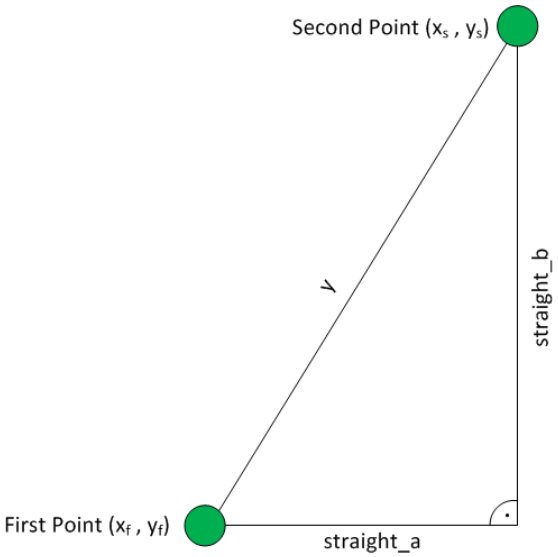


Abbildung 3.5.: Skizziertes Steigungsdreieck

Die Gerade $y$, die den neuen ROI darstellt, verbindet die beiden Punkte *First Point* und *Second Point*. Sie kann mittels der Geradengleichung

$$y = mx + b \tag{3.1}$$

dargestellt werden. Hierbei werden die Steigung $m$ und der y-Achsen-Abschnitt $b$ im späteren Verlauf der Algorithmus-Entwicklung ermittelt (vgl. Formel 3.14 und 3.15).
Da das entstandene Steigungsdreieck rechtwinklig ist, kann die Länge der beiden Katheten *straight_a* und *straight_b* aus der Differenz der jeweiligen $x$- beziehungsweise $y$-Koordinaten der beiden Punkte bestimmt werden:

$$straight\_a = x_f - x_s \tag{3.2}$$

$$straight\_b = y_f - y_s \tag{3.3}$$

Diese werden für die weitere Verwendung bei der Berechnung benötigt.

Damit die Gerade $y$ über das Ende der Mittellinie hinausgeht und somit die kurzen Kanten des Segments schneidet, muss sie um einen bestimmten Faktor verlängert werden.

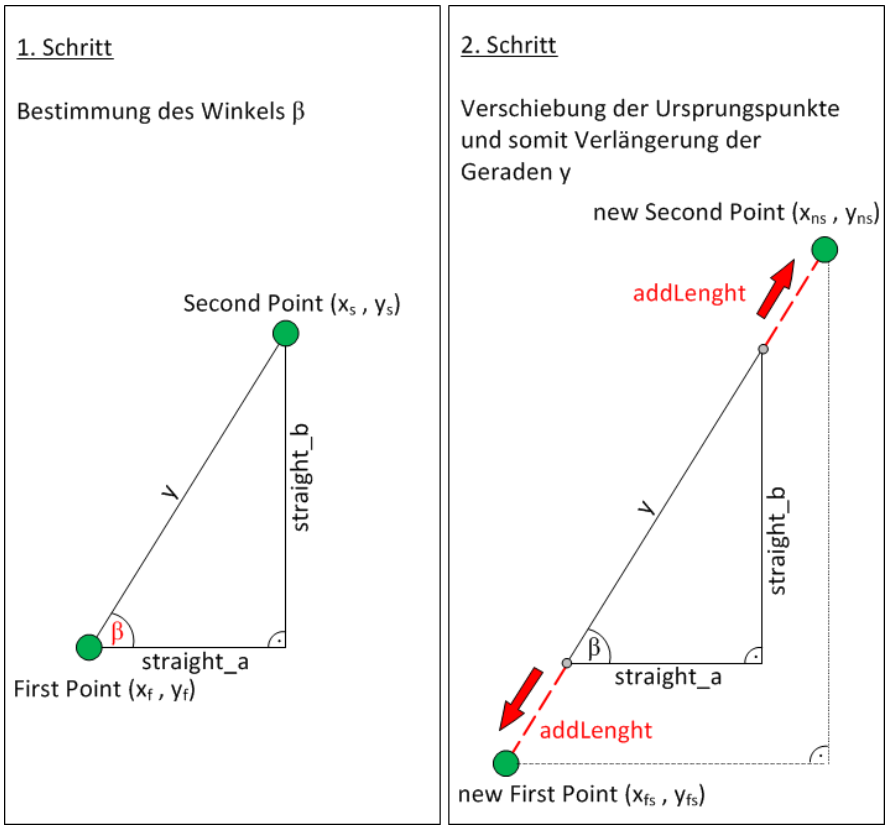


Abbildung 3.6.: Neue Punkte festlegen

Um die Strecke $y$ verlängern zu können, ist es zunächst erforderlich, die beiden Punkte *FirstPoint* $(x_f, y_f)$ und *SecondPoint* $(x_s, y_s)$ um einen definierten Abstand und eine bestimmte Richtung zu verschieben. Damit die versetzten Punkte hiernach weiterhin auf der Geraden $y$ liegen, muss deren Winkel im 2-dimensionalen Raum ermittelt werden. Im **1. Schritt** der Abbildung 3.6 ist zu erkennen, dass dies mittels Trigonometrie möglich ist. Somit ist der Tangens des Winkels $\beta$ das Verhältnis der Längen von Ankathete und Gegenkathete:

$$\tan(\beta) = \frac{straight\_b}{straight\_a} \tag{3.4}$$

Stellt man diese Formel nun um, so erhält man den Winkel $\beta$:

$$\beta = \arctan\left(\frac{straight\_b}{straight\_a}\right) \tag{3.5}$$

Ein zuvor definierter Wert *addLength* gibt die Länge der Verschiebung der Punkte und somit die Verlängerung der Geraden $y$ in Pixeln an (Abbildung 3.6, **2. Schritt**).

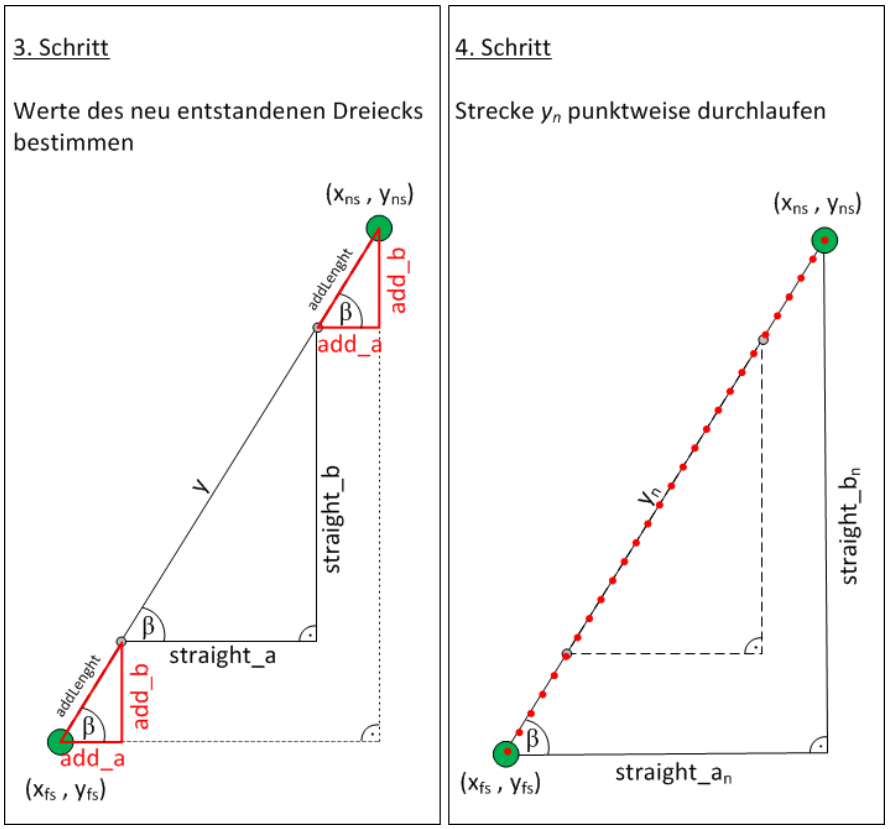


Abbildung 3.7.: Verlängerung berechnen

Unter Zuhilfenahme des errechneten Winkels β und der Strecke *addLength* lassen sich die Längen der Katheten *add_a* und *add_b* eines neuen rechtwinkligen Dreiecks bestimmen (**3. Schritt**), welches entsteht, wenn die Punkte *FirstPoint* $(x_f, y_f)$ und *SecondPoint* $(x_s, y_s)$ verschoben werden. Auch hier können die Werte unter Zuhilfenahme der Trigonometrie bestimmt werden. Da die Länge *addLength* und der Winkel β bekannt sind, sind folgende Zusammenhänge von Bedeutung: Das Verhältnis von Gegenkathete zu Hypotenuse ist der Cosinus des Winkels β, das von Ankathete zu Hypotenuse bestimmt den Sinus von β (vgl. Gleichung 3.6 und 3.7).

$$\cos(\beta) = \frac{add\_a}{addLength} \tag{3.6}$$

$$\sin(\beta) = \frac{add\_b}{addLength} \tag{3.7}$$

Stellt man die Formeln 3.6 und 3.7 um, so erhält man jeweils die Länge der Unbekannten *add_a* und *add_b*:

$$add\_a = \cos(\beta) \cdot addLength \tag{3.8}$$

$$add\_b = \sin(\beta) \cdot addLength \tag{3.9}$$

Mit den berechneten Längen von *add_a* und *add_b* lassen sich die $x$- und $y$-Werte der neuen Punkte *newFirstPoint* ($x_{nf}$, $y_{nf}$) und *newSecondPoint* ($x_{ns}$, $y_{ns}$) im Koordinatensystem berechnen. Da die Ursprungspunkte jeweils in eine andere Richtung verschoben werden, müssen für den Punkt *newFirstPoint* die neuen Kathetenlängen subtrahiert, für *newSecondPoint* jedoch addiert werden.

$$x_{nf} = x_f - add\_a \quad \text{und} \quad y_{nf} = y_f - add\_b \tag{3.10}$$

$$x_{ns} = x_s + add\_a \quad \text{und} \quad y_{ns} = y_s + add\_b \tag{3.11}$$

Somit ergeben sich für die Seiten $straight\_a_n$ und $straight\_b_n$ (3.5, **4. Schritt**) die neuen Längen

$$straight\_a_n = straight\_a + add\_a \tag{3.12}$$

$$straight\_b_n = straight\_b + add\_b \tag{3.13}$$

Wurde die Verlängerung berechnet, kann die ermittelte Gerade $y_n$ als neuer ROI betrachtet und der Algorithmus zur Ermittlung der Grauwertdifferenzen angewendet werden, um die kurzen Kanten zu detektieren. Wie in Kapitel 3.1 (Optische Merkmalsextraktion) beschrieben, ist auch hier ein wiederum herausstechendes Merkmal, dass das Liniensegment einen hohen Grauwert, der Untergrund jedoch einen sehr niedrigen Grauwert besitzt. In der folgenden Abbildung wurden mittels openCV die vertikalen Scanlinien und der neu berechnete ROI eingezeichnet.

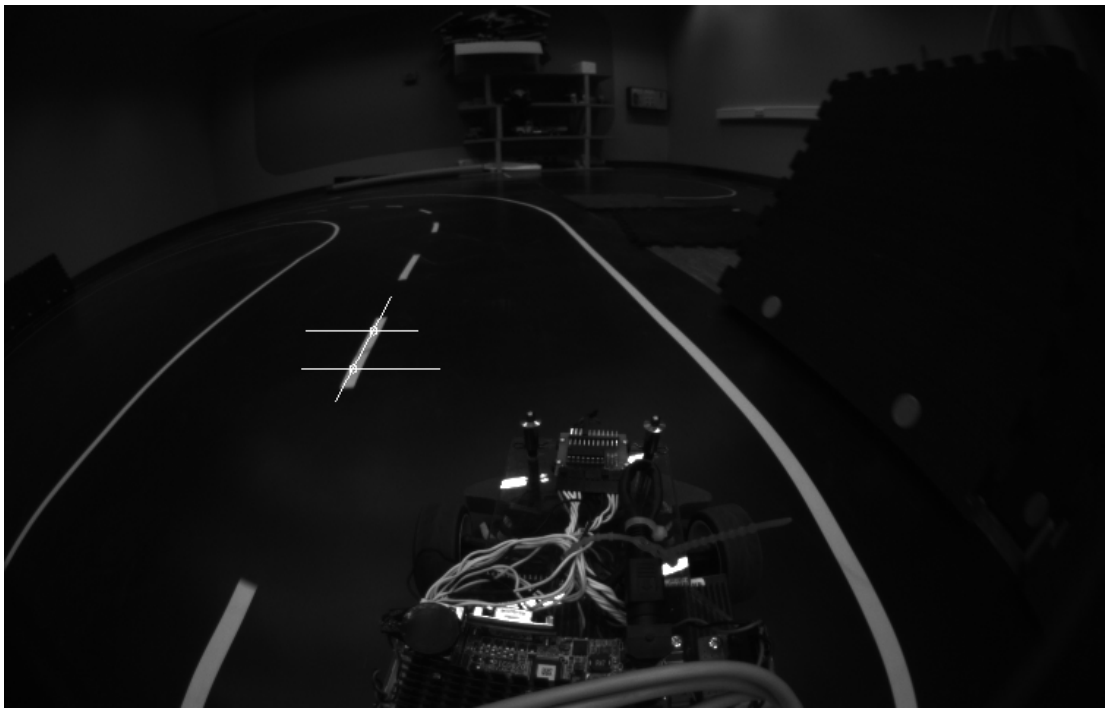

Abbildung 3.8.: Scanlinien eingezeichnet

Der **4. Schritt** der Abbildung 3.7 skizziert den Algorithmus, der durchgeführt wird, um o.g. Grauwertunterschiede im Bereich der neuen Scanlinie $y_n$ festzustellen. Beginnend von Punkt *newFirstPoint* ($x_{nf}$, $y_{nf}$), wird die Linie Pixel für Pixel durchlaufen. Dabei wird, wie bereits bei der Detektion der Mittellinien-Segmente, der jeweils vorherige Grauwert mit dem Aktuellen verglichen. Tritt eine Differenz der Werte auf, die eine definierte Schwelle überschreitet, wird die Koordinate des aktuellen Pixel als gefundener Kantenpunkt gespeichert.

Im Unterschied zu der horizontalen Erkennung der Mittellinie, bei der der $y$-Wert gleich bleibt, muss hier nicht nur in $x$-Richtung gesucht werden, sondern der neue ROI $y_n$ in einem Winkel $\beta$ durchlaufen werden. Dies bedeutet, dass zu jedem Schritt in $x$-Richtung ein neuer $y$-Wert errechnet werden muss.

Da eine Winkelangabe für die Berechnung von $x$- und $y$-Koordinaten im Koordinatensystem nicht ohne größeren Rechenaufwand zu verwenden ist, werden die Variablen der Geradengleichung 3.1 ($y = mx + b$) genutzt. Die Steigung $m$ der Geraden $y_n$ entspricht dem Verhältnis von

*straight_b*$_n$ zu *straight_a*$_n$.

$$m = \frac{straight\_b_n}{straight\_a_n} \tag{3.14}$$

Des Weiteren ist der $y$-Achsen-Abschnitt $b$ durch Umstellung der Geradengleichung 3.1 zu ermitteln. Die Steigung $m$ ist nun bekannt und für $x$ und $y$ können die Werte von einem der bekannten Punkte *newFirstPoint* ($x_{nf}$, $y_{nf}$) oder *newSecondPoint* ($x_{ns}$, $y_{ns}$) eingesetzt werden.

$$b = y_{nf} - m \cdot x_{nf} \quad \text{oder} \quad b = y_{ns} - m \cdot x_{ns} \tag{3.15}$$

Nachdem alle nötigen Parameter berechnet wurden, kann die neue Scanlinie nun pixelweise durchlaufen werden. Hierbei wird zunächst der aktuelle $x$-Wert um einen Schritt erhöht und gespeichert (vgl. Gleichung 3.16). Um den neuen $y$-Wert zu errechnen, wird dieser dann in die Geradengleichung (vgl. Gleichung 3.1) eingesetzt und als *nextY* gespeichert.

$$nextX = x_{nf} + 1 \tag{3.16}$$

$$nextY = m \cdot (nextX) + b \tag{3.17}$$

Dies geschieht so lange, bis der Wert von *nextY* dem Wert $y_{ns}$ des Punktes *newSecondPoint* gleicht und somit der gesamte neue ROI durchlaufen wurde. Die gefundenen Punkte werden als *edge1* und *edge2* gespeichert.

In der folgenden Tabelle ist die Veränderung der genutzten Variablen im Verlauf der Algorithmus-Entwicklung dargestellt:

| # | Input | Output | Nutzung |
|---|---|---|---|
| 1 | Ursprungskoordinate für Scanlinie | FirstPoint ($x_f$, $y_f$)<br>SecondPoint ($x_x$, $y_s$) | Erstellung der horizontalen ROIs, um Mittelliniensegment zu detektieren |
| 2 | FirstPoint ($x_f$, $y_f$)<br>SecondPoint ($x_x$, $y_s$) | straight_a<br>straight_b | Bestimmung der Katheten des entstandenen Steigungsdreiecks |
| 3 | straight_a<br>straight_b | $\beta$<br>m<br>b | Berechnung von Winkel, Steigung und y-Achsen-Abschnitt der Steigungs-Geraden |
| 4 | addLength<br>$\beta$ | add_a<br>add_b | Feststellung der Werte für die Erweiterung des Steigungsdreicks |
| 5 | FirstPoint ($x_f$, $y_f$)<br>SecondPoint ($x_x$, $y_s$)<br>add_a<br>add_b | newFirstPoint ($x_{nf}$, $y_{nf}$)<br>newSecondPoint ($x_{nx}$, $y_{ns}$) | Bestimmung der neuen Werte für *FirstPoint* und *SecondPoint* |
| 6 | newFirstPoint ($x_{nf}$, $y_{nf}$)<br>newSecondPoint ($x_{nx}$, $y_{ns}$)<br>m<br>b | edge1<br>edge2 | Strecke zwischen *newFirstPoint* und *newSecondPoint* gilt als neuer ROI; kurze Kanten werden gespeichert |

Tabelle 3.2.: Veränderung der genutzten Variablen

Zum besseren Verständnis der Nutzung der in Tabelle 3.2 genannten Daten ist nachfolgend der Datenfluss grafisch veranschaulicht:

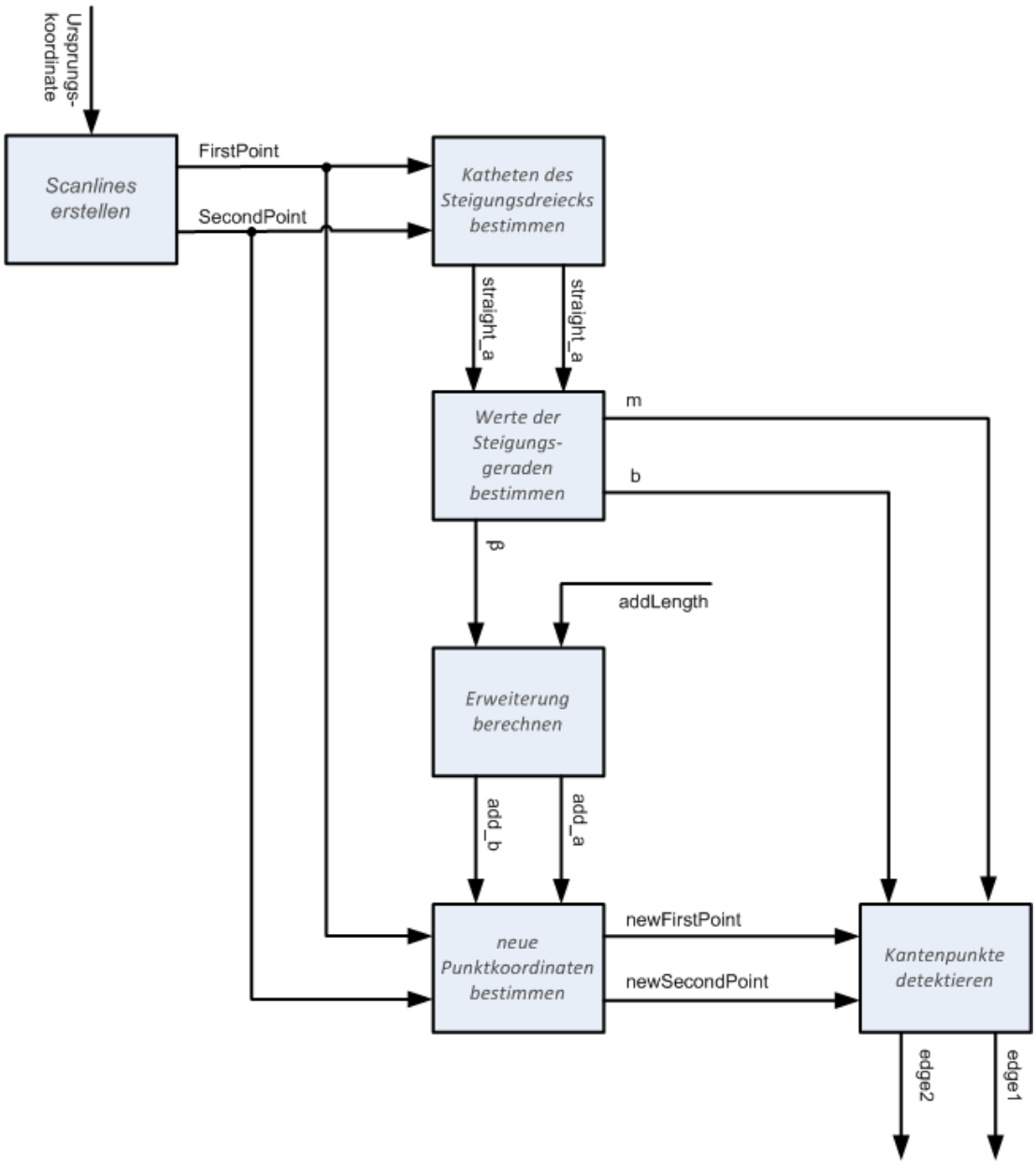


Abbildung 3.9.: Verarbeitungsablauf

### Laufzeitverbesserung

Der vorher beschriebene Algorithmus zu Detektion der kurzen Linienkanten (vgl. Kapitel 3.2) wird, wie nachfolgend beschrieben, nach Hensel (vgl. Hensel (2012)) erweitert:
Anstatt die komplette Linie vom ersten zum zweiten Punkt durchgängig zu scannen (Abb. 3.10), bietet sich als **2. Variante** die Möglichkeit, aus dem gesamten ROI zwei Bereiche zu schaffen, die nur das vordere und das hintere Ende der Mittellinie scannen, da die Erstellung der Punkte *FirstPoint* $(x_f, y_f)$ und *SecondPoint* $(x_s, y_s)$ bereits die Erkennung eines durchgängigen Mittelliniensegments impliziert und zwischen diesen Punkten keine kurzen Kanten existieren können. Somit wird der Rechenaufwand für die Kantenerkennung reduziert.
Ein weiterer Vorteil ist, dass die gespeicherten Positionen der kurzen Segmentkanten hierdurch besser auseinander gehalten werden können. Es werden die Formeln 3.16 und 3.17 zum Durchlaufen der Scanlinie nur auf die Bereiche zwischen *FirstPoint* und *newFirstPoint* beziehungsweise *SecondPoint* und *new SecondPoint* beschränkt.

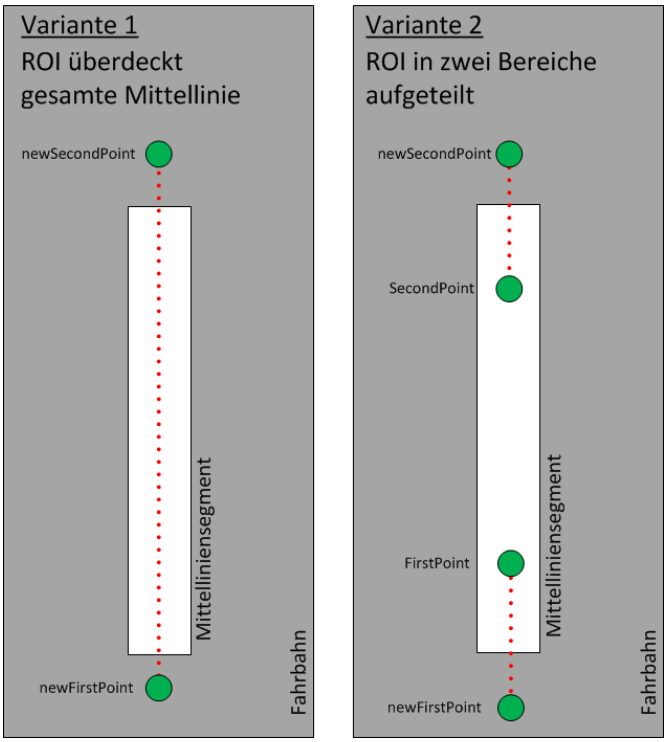


Abbildung 3.10.: Scan-Varianten

Die Detektion der kurzen Kanten wird nun in einem definierten Abstand auf einer im Bild weiter entfernten Mittellinie nochmals durchgeführt und die gefundenen Punkte gespeichert. Während sich das Fahrzeug fortbewegt, ändern sich die Koordinaten der kurzen Linienpunkte im Bild. Aufgrund dieser Verschiebung kann die Geschwindigkeit des Fahrzeugs ermittelt werden. Hierzu werden die Koordinaten der jeweils gespeicherten Punkte, sowie deren Detektionszeitpunkt, miteinander verglichen und aus der Differenz die gefahrene Strecke und die vergangene Zeit ermittelt. Mittels des Weg-Zeit-Verhältnisses kann somit die Geschwindigkeit berechnet werden, mit der sich das Versuchsfahrzeug bewegt.

#### Komplexität

Der Algorithmus zur Kantenfindung wird im Programmablauf für jedes einzelne erstellte horizontale ROI-Paar aufgerufen. Im Verlauf des Algorithmus werden die einzelnen Parameter zum Erstellen und Durchlaufen des ROI neu berechnet. Da hierfür jeweils die selbe Berechnung für $m$, $b$, $\beta$, $straight_a$ und $straight_b$ (vgl. Tabelle 3.1) durchgeführt wird, liegt die Komplexität bei $\mathcal{O}(1)$.
Im weiteren Verlauf der Funktion wird in einer Schleife der jeweilige neu erstellte ROI pixelweise durchlaufen. Die Komplexität hierfür ist $\mathcal{O}(n)$, da dieser Schleifenaufruf nicht weiter verschachtelt ist.

Die Schrankenfunktion für den Algorithmus zur optischen Kantendetektion ist also linear und liegt somit bei $\mathcal{O}(n)$.

## 3.3. Konzept zur Feststellung der Geschwindigkeit

Die Geschwindigkeit eines Fahrzeugs wird ermittelt, indem zwei Messungen hintereinander an einem hierfür vorgesehenen Sensor durchgeführt werden. Aus der Relation zwischen der Messwertdifferenz und der für die Messung benötigten Zeit kann die Geschwindigkeit ermittelt werden. Dabei ist es notwendig, Kenntnis über die gefahrene Strecke zu haben, die sich auf eine Messeinheit auswirkt. Als Beispiel ist hier der in Kapitel 1.3 analysierte Hall-Sensor zu nennen. Da Radumfang und Anzahl der Magnete bekannt sind, kann die Wegstrecke errechnet werden, die einen Impuls auslöst (vgl. Gleichung 1.1).

Ähnlich verhält es sich mit der optischen Geschwindigkeitsmessung. Mit der entwickelten Erkennung der kurzen Kanten eines Mittelliniensegments ist es möglich, eine Veränderung der Bilddaten über die Zeit festzustellen.

Jedes neue Bild, das von der Kamera an die Software übergeben wird, wird mittels des Algorithmus zur Kantenfindung (vgl. Kap. 3.2) analysiert und gefundene Kantenpunkte werden gespeichert. Bei der Speicherung der Koordinaten einer Kante wird gleichzeitig ein Zeitstempel gesetzt, der der Kante zugeordnet wird. Sobald das nächste Bild auf die selbe Art untersucht wurde, kann die Differenz von Koordinaten und Zeitstempeln eines Kantenpunktes genutzt werden, um die Geschwindigkeit des Fahrzeugs zu ermitteln (vgl. Abbildung 3.11).
Die Strecke zwischen den beiden Punkten wird hierbei mit dem Satz des Pythagoras festgestellt. Wie schon bei der Ermittlung des neuen ROI (vgl. Kapitel 3.2) kann die Verschiebung des Punktes im Koordinatensystem durch ein rechtwinkliges Dreieck dargestellt werden. Mit dem o.g. Satz ist es somit möglich, die Streckendifferenz $\Delta$s der Punkte *edge1* $(x_{e1}, y_{e1})$ und *edge2* $(x_{e2}, y_{e2})$ zu bestimmen:

$$\Delta s = \sqrt{(y_{e2} - y_{e1})^2 + (x_{e2} - x_{e1})^2} \tag{3.18}$$

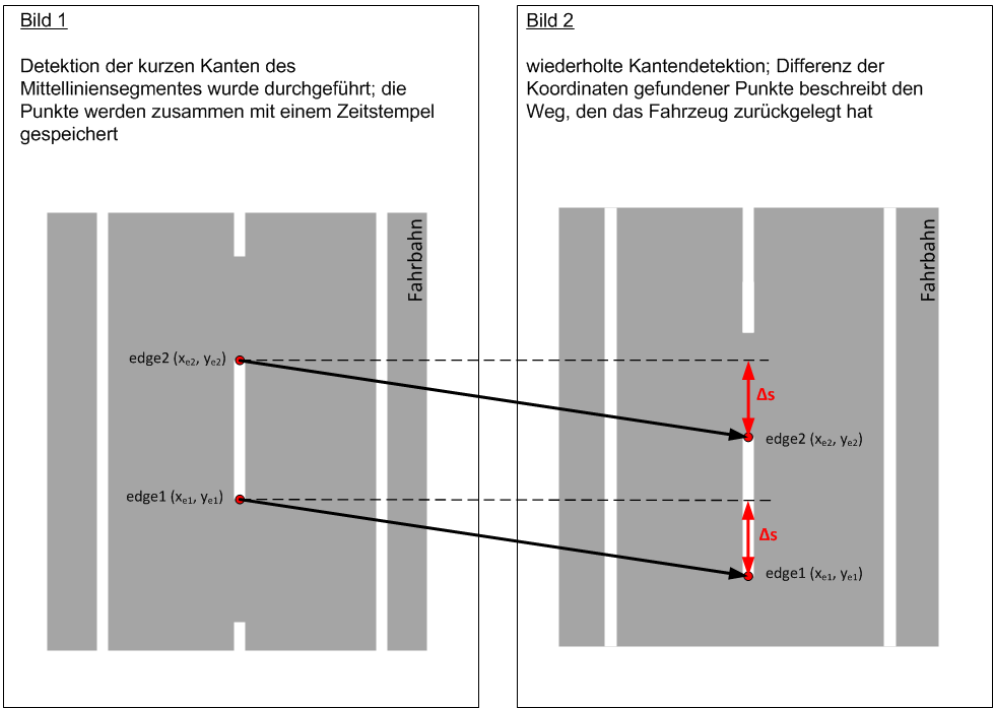


Abbildung 3.11.: Zurückgelegte Strecke bestimmen

Aufgrund der festgestellten Streckendifferenz $\Delta$s (in cm) und der Differenz der Zeitstempel (in ms) kann nun das Verhältnis von Strecke zu Zeit als Geschwindigkeit, also in **cm/s** oder auch **km/h** ausgedrückt werden.

#### Voraussetzungen

Für die korrekte Berechnung ist ausschlaggebend, dass die Werte folgende Richtlinien einhalten:

- Bilddaten wurden durch eine kalibrierte Kamera erstellt
- Linsenverzeichnung wurde herausgerechnet
- projektive Transformation wurde berechnet

Eine falsche oder ungenaue Kalibrierung der Kamera verfälscht die Berechnung der Geschwindigkeit, ähnlich wie die in der Einleitung erwähnte Abnutzung der Reifen für einen Hall-Sensor. Die Entzerrung der Linsenverzeichnung spielt gerade in Zusammenhang mit sog. „Fisheye-Objektiven“, welche im FAUST-Projekt genutzt werden, eine große Rolle. Diese ermöglichen zwar einen großen Sichtbereich, verfälschen aber die Bilddarstellung der realen Welt ungemein. Deswegen ist das Herausrechnen dieser Verzerrung für die Ermittlung korrekter Bilddaten unabdingbar (vgl. Kapitel 2.1.1).
Aufgrund perspektivischer Gegebenheiten ist eine Berechnung der Verschiebung der Punkte unter ebenfalls Berücksichtigung der projektiven Transformation nötig. Das nicht Beachten hätte zur Folge, dass Veränderungen im „hinteren“ Bildbereich eine geringere Streckenlänge ergeben würden, als im „Vorderen“ (Kapitel 2.1.2).

### 3.3.1. Einbeziehung des Hall-Sensors

Eine Erweiterung des Konzeptes zur Feststellung der Geschwindigkeit, die im Rahmen dieser Bachelorarbeit umgesetzt wurde, stellt die unterstützende Einbeziehung des auf dem Fahrzeug montierten Hall-Sensors dar. Zwar liefert dieser, wie in der Fehleranalyse (vgl. Kapitel 1.3) aufgezeigt, keine genauen Werte, es kann jedoch, angesichts des festgestellten relativen Fehlers, ein Näherungswert angenommen werden.

Die Werte des Hall-Sensors werden genutzt, um die optisch gemessene Geschwindigkeit zu verifizieren.

## 3.4. Implementierung des Algorithmus

Das folgende Kapitel beschreibt den Vorgang, der nötig war, um eine genaue Geschwindigkeitsanalyse erstellen zu können. Hierbei ist zu beachten, dass die Koordinatenangaben unterschiedlich ausgedrückt werden. So ist es teilweise notwendig, die Daten in Welt-Koordinaten vorliegen zu haben, Berechnungen werden jedoch oft in Bild-Koordinaten durchgeführt (vgl. Abbildung 2.2 - Kameramontage (Jenning, 2008)).

### Vorbereitung

Zu Beginn der Entwicklung muss eine Umgebung geschaffen werden, deren Parameter sich wiederholen und vorhersagen lassen. Im konkreten Fall bedeutet dies, dass die Werte des Algorithmus auf einem einzelnen Bild entwickelt werden. Hierbei handelt es sich um ein Einzelbild aus der Aufzeichnung einer Testfahrt. Wichtig ist, dass das Bild folgende Kriterien erfüllt:

1. das Bild wurde mit einer kalibrierten Kamera aufgenommen
2. das Fahrzeug befindet sich in optimaler Position (siehe Kapitel 3.1.1)
3. ein Mittelliniensegment befindet sich an einer sehr guten Position

Die Abbildung 2.3 zeigt das Ursprungs-Bild, auf dem anfangs die Algorithmus-Entwicklung stattfand.

### Erster Schritt - Mittelliniensegment erkennen

Als Grundlage für die Erkennung der kurzen Kanten eines Mittelliniensegmentes ist es, wie in Kapitel 3.2 beschrieben, notwendig, mittels horizontaler Scanlinien / ROIs zwei Punkte auf eben diesem Mittelliniensegment zu finden. Hierbei wird eine horizontale Scanlinie verwendet, wie sie in Abbildung 3.2 gezeigt ist. Auf dem festen Testbild werden also die Positionen für zwei Scanlinien ermittelt, die ein Mittelliniensegment schneiden. Hierbei wird darauf geachtet, dass die Scanbereiche jeweils im ersten und letzten Drittel des Segments liegen.
Zunächst wird der Bereich definiert, in dem die erste Scanlinie platziert werden muss. Diese dient dann als Ursprung für alle weiteren verwendeten Scan-Bereiche. Als Initialwert für den Beginn der ersten Scanlinie wurden folgende Werte ermittelt:

| Beginn Scan-Linie | x-Koordinate | y-Koordinate |
|---|---|---|
| vordere | 7 | -33 |

Tabelle 3.3.: Initialwerte

Wie zuvor erwähnt, wird die Position der zweiten Linie anhand der Koordinaten der initialen Scanlinie festgelegt. Hierfür ist in Welt-Koordinaten ein Abstand zu definieren, der die Verschiebung in x-Richtung definiert. Als optimale Entfernung wurde hier ein Wert von 5 Zentimetern ermittelt.

#### Zweiter Schritt - Konzept für neuen ROI

Das im ersten Schritt erstellte ROI-Paar wird nun genutzt, um zwei Punkte des Mittelliniensegments zu detektieren. Anhand dieser Punkte ist es im weiteren Verlauf möglich, einen neuen ROI zu erstellen, der direkt auf dem Mittelliniensegment liegt.
Das Konzept hierfür beinhaltet zunächst, die beiden gefundenen Punkte auf der Mittellinie miteinander zu verbinden (vgl. Tabelle 3.1, *FirstPoint* und *SecondPoint*). Unter der Vorraussetzung, dass die Punkte des im ersten Schritt erstellten ROI-Paares mittig auf dem Mittelliniensegment liegen, wird durch das Verbinden dieser Punkte automatisch eine Linie auf die Mittellinie gelegt (vgl. Abbildung 3.5, Gerade $y$).
Um das Verbinden der Punkte zunächst optisch darstellen zu können, wird die Linie unter Verwendung von openCV erstellt. Hierbei stellt sich klar heraus, dass zwar eine Verbindung zwischen den gefundenen Punkten besteht, die kurzen Linienkanten des Mittelliniensegments allerdings nicht erkannt werden können (die Linine reicht nicht wie gewünscht über das Ende der Mittellinine heraus). Es muss also zunächst eine Möglichkeit gefunden werden, die Verbindung zwischen den im ersten Schritt gefundenen Punkten so weit in beide Richtungen zu verlängern, dass sie über die Enden der Mittellinine herausragt.

#### Dritter Schritt - Neuen ROI korrekt formatieren

Der erste Ansatz für die Verlängerung der Verbindung zwischen den beiden gefundenen Punkten des Mittelliniensegments beinhaltete das Multiplizieren der Katheten *straight_a* und *straight_b* (vgl. Tabelle 3.1) um einen einstellbaren Faktor. Hierbei wurde bereits auf den Winkel des ROI im Raum Rücksicht genommen.
Nach einigen Tests stellte sich jedoch heraus, dass die neu berechnete Länge der ROI stark von

dem Winkel im Raum abhängt. So wird der ROI schnell länger, wenn der Winkel kleiner wird. Nach einigen Überlegungen wurde dann das Konzept, wie es in Kapitel 3.2 - Algorithmus zur Kantenfindung beschrieben ist, umgesetzt. Anstatt die Linie von sich aus zu verlängern, werden zunächst die neuen Endpunkte des ROI berechnet, um die Linie erst dann zu verlängern bzw. zu nutzen.

#### Vierter Schritt - ROI durchlaufen

Nachdem ein ROI geschaffen wurde, der den Anforderungen entspricht, nämlich sowohl direkt auf dem Mittelliniensegment zu liegen, als auch über die kurzen Enden des Segments hinauszuragen, gilt es herauszufinden, wie diese kurzen Enden detektiert werden können. Die einfache und dennoch wirkungsvollste Lösung hierfür ist die Nutzung der Idee, den ROI pixelweise zu durchlaufen und dabei die Grauwerte der jeweils nebeneinanderliegenden Pixel miteinander zu vergleichen (vgl. Kapitel 3.1.2 - ROIs nutzen). Anstatt jedoch jeweils nur den x-Wert zu erhöhen (für einen vertikal liegenden ROI), muss der Winkel des neuen Scanbereichs beachtet werden und auch der entsprechende y-Wert eines neuen x-Wertes errechnet werden. Hierbei werden zunächst die bekannten Werte des Startpunktes *newFirstPoint* (vgl. Tabelle 3.1) verwendet und in allgemeine Geradengleichung

$$y = m \cdot x + b \tag{3.19}$$

eingesetzt. Die Steigung $m$ und der y-Achsen-Abschnitt $b$ sind bereits aus vorherigen Berechnungen bekannt (vgl. Formel 3.14 und 3.15). Der nächste Bildpunkt wird festgelegt, indem der x-Wert um 1 erhöht wird und der dazugehörige y-Wert durch Umstellung der Gleichung 3.19 errechnet wird.
Durch den Vergleich der Grauwerte der beiden Pixel kann nun festgestellt werden, ob es sich um einen schwarz-weiß- bzw. weiß-schwarz-Übergang handelt.

#### Fünfter Schritt - Scaneigenschaften verbessern

In Anlehnung an die Masterarbeit von E. Hensel (vgl. Hensel (2012)) wird das im vierten Schritt erläuterte Durchlaufen des neuen ROI verändert, indem nicht mehr der gesamte Scanbereich durchlaufen wird, sondern nur die Abschnitte, in denen sich logisch betrachtet eine kurze Kante eines Mittelliniensegmentes befinden kann. Der Bereich zwischen den im ersten Schritt detektierten Punkte enthält niemals eine der gesuchten Kanten und kann somit ausgeschlossen werden (vgl. Abbildung 3.10).
Für die Umsetzung der Idee wird das Konzept genutzt, wie es im vierten Schritt beschrieben

ist. Aus einem ROI werden jedoch Zwei geschaffen, wodurch auch der Suchalgorithmus ein zweites Mal ausgeführt werden muss. Als Startwert für den jeweiligen ROI wird hierfür der ursprünglich detektierte Linienpunkte gewählt. Je nach Suchrichtung muss nun der x- und der zugehörige y-Wert erhöht **oder** verringert werden. Den Endwert des Suchalgorithmus stellt jeweils die neu festgelegten Endpunkte *newFirstPoint* bzw. *newSecondPoint* des neuen ROI dar (vgl. Abbildung 3.10).

# 4. Testergebnisse

In diversen Versuchsreihen wurde der entwickelte Algorithmus (vgl. Kapitel 3.2) auf dem Versuchsfahrzeug getestet. Die Ergebnisse der Messungen wurden unter dem Aspekt zweier relevanter Kriterien ausgewertet. Zum Einen handelt es sich um das Hauptmerkmal des Algorithmus, nämlich die festgestellte Geschwindigkeit, und zum Anderen die Zeit für die Berechnung eben dieser Feststellung.
Um den Vergleich von verschiedenen Messungen besser darstellen zu können, werden hier nicht nur die Ergebnisse der Tests auf dem Versuchsfahrzeug, sondern auch die Testdaten vergleichbarer Simulationen auf unterschiedlichen PC-Plattformen hinzugezogen.

## 4.1. Testdurchführung

Wie eingangs erwähnt, wurden die Tests auf verschiedenen Plattformen durchgeführt. In erster Linie sind jedoch die Ergebnisse der Versuche auf dem Fahrzeug von Bedeutung. Die Vergleichbarkeit festgestellter Daten ist hierbei von großer Wichtigkeit. Es wurde also eine Methode angewendet, die die gespeicherten Werte aller getesteten Plattformen kompatibel macht:

Der zur Geschwindigkeitsdetektion entwickelte Algorithmus (vgl. Kapitel 3.2) wird durch einen *Scheduler*[1] regelmäßig während des Programmablaufs aufgerufen. Wird bei einem Aufruf die Geschwindigkeit festgestellt, so wird das Ergebnis in einem Terminal bzw. in einer Datei ausgegeben. Dies geschieht fortlaufend und dient zur späteren Auswertung der Messergebnisse. Ähnlich verhält es sich mit der Feststellung der Geschwindigkeit durch den am Fahrzeug angebrachten Hall-Sensor: Die Funktion zur Messung der Geschwindigkeit mittels dieses Sensors ist in die Methode integriert, in der auch der o.g. Algorithmus aufgerufen wird. Bei jedem Aufruf wird der aktuelle Messwert zur Ermittlung der zurückgelegten Wegstrecke und somit zur gefahrenen Geschwindigkeit festgestellt. Bei Bedarf wird auch dieser Wert in das Terminal bzw. die Datei zur späteren Auswertung ausgegeben.

[1] Steuerprogramm für die Ausführung mehrerer Prozesse

Die Ermittlung der Rechenzeit, also der Zeit, die benötigt wird, um die Methode mit samt des Algorithmus zur Geschwindigkeitsdetektion auszuführen, wird schlicht mit der Verwendung zweier Zeitstempel realisiert: Bei einem Aufruf der Methode wird zu Beginn ein Zeitwert gemessen und gespeichert. Nach Ausführung des Algorithmus wird eine weitere Zeit gemessen, von der der anfangs ermittelte Wert subtrahiert wird. Die so festgestellte Differenz stellt die Zeit (üblicherweise in Millisekunden) dar, die für die Berechnung erforderlich war. Auch diese Werte werden schließlich ausgegeben.

Zum Vergleich der Daten aller Plattformen werden diese tabellarisch erfasst (vgl. Anhang) und zusammenfassend in Kapitel 4.2 erläutert.

Zur besseren Übersicht ist die Erfassung der Daten in der folgenden Darstellung bildlich verdeutlicht.

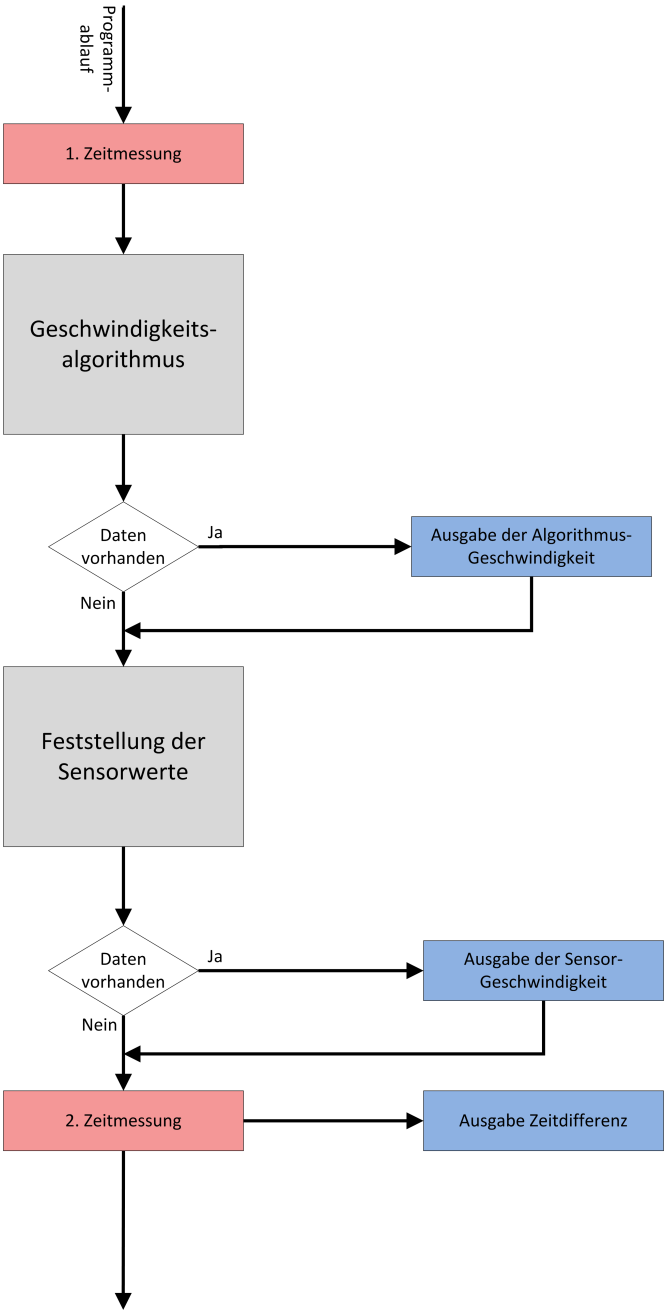


Abbildung 4.1.: Feststellung der Ergebnisse

### Vergleichbarkeit von Messwerten

Für die Auswertung der Messdaten wurden jeweils mehrere Tests auf den verschiedenen Plattformen durchgeführt. Bei den Versuchen mit dem Fahrzeug wurden die Daten ausgewertet, die direkt von den Live-Bildern für den Geschwindigkeitsalgorithmus, beziehungsweise den aktuellen Werten der Sensoren für die sensorische Geschwindigkeitsfeststellung geliefert wurden.
Die Simulationen konnten mittels der aufgezeichneten Werte ausgewertet werden. Die gespeicherten Daten beinhalten sowohl aufgenommene Bilddaten der Kamera, als auch die zu jedem Bild gehörigen Werte der Sensorik des Fahrzeugs. Somit wird die Vergleichbarkeit von Simulations- und Echtzeitdaten sichergestellt.

### Veränderbare Einstellungen

Um die Funktion des Algorithmus zur optischen Geschwindigkeitsdetektion (vgl. Kapitel 3.2) unter verschiedenen Testbedingungen optimal nutzen zu können, ist es möglich, diverse Parameter mittels einer Web-Oberfläche zu ändern, die sich auf die Funktion des Algorithmus auswirken. Folgende Werte können hierbei variiert werden:

| Parameter | Standardwert | Erläuterung |
|---|---|---|
| priority | 3 | Priorität, mit der der Scheduler die Methode aufruft |
| addLenght | 35 | Distanz in Pixeln, um die die Punkte *FirstPoint*[2] und *secondPoint*[2] jeweils verschoben werden sollen |
| draw | 1 | Schaltet das Einzeichnen der Scanlinien ein oder aus (Werte 1 oder 0) |
| getSpeedSensorOutput | 1 | Schaltet die Ausgabe der Geschwindigkeit des Hall-Sensors ein oder aus (Werte 1 oder 0) |
| initialXPos | 7 | Ursprungs-X-Koordinate für die erste Scanlinie[3] |
| initialYPos | 33 | Ursprungs-Y-Koordinate für die erste Scanlinie[3] |
| maxTime | 1 | maximale Zeit, die für die Messung der Differenz zweier Kantenpunkte genutzt wird |
| roiInterval | 5 | Abstand zwischen den horizontalen Scanlinien eines ROI-Paares in Pixeln |
| roiSpacing | 15 | Abstand zwischen den jeweiligen horizontalen ROI-Paaren in Pixeln |
| roiWidth | 18 | Breite einer horizontalen Scanlinie in Pixeln |
| showImage | 1 | Schaltet das Anzeigen der Kamerabilder in einem neuen Fenster ein oder aus (Werte 1 oder 0) |
| showImageHeight | 480 | Höhe des anzuzeigenden Bildes in Pixeln |
| showImageWidth | 752 | Breite des anzuzeigenden Bildes in Pixeln |
| threshold | 15 | Grenzwert für die Erkennung einer Grauwertdifferenz |

Tabelle 4.1.: Parameter

Im Verlauf der verschiedenen Testdurchführungen wurde der Algorithmus jeweils durch Veränderungen der oben aufgezeigten Parameter angepasst, um optimale Ergebnisse erzielen zu können. Diese Parameter werden im folgenden Kapitel aufgezeigt.

[2] vgl. Tabelle 3.1
[3] vgl. Kapitel 3.4 - Erster Schritt

## 4.2. Feststellung korrekter Messergebnisse in Anlehnung an Sensorwerte

Dieses Kapitel beinhaltet den Vergleich der Messungen zwischen dem Algorithmus zu optischen Geschwindigkeitsdetektion und den Werten des Hall-Sensors. Hierbei wird festgestellt, in wie fern die Ergebnisse übereinstimmen.

Für die Simulationen wurden immer die selben Aufnahmen des Testfahrzeugs *Diamond* verwendet, um Vergleichbarkeit herzustellen. Die Veränderungen der Parameter in Bezug auf die Standard-Parameter aus Tabelle 4.1 wurden farblich hinterlegt.

### Simulation auf dem Computer

Die ersten Simulationsdaten wurden mittels einer virtuellen Maschine auf dem bereitgestellten Computer der HAW Hamburg ermittelt. Die Spezifikationen des Rechner sind aus dem Anhang zu entnehmen (vgl. Anhang C).

Folgende Einstellungen der Parameter wurden genutzt, um optimale Ergebnisse zu erzielen:

| **Parameter** | **Wert** |
|---|---|
| priority | 3 |
| addLenght | 45 |
| draw | 1 |
| getSpeedSensorOutput | 1 |
| initialXPos | 1.0 |
| initialYPos | -33.0 |
| maxTime | 1 |
| roiInterval | 12 |
| roiSpacing | 26 |
| roiWidth | 20 |
| showImage | 1 |
| showImageHeight | 480 |
| showImageWidth | 752 |
| threshold | 50 |

Tabelle 4.2.: Parameter Computer-Simulation

Die Option *addLenght* wurde erhöht, um den vertikalen Scanbereich zu erweitern (vgl. Abbildung 3.7). Um auch in engeren Kurven eine Erkennung zu gewährleisten, wurden die Parameter *roiInterval*, *roiSpacing* und *roiWidth* vergrößert. Der *threshold* wurde erhöht, um die Erkennung von Schwarz-Weiß-Unterschieden etwas „aufzuweichen“.

Bei einer Messreihe, die im Anhang A einzusehen ist, wurde mit dem entwickelten Algorithmus eine durchschnittliche Geschwindigkeit von **31,50 cm/s** bzw. **1,13 km/h** festgestellt. Die sensorische Messung ergab **31,34 cm/s** bzw. **1,13 km/h**. Die Geschwindigkeit des Algorithmus weicht bei dieser Messung also um **0,53 %** ab.

### Simulation auf dem Notebook

Eine weitere Simulation wurde auf einem privaten Notebook durchgeführt, dessen Spezifikationen ebenfalls im Anhang A zu finden sind. Bei diesen Versuchen wurde die selbe virtuelle Maschine genutzt, wie bereits in der o.g. Simulation mit dem Computer.
Die verwendeten Parameter gleichen denen des Tests mit dem Computer, da hier die selben Bilddaten verwendet wurden und somit eine Anpassung nicht notwendig war.

Die Messreihe ergab folgende Ergebnisse: Die durchschnittliche Geschwindigkeit, die mit dem Algorithmus gemessen wurde, betrug **23,59 cm/s** bzw. **0,85 km/h**. Mit dem Sensor wurden **29,96 cm/s** bzw. **1,08 km/h** festgestellt. Die Abweichung beträgt somit durchschnittlich **21,23 %**.

### Messungen mit dem Fahrzeug

Aufgrund ständig variierender Einflüsse für das Fahrzeug, beispielsweise veränderte Lichtverhältnisse oder Streckenführungen, werden hier verschiedene Messreihen dargestellt, die für das Fahrzeug durchgeführt wurden. Die Ergebnisse wurden mit unterschiedlichen Parametern erzielt, welche jeweils im Folgenden aufgeführt werden. Die farbliche Markierung der Veränderungen von Parametern bezieht sich hierbei nur bei der ersten Messung auf die Standardwerte. Die Hervorhebungen der folgenden Messungen beziehen sich dann jeweils auf den vorhergehenden Versuch. Die genauen Messergebnisse der Messreihen könne im Anhang B eingesehen werden.

1. Messreihe:

| Parameter | Wert |
|---|---|
| priority | 3 |
| addLenght | 35 |
| draw | 1 |
| getSpeedSensorOutput | 1 |
| initialXPos | 0.0 |
| initialYPos | -22.0 |
| maxTime | 1 |
| roiInterval | 9 |
| roiSpacing | 14 |
| roiWidth | 13 |
| showImage | 0 |
| showImageHeight | 480 |
| showImageWidth | 752 |
| threshold | 60 |

Tabelle 4.3.: Parameter Messreihe 1

Im Vergleich zu den Standard-Parametern (vgl. Tabelle 4.1) wurde der Ursprung des ersten ROI verschoben. Außerdem wurden Breite und Abstände der ROIs angepasst; der *threshold* wurde heraufgesetzt.

Der entwickelte Algorithmus stellte im Schnitt eine Geschwindigkeit von **22,88 cm/s** bzw. **0,82 km/h** fest. Der Sensor maß **36,76 cm/s** bzw. **1,32 km/h**. Der Unterschied entspricht **37,78 %**.

2. Messreihe:

| Parameter | Wert |
|---|---|
| priority | 3 |
| addLenght | 35 |
| draw | 1 |
| getSpeedSensorOutput | 1 |
| initialXPos | 0.0 |
| initialYPos | -22.0 |
| maxTime | 1 |
| roiInterval | 9 |
| roiSpacing | 14 |
| roiWidth | 13 |
| showImage | 0 |
| showImageHeight | 480 |
| showImageWidth | 752 |
| threshold | 15 |

Tabelle 4.4.: Parameter Messreihe 2

Im Gegensatz zu Messreihe 1 wurde lediglich der *threshold* herabgesetzt.

Mit dem Algorithmus wurden im Mittel **31,78 cm/s** bzw. **1,14 km/h** gemessen. Der Sensor stellte hier eine Geschwindigkeit von **37,01 cm/s** bzw. **1,33 km/h** fest. Die Geschwindigkeitsabweichung liegt also bei **14,08 %**.

3. Messreihe:

| Parameter | Wert |
|---|---|
| priority | 3 |
| addLenght | 40 |
| draw | 1 |
| getSpeedSensorOutput | 1 |
| initialXPos | -5.0 |
| initialYPos | -22.0 |
| maxTime | 1 |
| roiInterval | 8 |
| roiSpacing | 14 |
| roiWidth | 13 |
| showImage | 0 |
| showImageHeight | 480 |
| showImageWidth | 752 |
| threshold | 15 |

Tabelle 4.5.: Parameter Messreihe 3

Mit *addLenght* wurde die Länge des Scanbereichs heraufgesetzt. Die Koordinate des Ursprungs - ROI wurde geändert (*initialXPos*) und der Abstand der einzelnen ROI wurde um 1 verringert (*roiInterval*).

Bei dieser Messreihe wurden mit dem Algorithmus **37,04 cm/s** bzw. **1,33 km/h** gemessen, mit dem Sensor wurden **37,13 cm/s** bzw. **1,34 km/h** festgestellt. Die Geschwindigkeitsabweichung beträgt also **0,26 %**.

4. Messreihe:

| Parameter | Wert |
|---|---|
| priority | 3 |
| addLenght | 40 |
| draw | 1 |
| getSpeedSensorOutput | 1 |
| initialXPos | -5.0 |
| initialYPos | -22.0 |
| maxTime | 1 |
| roiInterval | 9 |
| roiSpacing | 18 |
| roiWidth | 13 |
| showImage | 0 |
| showImageHeight | 480 |
| showImageWidth | 752 |
| threshold | 10 |

Tabelle 4.6.: Parameter Messreihe 4

In der Messreihe 4 wurden die jeweiligen Abstände zwischen den einzelnen ROIs und den ROI-Paaren verändert. Des Weiteren wurden der *threshold* ein weiteres Mal herabgesetzt.

Der Algorithmus stellte hier durchschnittlich **43,06 cm/s** bzw. **1,55 km/h** fest. Der Sensor maß **36,73 cm/s** bzw.**1,31 km/h** und weicht somit um **17,24 %** von der Geschwindigkeitsmessung des Algorithmus ab.

5. Messreihe:

| Parameter | Wert |
|---|---|
| priority | 3 |
| addLenght | 40 |
| draw | 1 |
| getSpeedSensorOutput | 1 |
| initialXPos | -5.0 |
| initialYPos | -22.0 |
| maxTime | 1 |
| roiInterval | 9 |
| roiSpacing | 18 |
| roiWidth | 14 |
| showImage | 0 |
| showImageHeight | 480 |
| showImageWidth | 752 |
| threshold | 8 |

Tabelle 4.7.: Parameter Messreihe 5

In der letzten Messreihe wurden sowohl die Breite der ROIs noch einmal um 1 erweitert, als auch der *threshold* für die Erkennung des Grauwertunterschieds um 2 verringert.

Im Schnitt betrug die durch den Algorithmus berechnete Geschwindigkeit **29,43 cm/s** bzw. **1,06 km/h**. Der Sensor maß durchschnittlich **40,39 cm/s** bzw. **1,45 km/h**. Dies entspricht einer Abweichung von **27,12 %**.

## 4.3. Zusammenfassung der Ergebnisse

Die in Kapitel 4.2 - Feststellung korrekter Messergebnisse in Anlehnung an Sensorwerte erläuterten Ergebnisse der verschiedenen Testreihen sind in der nachfolgenden Tabelle 4.8 noch einmal zusammengefasst:

| Messreihe | Sensor | Algorithmus | % - Abweichung | Ø- Abweichung |
|---|---|---|---|---|
| Computer-Simulation | 29,96 cm/s | 23,59 cm/s | 21,23 % | 16,69 % |
| Notebook-Simulation | 31,34 cm/s | 31,50 cm/s | -0,52 % | -2,23 % |
| 1. Messreihe | 36,76 cm/s | 22,88 cm/s | 37,78 % | 28,39 % |
| 2. Messreihe | 37,01 cm/s | 31,79 cm/s | 14,08 % | -0,14 % |
| 3. Messreihe | 37,13 cm/s | 37,04 cm/s | 0,26 % | -20,38 % |
| 4. Messreihe | 36,72 cm/s | 43,06 cm/s | -17,24 % | -34,30 % |
| 5. Messreihe | 40,39 cm/s | 29,43 cm/s | 27,12 % | 16,61 % |

Tabelle 4.8.: Zusammenfassung Messreihen

Hieraus geht hervor, dass mit der Geschwindigkeitserkennung mittels der Simulation auf dem Notebook die größte Präzision erzielt werden konnte. Die Messergebnisse wichen hier nur um 0,52% von den sensorischen Werten ab. Auch die durchschnittliche Abweichung von 2,23% der einzelnen Messergebnisse (vgl. Anhang B) bestätigen die Genauigkeit.
Die Parameterwahl der 3. Messreihe sind eine Gute Wahl für die Anwendung des Algorithmus auf dem Versuchsfahrzeug *Diamond*. Zwar weichen die Endergebnisse der Messung um 14,08% voneinander ab, der durchschnittliche Unterschied zwischen den Daten des Algorithmus und denen des Sensors beträgt jedoch nur 0,14%.

Die wiederholten Testreihen zeigen, dass der Algorithmus je nach Einstellung der Parameter starke Abweichungen aufzeigt. Da der Hall-Sensor, wie in Kapitel 1.3 - Geschwindigkeits-/Fehleranalyse dargestellt, eine Abweichung von circa 5 % zu der eigentlich gefahrenen Geschwindigkeit hat, sollte davon ausgegangen werden, dass die Werte der optischen Detektion um 5 % höher liegen. Dies ist nur durch die jeweilige optimale Einstellung der Parameter zu erreichen, die den Umgebungsvariablen, wie Licht und Streckenführung, angepasst werden muss. Außerdem stellte sich in der Testphase heraus, dass der Hall-Sensor einen Defekt aufwies, der in Kapitel 5.3 - Komplikationen erläutert wird.

# 5. Zusammenfassung und Aussichten

## 5.1. Zusammenfassung

Als Grundlage für diese Arbeit diente die autonome Fahrzeugplattform ”Saphir“, die im Rahmen des Projektes FAUST an der HAW Hamburg aufgebaut und entwickelt wurde. Ziel war es, die vorhandene Kamera des Fahrzeugs und den montierten Rechner zu nutzen, um die Geschwindigkeit mittels Bildverarbeitung feststellen zu können. Die Verwendung von Sensoren sollte hierbei ausschließlich zur Verifikation der ermittelten Daten genutzt werden.

Vor der Entwicklung des Algorithmus zur optischen Geschwindigkeitserkennung wird analysiert, welche Ungenauigkeit der montierte Hall-Sensor aufweist. Außerdem wird erörtert, in welcher Form die Bilddaten der Kamera vorliegen und nutzbar gemacht werden können. Hierbei wird vor allem auf die Linsenverzeichnung und die perspektivische Transformation eingegangen. Diese müssen für die weitere Bearbeitung bekannt sein und in die Berechnungen mit einbezogen werden.

Der Hauptteil der Arbeit beinhaltet die Konzeption des Algorithmus, der für die Feststellung der Geschwindigkeit verantwortlich ist. An erster Stelle steht eine Analyse der Kameradaten auf eindeutig identifizierbare Merkmale, wobei sich die Segmente der Mittellinie als geeignet herausstellten. Durch wiederholte Anwendung eines Algorithmus zur Erkennung markanter Grauwertunterschiede wird ein weißes Mittelliniensegment auf schwarzen Untergrund erkannt und gespeichert. Im weiteren Verlauf werden diese gespeicherten Punkte genutzt, um mittels des oben genannten Algorithmus die kurzen Linienkanten zu detektieren. Da die kurzen Enden der Mittelliniensegmente bei Bewegung des Fahrzeugs in Richtung der Kamera laufen, können die Strecke und die dafür benötigte Zeit für diese Bewegung zur Errechnung der Geschwindigkeit genutzt werden.

Die Testergebnisse unter verschiedenen Bedingungen zeigen, dass der Algorithmus durchaus verwendbar ist. Veränderbare Parameter machen die Ausführung flexibel und der Algorith-

mus lässt sich an die jeweiligen Umgebungs- und Fahrzeugeigenschaften anpassen. Zwar unterliegen die einzelnen Messdaten im Vergleich zu sensorischen Messungen teilweise Ungenauigkeiten, im Mittel jedoch stellen sich die Werte als äußerst präzise dar.

Die Intention, eine Geschwindigkeitserkennung für autonome Fahrzeuge zu schaffen, die keine sensorischen Werte benötigt und alleine mittels Bildverarbeitung modelliert wird, konnte umgesetzt werden.

## 5.2. Einsatzmöglichkeiten

Für den Einsatz der optische Geschwindigkeitserkennung gibt es diverse Möglichkeiten, die hier konzeptionell erörtert werden.

Es ist durchaus denkbar, den Algorithmus als alleinige Messeinheit für die Geschwindigkeit eines Fahrzeugs zu nutzen. Hierbei muss jedoch beachtet werden, dass eine Vielzahl von unregelmäßigen Umgebungsvariablen stark variieren können. So kann zwar der Einfluss nachlassender Motorleistung oder starker Abrieb der Reifen vernachlässigt werden, jedoch wirken sich Veränderungen der Fahrbahn, gerade in Bezug auf die Mittelliniensegmente, stark auf die Funktion des Algorithmus aus. Auch die Beeinflussung unterschiedlicher Lichtverhältnisse und deren Auswirkung auf die Kameradaten muss berücksichtigt werden. Als sinnvoll stellt sich also die Nutzung auf genormten Fahrbahnen, wie sie beispielsweise im Carolo-Cup zu finden sind, heraus.

Die Nutzung des entwickelten Algorithmus als unterstützende Einheit stellt eine weitere Einsatzmöglichkeit dar. Hierbei kann die optische Geschwindigkeitserkennung zum Abgleich sensorisch ermittelter Daten unterstützend genutzt werden. Unterschiede zwischen den Messungen können als Hinweis auf diverse Vorkommnisse gewertet werden. So kann unter Anderem der Ausfall eines Sensors schnell erkannt werden. Auch die oben genannten physikalischen Einflüsse wie Reifenabrieb oder nachlassende Motorleistung können registriert werden. Als mögliche Reaktion sind optische Meldungen als Textausgabe auf einem Terminal oder das Einschalten einer Warnleuchte denkbar. Stark abweichende Werte können final auch für ein Sicherheitssystem genutzt werden und das Fahrzeug stoppen, wenn beispielsweise wegen einer ausgefallenen Kamera keine optischen Daten ermittelt werden können, die Sensoren jedoch noch Bewegung anzeigen.

## 5.3. Komplikationen

Bei der Erstellung dieser Arbeit und den anschließenden Tests traten Hürden auf, die an dieser Stelle erwähnt werden sollten.

Zum Zeitpunkt der Entwicklung des Algorithmus zur optischen Geschwindigkeitsmessung existierten zwei unterschiedliche Linux Kernelversionen, die je auf einem der Fahrzeuge installiert waren. Der eigentliche Entwurf für den neueren Kernel 3.2 musste unterbrochen werden, da dieses Fahrzeug nach Teilnahme an dem Carolo-Cup nicht fahrbereit war. Um die Weiterentwicklung nicht unterbrechen zu müssen, wurde die Programmierung auf den älteren Kernel 2.6 umgestellt. Hauptaugenmerk liegt hierbei auf der Veränderung der openCV-Version. Während der Kernel 3.2 openCV 2 benutzt, arbeitet der Kernel 2.6 noch mit openCV 1. Der Unterschied besteht hauptsächlich in der Nutzung und dem Aufruf von Methoden, sowie den Eingabeparametern, die bei der Umstellung geändert werden mussten.

Die im Rahmen dieser Bachelorabeit genutzte Hardware unterliegt ständigen Veränderungen. So wurde gegen Ende der Arbeit eine Erneuerung der Hardware an beiden Versuchsfahrzeugen durchgeführt, um diese Bau- und Funktionsgleich zu machen. Der für die Verifikation der optischen Geschwindigkeitsdaten genutzte Hall-Sensor existiert nach dem Umbau nicht mehr. Dafür ist jedoch ein im Elektromotor bereits vorhandener Hall-Sensor nun ansprechbar. Des Weiteren wurde der Pico-PC, mit dem die Berechnungen des Algorithmus durchgeführt werden, durch ein performanteres Gerät ersetzt. Auch die Kameralinse wurde ersetzt, was jedoch nach ordentlicher Kalibrierung für die Funktion des Algorithmus keine Einbußen bedeutet.

Nach Beendigung der Testphase stellte sich heraus, dass der genutzte Hall-Sensor einen Defekt aufwies. Aufgrund häufiger Nutzung des Fahrzeugs hatten sich zwei der 20 Neodym-Magnete gelöst. Hierdurch fehlten demnach bei jeder Radumdrehung zwei Ticks, die das Ergebnis stärker verfälschten, als in Kapitel 1.3 nachgewiesen. Dies wirkt sich ebenfalls leicht auf die dargestellten Testergebnisse aus.

## 5.4. Bedeutung für die Zukunft

In der Automobilindustrie werden kontinuierlich neue Ideen und Konzepte entwickelt, die das Fahren einfacher und komfortabler machen. Hierzu zählen auch die Fortschritte im Bereich der optischen Systeme. Ob in unterstützender Form, wie beispielsweise als Rückfahrhilfe oder Informationseinrichtung, oder in sicherheitsrelevanter Hinsicht - Kameras sind fester Bestandteil moderner Fahrzeuge.

In Anbetracht dieser Entwicklungen ist der im Rahmen dieser Bachelorarbeit entworfene Algorithmus auch außerhalb des universitären Rahmens anwendbar. Die genutzte Arbeitsumgebung bildet die realen Verhältnisse im Straßenverkehr zwar nur eingeschränkt ab, zeigt jedoch einen guten Überblick über die Möglichkeiten, die durch weitere Ausformung dieser Arbeit geboten werden können.

# Literaturverzeichnis


[Carolo-Cup 2013] Carolo-Cup: *Carolo-Cup*. 2013. – URL http://www.carolo-cup.de

[FAUST 2013] FAUST: *Carolo Cup v.2*. 2013. – URL http://faust.informatik.haw-hamburg.de/index.php?section=fahrzeuge&sub=1

[Hensel 2012] Hensel, Enrico: *Kamerabasierte Fahrspuridentifikation und -repräsentation unter Berücksichtigung eines kinematischen Bewegungsmodells*. 2012

[Jacobi 2009] Jacobi, Dirk: *Identifikation und räumliche Lokalisierung skalierungsinvarianter Merkmale für die visuelle Navigation*. 2009

[Jenning 2008] Jenning, Eike: *Systemidentifikation eines autonomen Fahrzeugs mit einer robusten, kamerabasierten Fahrspurerkennung in Echtzeit*. 2008

[Liedtke 2011] Liedtke, Andreas: *Realisierung eines robusten Verfahrens zur Identifikation von Positionsmarken unter Verwendung von Bildverarbeitung in Videobildern*. 2011

[OpenCV 2013] OpenCV: *About*. 2013. – URL http://opencv.org/about.html

[Tamiya 2013] Tamiya: *Tamiya*. 2013. – URL http://www.tamiya.de

# Abbildungsverzeichnis

# Tabellenverzeichnis

# A. Messdaten der Simulationen

Computer-Simulation

| | Sensor | | Algorithmus | | Prozentuale Abweichung |
|---|---|---|---|---|---|
| | cm / s | km / h | cm / s | km / h | |
| 1 | 30,5984 | 1,1015424 | 43,4537 | 1,5643332 | -42,0129810709 |
| 2 | 36,445 | 1,31202 | | | |
| 3 | 31,0514 | 1,1178504 | 38,0178 | 1,3686408 | -22,4350592888 |
| 4 | 30,4106 | 1,0947816 | 26,2325 | 0,94437 | 13,7389594418 |
| 5 | 31,7196 | 1,1419056 | | | |
| 6 | 30,8152 | 1,1093472 | | | |
| 7 | 31,3094 | 1,1271384 | 26,4672 | 0,9528192 | 15,465642906 |
| 8 | 30,0758 | 1,0827288 | 23,117 | 0,832212 | 23,1375391511 |
| 9 | 31,9515 | 1,150254 | | | |
| 10 | 30,8964 | 1,1122704 | | | |
| 11 | 31,2104 | 1,1235744 | | | |
| 12 | 30,4254 | 1,0953144 | 25,2869 | 0,9103284 | 16,8888494482 |
| 13 | 31,2483 | 1,1249388 | 39,4209 | 1,4191524 | -26,1537427636 |
| 14 | 31,0592 | 1,1181312 | 24,1938 | 0,8709768 | 22,1042396456 |
| 15 | 31,5198 | 1,1347128 | 30,2445 | 1,088802 | 4,0460282108 |
| 16 | 30,4151 | 1,0949436 | 35,6317 | 1,2827412 | -17,1513491654 |
| 17 | 31,1489 | 1,1213604 | 32,129 | 1,156644 | -3,1464995554 |
| 18 | 31,1503 | 1,1214108 | 26,8221 | 0,9655956 | 13,8945692337 |
| 19 | 31,4786 | 1,1332296 | | | |
| 20 | 30,8753 | 1,1115108 | | | |
| 21 | 30,7082 | 1,1054952 | 30,3787 | 1,0936332 | 1,073003302 |
| 22 | 30,8924 | 1,1121264 | 30,2192 | 1,0878912 | 2,1791767554 |
| 23 | 31,7308 | 1,1423088 | | | |
| 24 | 30,5128 | 1,0984608 | 25,0196 | 0,9007056 | 18,0029364726 |
| 25 | 31,2172 | 1,1238192 | 46,7256 | 1,6821216 | -49,6790231026 |
| 26 | | | 29,5633 | 1,0642788 | |
| 27 | 30,5133 | 1,0984788 | 29,2652 | 1,0535472 | 4,0903474878 |
| 28 | 31,4206 | 1,1311416 | 29,7374 | 1,0705464 | 5,3569950924 |
| 29 | 31,4668 | 1,1328048 | | | |
| 30 | 30,8519 | 1,1106684 | 37,8432 | 1,3623552 | -22,6608409855 |
| 31 | 30,8903 | 1,1120508 | 28,029 | 1,009044 | 9,2627782832 |
| 32 | 30,874 | 1,111464 | 23,7662 | 0,8555832 | 23,0219602254 |
| 33 | 31,6177 | 1,1382372 | | | |
| 34 | 30,6953 | 1,1050308 | 30,1298 | 1,0846728 | 1,8423015901 |
| 35 | 31,5013 | 1,1340468 | | | |
| 36 | 31,5013 | 1,1340468 | | | |
| 37 | 31,1778 | 1,1224008 | | | |
| 38 | 30,5919 | 1,1013084 | 31,331 | 1,127916 | -2,4159990063 |
| 39 | 31,0122 | 1,1164392 | 31,8511 | 1,1466396 | -2,7050644585 |
| 40 | 31,6129 | 1,1380644 | | | |
| 41 | 30,449 | 1,096164 | 32,8742 | 1,1834712 | -7,9647935893 |
| 42 | 30,9437 | 1,1139732 | 32,4734 | 1,1690424 | -4,9434941523 |
| 43 | 31,2946 | 1,1266056 | 30,9992 | 1,1159712 | 0,9439328191 |
| 44 | 31,473 | 1,133028 | | | |
| 45 | 30,8147 | 1,1093292 | | | |
| 46 | 31,2985 | 1,126746 | | | |
| 47 | | | 34,0465 | 1,225674 | |

| | | | | | |
|---|---|---|---|---|---|
| 48 | 30,0975 | 1,08351 | 21,2413 | 0,7646868 | 29,4250353019 |
| 49 | 31,8173 | 1,1454228 | | | |
| 50 | 31,008 | 1,116288 | | | |
| 51 | 31,2461 | 1,1248596 | | | |
| 52 | 30,3953 | 1,0942308 | 34,4824 | 1,2413664 | -13,4464867924 |
| 53 | 31,7373 | 1,1425428 | | | |
| 54 | 30,6839 | 1,1046204 | 46,2028 | 1,6633008 | -50,5766867967 |
| 55 | 30,7831 | 1,1081916 | 24,1996 | 0,8711856 | 21,3867349292 |
| 56 | 31,3013 | 1,1268468 | 29,0106 | 1,0443816 | 7,3182263995 |
| 57 | 31,0982 | 1,1195352 | 5,026 | 0,180936 | 83,8382928916 |
| 58 | 30,9137 | 1,1128932 | 4,51372 | 0,16249392 | 85,3989655072 |
| 59 | 31,5659 | 1,1363724 | 5,62407 | 0,20246652 | 82,1830836441 |
| 60 | 31,0286 | 1,1170296 | | | |
| 61 | 30,6082 | 1,1018952 | 48,7152 | 1,7537472 | -59,1573499912 |
| 62 | 31,178 | 1,122408 | 40,4944 | 1,4577984 | -29,8813265764 |
| 63 | 30,7854 | 1,1082744 | 29,9097 | 1,0767492 | 2,8445301994 |
| 64 | 31,6606 | 1,1397816 | 38,6077 | 1,3898772 | -21,9424142309 |
| 65 | 30,7504 | 1,1070144 | 34,1514 | 1,2294504 | -11,0600187315 |
| 66 | 30,7006 | 1,1052216 | 27,8925 | 1,00413 | 9,1467267741 |
| 67 | 31,7832 | 1,1441952 | | | |
| 68 | 31,0078 | 1,1162808 | | | |
| 69 | 30,5895 | 1,101222 | 29,9812 | 1,0793232 | 1,9885908563 |
| 70 | 31,373 | 1,129428 | | | |
| 71 | 31,234 | 1,124424 | | | |
| 72 | 31,0171 | 1,1166156 | | | |
| 73 | 31,0568 | 1,1180448 | | | |
| 74 | 31,0441 | 1,1175876 | | | |
| 75 | 30,9295 | 1,113462 | | | |
| 76 | 31,1656 | 1,1219616 | | | |
| 77 | 30,9639 | 1,1147004 | | | |
| 78 | 32,5876 | 1,1731536 | 42,965 | 1,54674 | -31,844628018 |
| 79 | 31,3405 | 1,128258 | | | |
| 80 | 30,7398 | 1,1066328 | 33,7473 | 1,2149028 | -9,7837331407 |
| 81 | 31,1235 | 1,120446 | | | |
| 82 | 29,113 | 1,048068 | 78,6052 | 2,8297872 | -170,0003434892 |
| 83 | 33,6403 | 1,2110508 | | | |
| 84 | 30,9506 | 1,1142216 | | | |
| 85 | 30,8302 | 1,1098872 | | | |
| 86 | 31,3402 | 1,1282472 | | | |
| 87 | 47,7501 | 1,7190036 | | | |
| | | | | | |
| Ø | 31,3389 | 1,1282004 | 31,50297426 | 1,134107073 | -2,2307197408 |
| | | | | Prozentual: | -0,523548227 |

Notebook_Simulation

| | Sensor | | Algorithmus | | Prozentuale Abweichung |
|---|---|---|---|---|---|
| | cm / s | km / h | cm / h | km / h | |
| 1 | 34,281 | 1,234116 | | | |
| 2 | 29,4868 | 1,0615248 | 14,3516 | 0,5166576 | 51,3287301437 |
| 3 | 32,6593 | 1,1757348 | | | |
| 4 | 30,457 | 1,096452 | 29,6079 | 1,0658844 | 2,7878648587 |
| 5 | 29,9731 | 1,0790316 | | | |
| 6 | 29,562 | 1,064232 | 7,35764 | 0,26487504 | 75,1111562141 |
| 7 | 28,8835 | 1,039806 | 31,2389 | 1,1246004 | -8,1548288815 |
| 8 | 18,2134 | 0,6556824 | 17,5065 | 0,630234 | 3,8812083411 |
| 9 | 30,3073 | 1,0910628 | | | |
| 10 | 14,4521 | 0,5202756 | 6,46028 | 0,23257008 | 55,2986763169 |
| 11 | 30,8638 | 1,1110968 | | | |
| 12 | 30,1212 | 1,0843632 | 25,2202 | 0,9079272 | 16,270932101 |
| 13 | 30,1912 | 1,0868832 | 23,6744 | 0,8522784 | 21,5850976443 |
| 14 | 33,0504 | 1,1898144 | | | |
| 15 | | | 11,2669 | 0,4056084 | |
| 16 | 28,5689 | 1,0284804 | 13,2804 | 0,4780944 | 53,514486032 |
| 17 | | | 27,0444 | 0,9735984 | |
| 18 | 31,3214 | 1,1275704 | 29,6593 | 1,0677348 | 5,3065954906 |
| 19 | | | 29,1428 | 1,0491408 | |
| 20 | 31,1537 | 1,1215332 | 41,8939 | 1,5081804 | -34,4748777834 |
| 21 | | | 19,0813 | 0,6869268 | |
| 22 | 30,9231 | 1,1132316 | 23,0561 | 0,8300196 | 25,4405282782 |
| 23 | 30,7872 | 1,1083392 | | | |
| 24 | 29,0112 | 1,0444032 | 17,2483 | 0,6209388 | 40,546064968 |
| 25 | 31,24 | 1,12464 | 40,9493 | 1,4741748 | -31,0797055058 |
| 26 | 29,6116 | 1,0660176 | 23,7992 | 0,8567712 | 19,6287941212 |
| 27 | 29,1701 | 1,0501236 | 10,0499 | 0,3617964 | 65,5472555802 |
| 28 | 31,7938 | 1,1445768 | | | |
| 29 | 31,1356 | 1,1208816 | 77,5377 | 2,7913572 | -149,0322974344 |
| 30 | | | 13,4918 | 0,4857048 | |
| 31 | 29,8125 | 1,07325 | 25,9644 | 0,9347184 | 12,907672956 |
| 32 | 30,45 | 1,0962 | 10,664 | 0,383904 | 64,9786535304 |
| 33 | 29,9696 | 1,0789056 | 18,2632 | 0,6574752 | 39,0609150606 |
| 34 | 31,3209 | 1,1275524 | 15,6312 | 0,5627232 | 50,093388121 |
| 35 | 31,8913 | 1,1480868 | | | |
| 36 | 30,5395 | 1,099422 | | | |
| 37 | | | 28,9104 | 1,0407744 | |
| 38 | 29,8436 | 1,0743696 | 18,7453 | 0,6748308 | 37,188207857 |
| 39 | | | 19,1328 | 0,6887808 | |
| 40 | 30,4714 | 1,0969704 | 36,6661 | 1,3199796 | -20,3295549269 |
| 41 | | | 34,9181 | 1,2570516 | |
| 42 | 31,287 | 1,126332 | 32,9721 | 1,1869956 | -5,3859430434 |
| 43 | | | 8,56632 | 0,30838752 | |
| 44 | 28,9427 | 1,0419372 | 16,0163 | 0,5765868 | 44,6620391325 |
| 45 | 32,3988 | 1,1663568 | 7,23907 | 0,26060652 | 77,6563638159 |
| 46 | 29,1063 | 1,0478268 | 49,987 | 1,799532 | -71,7394515964 |
| 47 | 30,5851 | 1,1010636 | 21,8962 | 0,7882632 | 28,4089311462 |

| | | | | | |
|---|---|---|---|---|---|
| 48 | 29,5799 | 1,0648764 | 14,4093 | 0,5187348 | 51,2868535729 |
| 49 | 30,0268 | 1,0809648 | 26,2665 | 0,945594 | 12,5231459896 |
| 50 | 31,8837 | 1,1478132 | 23,5797 | 0,8488692 | 26,0446560468 |
| 51 | 31,6517 | 1,1394612 | | | |
| 52 | 29,6605 | 1,067778 | 7,90407 | 0,28454652 | 73,3515281266 |
| 53 | | | 14,3134 | 0,5152824 | |
| 54 | 31,4093 | 1,1307348 | 11,0424 | 0,3975264 | 64,8435336031 |
| 55 | 31,8729 | 1,1474244 | | | |
| 56 | 27,8849 | 1,0038564 | 39,3215 | 1,415574 | -41,0135951716 |
| 57 | 28,5252 | 1,0269072 | 14,0832 | 0,5069952 | 50,6289175887 |
| 58 | 17,7896 | 0,6404256 | 32,7392 | 1,1786112 | -84,0356163152 |
| 59 | 29,8085 | 1,073106 | 61,8313 | 2,2259268 | -107,4284180687 |
| 60 | 30,7877 | 1,1083572 | 30,9376 | 1,1137536 | -0,4868827486 |
| 61 | 29,968 | 1,078848 | 15,2265 | 0,548154 | 49,1908035238 |
| 62 | 29,4903 | 1,0616508 | 8,98902 | 0,32360472 | 69,5187231056 |
| 63 | 33,2326 | 1,1963736 | | | |
| 64 | 28,4259 | 1,0233324 | 63,2075 | 2,27547 | -122,3588347247 |
| 65 | 29,7407 | 1,0706652 | 16,7017 | 0,6012612 | 43,8422767453 |
| 66 | 31,2382 | 1,1245752 | | | |
| 67 | 31,8391 | 1,1462076 | | | |
| 68 | 17,4335 | 0,627606 | 20,0128 | 0,7204608 | -14,7950784409 |
| 69 | 29,4578 | 1,0604808 | | | |
| 70 | 29,398 | 1,058328 | 12,3924 | 0,4461264 | 57,8461119804 |
| 71 | 32,8839 | 1,1838204 | 27,3721 | 0,9853956 | 16,7613938736 |
| 72 | 30,9011 | 1,1124396 | 39,2025 | 1,41129 | -26,8644158299 |
| 73 | 31,6449 | 1,1392164 | 20,4841 | 0,7374276 | 35,2688742894 |
| 74 | 29,5088 | 1,0623168 | 17,5783 | 0,6328188 | 40,4303123136 |
| 75 | 32,6087 | 1,1739132 | | | |
| 76 | 29,402 | 1,058472 | 28,5993 | 1,0295748 | 2,7300863887 |
| 77 | 31,5697 | 1,1365092 | 26,0511 | 0,9378396 | 17,4806855941 |
| 78 | 29,8341 | 1,0740276 | 10,7807 | 0,3881052 | 63,8645040407 |
| 79 | 30,8089 | 1,1091204 | 24,4132 | 0,8788752 | 20,7592611226 |
| 80 | 31,7997 | 1,1447892 | | | |
| 81 | 28,1588 | 1,0137168 | 9,15099 | 0,32943564 | 67,5022017984 |
| 82 | 33,6183 | 1,2102588 | 25,9556 | 0,9344016 | 22,7932405862 |
| 83 | 30,0753 | 1,0827108 | 12,8986 | 0,4643496 | 57,1123147566 |
| 84 | 30,9431 | 1,1139516 | | | |
| 85 | 29,1858 | 1,0506888 | | | |
| 86 | 33,6679 | 1,2120444 | | | |
| 87 | 30,9207 | 1,1131452 | | | |
| 88 | 30,9374 | 1,1137464 | | | |
| 89 | 31,0021 | 1,1160756 | | | |
| 90 | 31,0937 | 1,1193732 | | | |
| 91 | 31,0655 | 1,118358 | | | |
| Ø | 29,95768642 | 1,078476711 | 23,59901215 | 0,849564438 | 16,6873361143 |

| | |
|---|---|
| Prozentual: | 21,2255184757 |

# B. Messdaten auf dem Fahrzeug

Diamond - Messreihe 1

| | Sensor | | Algorithmus | | Prozentuale Abweichung |
|---|---|---|---|---|---|
| | cm / s | km / h | cm / s | km / h | |
| 1 | 30,8851 | 1,1118636 | 6,27972 | 0,22606992 | 79,6674771977 |
| 2 | 36,6209 | 1,3183524 | | | |
| 3 | 36,1774 | 1,3023864 | 8,92241 | 0,32120676 | 75,3370612592 |
| 4 | 25,9272 | 0,9333792 | 12,3275 | 0,44379 | 52,4534080039 |
| 5 | 36,1105 | 1,299978 | | | |
| 6 | 36,2377 | 1,3045572 | | | |
| 7 | 25,9188 | 0,9330768 | | | |
| 8 | 41,4232 | 1,4912352 | | | |
| 9 | 36,2422 | 1,3047192 | | | |
| 10 | 25,9006 | 0,9324216 | 2,88018 | 0,10368648 | 88,8798715088 |
| 11 | 46,1919 | 1,6629084 | 11,2353 | 0,4044708 | 75,6769043923 |
| 12 | 25,966 | 0,934776 | | | |
| 13 | 41,4769 | 1,4931684 | | | |
| 14 | 36,2538 | 1,3051368 | 2,62281 | 0,09442116 | 92,7654204525 |
| 15 | 46,8854 | 1,6878744 | | | |
| 16 | 36,2107 | 1,3035852 | | | |
| 17 | 51,8345 | 1,866042 | | | |
| 18 | 30,9959 | 1,1158524 | | | |
| 19 | 46,5769 | 1,6767684 | 5,79481 | 0,20861316 | 87,5586181133 |
| 20 | 36,185 | 1,30266 | | | |
| 21 | 36,2814 | 1,3061304 | 3,7377 | 0,1345572 | 89,6980270883 |
| 22 | 41,3986 | 1,4903496 | | | |
| 23 | 31,0362 | 1,1173032 | | | |
| 24 | 41,4037 | 1,4905332 | | | |
| 25 | 36,1359 | 1,3008924 | 28,1396 | 1,0130256 | 22,1284096978 |
| 26 | 25,9603 | 0,9345708 | | | |
| 27 | 41,345 | 1,48842 | | | |
| 28 | | | 27,6737 | 0,9962532 | |
| 29 | 25,7257 | 0,9261252 | 44,8781 | 1,6156116 | -74,4485086898 |
| 30 | 36,5959 | 1,3174524 | | | |
| 31 | 35,988 | 1,295568 | 7,70055 | 0,2772198 | 78,6024508169 |
| 32 | 30,8671 | 1,1112156 | 23,5247 | 0,8468892 | 23,7871390574 |
| 33 | 41,683 | 1,500588 | | | |
| 34 | 36,3652 | 1,3091472 | | | |
| 35 | 30,8647 | 1,1111292 | | | |
| 36 | | | 37,1805 | 1,338498 | |
| 37 | 46,5683 | 1,6764588 | 24,515 | 0,88254 | 47,3568929937 |
| 38 | 30,949 | 1,114164 | 5,3192 | 0,1914912 | 82,8130149601 |
| 39 | 41,6369 | 1,4989284 | 32,7379 | 1,1785644 | 21,3728687775 |
| 40 | 36,0938 | 1,2993768 | | | |
| 41 | 36,3809 | 1,3097124 | | | |
| 42 | 41,2295 | 1,484262 | | | |
| 43 | | | 35,3282 | 1,2718152 | |
| 44 | 31,0666 | 1,1183976 | 23,8671 | 0,8592156 | 23,1744059537 |
| 45 | 41,4516 | 1,4922576 | | | |
| 46 | 36,2308 | 1,3043088 | 28,693 | 1,032948 | 20,8049504841 |
| 47 | 36,2188 | 1,3038768 | | | |

| | | | | | |
|---|---|---|---|---|---|
| 48 | 41,4166 | 1,4909976 | 12,2573 | 0,4412628 | 70,4048618187 |
| 49 | | | 4,65865 | 0,1677114 | |
| 50 | 36,1861 | 1,3026996 | 44,6039 | 1,6057404 | -23,2625234551 |
| 51 | 46,6204 | 1,6783344 | 5,22295 | 0,1880262 | 88,7968571698 |
| 52 | 31,0869 | 1,1191284 | | | |
| 53 | 41,2509 | 1,4850324 | 36,1492 | 1,3013712 | 12,3674877397 |
| 54 | 36,4273 | 1,3113828 | | | |
| 55 | 30,9041 | 1,1125476 | | | |
| 56 | 41,4763 | 1,4931468 | 46,0832 | 1,6589952 | -11,1073070645 |
| 57 | 30,9919 | 1,1157084 | | | |
| 58 | 36,2431 | 1,3047516 | 8,86283 | 0,31906188 | 75,546159131 |
| 59 | 41,3279 | 1,4878044 | 38,0083 | 1,3682988 | 8,0323461874 |
| 60 | 31,2267 | 1,1241612 | | | |
| 61 | 14,194 | 0,510984 | | | |
| 62 | 41,5679 | 1,4964444 | 34,8476 | 1,2545136 | 16,1670423572 |
| 63 | 31,0309 | 1,1171124 | 12,8266 | 0,4617576 | 58,6650725567 |
| 64 | 41,4367 | 1,4917212 | 33,4379 | 1,2037644 | 19,3036607645 |
| 65 | 36,2531 | 1,3051116 | | | |
| 66 | 41,3085 | 1,487106 | 4,54948 | 0,16378128 | 88,9865766126 |
| 67 | 36,3187 | 1,3074732 | | | |
| 68 | 31,028 | 1,117008 | | | |
| 69 | 41,4118 | 1,4908248 | | | |
| 70 | 36,1527 | 1,3014972 | 33,6019 | 1,2096684 | 7,0556279337 |
| 71 | 46,6237 | 1,6784532 | | | |
| 72 | | | 8,69887 | 0,31315932 | |
| 73 | 31,0165 | 1,116594 | 13,8299 | 0,4978764 | 55,4111521287 |
| 74 | 46,5078 | 1,6742808 | 46,7285 | 1,682226 | -0,474544055 |
| 75 | 41,5938 | 1,4973768 | | | |
| 76 | 31,0097 | 1,1163492 | | | |
| 77 | 41,3205 | 1,487538 | 32,7261 | 1,1781396 | 20,799361092 |
| 78 | 31,1811 | 1,1225196 | | | |
| 79 | | | 6,40616 | 0,23062176 | |
| 80 | 41,1887 | 1,4827932 | 10,139 | 0,365004 | 75,3840252302 |
| 81 | | | 29,3203 | 1,0555308 | |
| 82 | 36,2372 | 1,3045392 | 45,7733 | 1,6478388 | -26,3157749495 |
| 83 | 31,2128 | 1,1236608 | | | |
| 84 | 46,3756 | 1,6695216 | 6,69133 | 0,24088788 | 85,5714427414 |
| 85 | 25,8848 | 0,9318528 | 24,4467 | 0,8800812 | 5,5557701817 |
| 86 | 36,3504 | 1,3086144 | | | |
| 87 | 41,3948 | 1,4902128 | | | |
| 88 | 25,7997 | 0,9287892 | | | |
| 89 | | | 36,2635 | 1,305486 | |
| 90 | 46,3938 | 1,6701768 | 35,3131 | 1,2712716 | 23,8840103635 |
| 91 | 36,3477 | 1,3085172 | | | |
| 92 | 41,3252 | 1,4877072 | 41,9456 | 1,5100416 | -1,5012631518 |
| 93 | 46,6653 | 1,6799508 | | | |
| 94 | | | 12,3137 | 0,4432932 | |
| 95 | 36,0864 | 1,2991104 | 16,844 | 0,606384 | 53,3231355857 |
| 96 | | | 40,3498 | 1,4525928 | |
| 97 | 41,3564 | 1,4888304 | 36,4009 | 1,3104324 | 11,9824259365 |

Diamond - Messreihe 1

| | | | | | |
|---|---|---|---|---|---|
| 98 | 36,5317 | 1,3151412 | | | |
| 99 | 46,4336 | 1,6716096 | | | |
| 100 | 31,0007 | 1,1160252 | | | |
| 101 | 36,1053 | 1,2997908 | 4,50872 | 0,16231392 | 87,5123042877 |
| 102 | 36,4132 | 1,3108752 | | | |
| 103 | 31,127 | 1,120572 | | | |
| 104 | 41,4613 | 1,4926068 | | | |
| 105 | 36,2147 | 1,3037292 | | | |
| 106 | 36,1147 | 1,3001292 | | | |
| 107 | 41,3762 | 1,4895432 | | | |
| 108 | 36,0511 | 1,2978396 | 38,8769 | 1,3995684 | -7,838318387 |
| 109 | 46,8498 | 1,6865928 | | | |
| 110 | 30,8763 | 1,1115468 | 43,3376 | 1,5601536 | -40,3587865126 |
| 111 | 41,7375 | 1,50255 | | | |
| 112 | 41,0673 | 1,4784228 | 10,5072 | 0,3782592 | 74,4146802931 |
| 113 | 31,1439 | 1,1211804 | 55,8188 | 2,0094768 | -79,2286772048 |
| 114 | 41,3535 | 1,488726 | | | |
| 115 | 31,1272 | 1,1205792 | 2,80643 | 0,10103148 | 90,9839947056 |
| 116 | 36,108 | 1,299888 | 2,01625 | 0,072585 | 94,4160573834 |
| 117 | 36,1224 | 1,3004064 | 22,4631 | 0,8086716 | 37,8139326291 |
| 118 | 25,989 | 0,935604 | | | |
| 119 | 41,4371 | 1,4917356 | 27,7367 | 0,9985212 | 33,0631245912 |
| 120 | 36,1825 | 1,30257 | 28,1721 | 1,0141956 | 22,1388792925 |
| 121 | 30,7627 | 1,1074572 | | | |
| 122 | 46,2379 | 1,6645644 | 26,5034 | 0,9541224 | 42,680355293 |
| 123 | 31,2387 | 1,1245932 | | | |
| | | | | | |
| Ø | 36,76365664 | 1,323491639 | 22,8755041 | 0,823518148 | 38,3882659077 |

| | |
|---|---|
| Prozentual: | 37,7768530369 |

Diamond - Messreihe 2

| | Sensor | | Algorithmus | | Prozentuale Abweichung |
|---|---|---|---|---|---|
| | cm / s | km / h | cm / s | km / h | |
| 1 | 41,4822 | 1,4933592 | | | |
| 2 | 41,2408 | 1,4846688 | | | |
| 3 | | | 6,83043 | 0,24589548 | |
| 4 | 20,386 | 0,733896 | 8,42109 | 0,30315924 | 58,6917982929 |
| 5 | 41,5329 | 1,4951844 | 44,1492 | 1,5893712 | -6,2993434121 |
| 6 | 20,2473 | 0,7289028 | | | |
| 7 | 20,9036 | 0,7525296 | | | |
| 8 | | | 50,1995 | 1,807182 | |
| 9 | 39,4563 | 1,4204268 | 36,3917 | 1,3101012 | 7,7670739527 |
| 10 | 44,1128 | 1,5880608 | | | |
| 11 | 41,1375 | 1,48095 | 37,2609 | 1,3413924 | 9,4235186873 |
| 12 | 41,7171 | 1,5018156 | 12,5717 | 0,4525812 | 69,8643961349 |
| 13 | 41,6843 | 1,5006348 | 44,7914 | 1,6124904 | -7,4538855156 |
| 14 | 60,3733 | 2,1734388 | 34,0197 | 1,2247092 | 43,6510841713 |
| 15 | 21,0208 | 0,7567488 | 44,882 | 1,615752 | -113,5123306439 |
| 16 | 41,9964 | 1,5118704 | 26,9398 | 0,9698328 | 35,852120658 |
| 17 | | | 13,7815 | 0,496134 | |
| 18 | 20,6266 | 0,7425576 | 67,5474 | 2,4317064 | -227,4771411672 |
| 19 | 20,7665 | 0,747594 | 45,2995 | 1,630782 | -118,1373847302 |
| 20 | | | 37,6618 | 1,3558248 | |
| 21 | 40,4479 | 1,4561244 | 9,44874 | 0,34015464 | 76,6397266607 |
| 22 | 41,0169 | 1,4766084 | 19,1199 | 0,6883164 | 53,385311908 |
| 23 | 42,6347 | 1,5348492 | | | |
| 24 | 41,6273 | 1,4985828 | 40,9385 | 1,473786 | 1,6546833448 |
| 25 | 62,222 | 2,239992 | 33,3475 | 1,20051 | 46,4056121629 |
| 26 | 41,434 | 1,491624 | 35,4624 | 1,2766464 | 14,4123183859 |
| 27 | 20,6694 | 0,7440984 | | | |
| 28 | 41,7129 | 1,5016644 | | | |
| 29 | 20,5798 | 0,7408728 | | | |
| 30 | 41,6457 | 1,4992452 | | | |
| 31 | 20,5223 | 0,7388028 | 10,978 | 0,395208 | 46,5069704663 |
| 32 | | | 24,7915 | 0,892494 | |
| 33 | 60,9433 | 2,1939588 | 46,343 | 1,668348 | 23,9571864339 |
| 34 | | | 49,7092 | 1,7895312 | |
| 35 | 42,0049 | 1,5121764 | 23,7515 | 0,855054 | 43,4554063931 |
| 36 | 41,7945 | 1,504602 | | | |
| 37 | 41,2458 | 1,4848488 | 12,8175 | 0,46143 | 68,9241086365 |
| 38 | 41,4365 | 1,491714 | | | |
| 39 | 41,2006 | 1,4832216 | 15,4197 | 0,5551092 | 62,5740887269 |
| 40 | 20,6504 | 0,7434144 | 43,5816 | 1,5689376 | -111,0448223763 |
| 41 | 20,8073 | 0,7490628 | 20,2113 | 0,7276068 | 2,8643793284 |
| 42 | 41,3257 | 1,4877252 | 61,3218 | 2,2075848 | -48,3865972022 |
| 43 | 41,5555 | 1,495998 | | | |
| 44 | 39,6726 | 1,4282136 | | | |
| 45 | | | 35,0867 | 1,2631212 | |
| 46 | 41,665 | 1,49994 | 15,5387 | 0,5593932 | 62,7056282251 |
| 47 | | | 47,2276 | 1,7001936 | |

| | | | | | |
|---|---|---|---|---|---|
| 48 | | | 11,2473 | 0,4049028 | |
| 49 | 40,6002 | 1,4616072 | 62,5946 | 2,2534056 | -54,1731321521 |
| 50 | | | 41,0984 | 1,4795424 | |
| 51 | 42,0972 | 1,5154992 | 22,4941 | 0,8097876 | 46,5662799426 |
| 52 | 41,6172 | 1,4982192 | 40,8259 | 1,4697324 | 1,9013773151 |
| 53 | 40,5239 | 1,4588604 | 35,7729 | 1,2878244 | 11,7239456222 |
| 54 | 41,8783 | 1,5076188 | 40,9898 | 1,4756328 | 2,1216238482 |
| 55 | | | 37,7036 | 1,3573296 | |
| 56 | 43,6175 | 1,57023 | 26,438 | 0,951768 | 39,3867140483 |
| 57 | 40,7319 | 1,4663484 | | | |
| 58 | 42,2173 | 1,5198228 | | | |
| 59 | 60,9277 | 2,1933972 | | | |
| 60 | 42,3357 | 1,5240852 | | | |
| 61 | | | 11,4986 | 0,4139496 | |
| 62 | 39,6659 | 1,4279724 | 11,3288 | 0,4078368 | 71,4394479893 |
| 63 | | | 52,6615 | 1,895814 | |
| 64 | 21,6445 | 0,779202 | 55,1989 | 1,9871604 | -155,025064104 |
| 65 | 41,2244 | 1,4840784 | 11,5241 | 0,4148676 | 72,0454391089 |
| 66 | 42,3833 | 1,5257988 | | | |
| 67 | | | 42,0678 | 1,5144408 | |
| 68 | 20,0445 | 0,721602 | 23,0045 | 0,828162 | -14,7671431066 |
| 69 | | | 7,48237 | 0,26936532 | |
| 70 | 21,2526 | 0,7650936 | 39,2029 | 1,4113044 | -84,4616658668 |
| 71 | | | 96,0088 | 3,4563168 | |
| 72 | 60,8966 | 2,1922776 | 48,7328 | 1,7543808 | 19,9745141765 |
| 73 | 42,8226 | 1,5416136 | | | |
| 74 | 41,2351 | 1,4844636 | 29,5307 | 1,0631052 | 28,3845558759 |
| 75 | 41,231 | 1,484316 | 31,0835 | 1,119006 | 24,6113361306 |
| 76 | | | 6,39917 | 0,23037012 | |
| 77 | 41,3199 | 1,4875164 | 26,6258 | 0,9585288 | 35,5617995203 |
| 78 | 20,4967 | 0,7378812 | 37,7562 | 1,3592232 | -84,2062380773 |
| 79 | | | 34,1664 | 1,2299904 | |
| 80 | 41,0788 | 1,4788368 | 29,909 | 1,076724 | 27,1911545615 |
| 81 | 42,1581 | 1,5176916 | 48,1215 | 1,732374 | -14,1453243861 |
| 82 | 20,6262 | 0,7425432 | | | |
| 83 | 20,5823 | 0,7409628 | 41,026 | 1,476936 | -99,3266058701 |
| 84 | 62,9472 | 2,2660992 | 56,2696 | 2,0257056 | 10,608255808 |
| 85 | 41,2113 | 1,4836068 | 34,3348 | 1,2360528 | 16,6859574922 |
| 86 | 41,1923 | 1,4829228 | | | |
| 87 | 40,9616 | 1,4746176 | | | |
| 88 | 63,9956 | 2,3038416 | | | |
| 89 | | | 11,5285 | 0,415026 | |
| 90 | 20,3824 | 0,7337664 | 14,7505 | 0,531018 | 27,6311916163 |
| 91 | | | 12,296 | 0,442656 | |
| 92 | 20,6529 | 0,7435044 | 20,2548 | 0,7291728 | 1,9275743358 |
| 93 | | | 29,1629 | 1,0498644 | |
| 94 | 20,6789 | 0,7444404 | 30,3357 | 1,0920852 | -46,6988089308 |
| 95 | | | 22,8165 | 0,821394 | |
| 96 | 20,7274 | 0,7461864 | 18,3225 | 0,65961 | 11,6025164758 |
| 97 | | | 17,7964 | 0,6406704 | |

Diamond - Messreihe 2

| | | | | | |
|---|---|---|---|---|---|
| 98 | 0 | 0 | 26,7417 | 0,9627012 | |
| Ø | 37,006112 | 1,332220032 | 31,79657568 | 1,144676724 | -0,1403278221 |
| | | | | Prozentual: | 14,0775024524 |

Diamond - Messreihe 3

| | Sensor | | Algorithmus | | Prozentuale Abweichung |
|---|---|---|---|---|---|
| | cm / s | km / h | cm / s | km / h | |
| 1 | 41,3752 | 1,4895072 | | | |
| 2 | 41,1187 | 1,4802732 | 41,8044 | 1,5049584 | -1,6676110869 |
| 3 | 41,825 | 1,5057 | 55,0233 | 1,9808388 | -31,5560071727 |
| 4 | 41,5329 | 1,4951844 | | | |
| 5 | | | 6,45399 | 0,23234364 | |
| 6 | 40,7945 | 1,468602 | 44,1515 | 1,589454 | -8,2290504847 |
| 7 | 41,8917 | 1,5081012 | 37,8341 | 1,3620276 | 9,6859282388 |
| 8 | | | 61,9036 | 2,2285296 | |
| 9 | | | 33,764 | 1,215504 | |
| 10 | 41,0063 | 1,4762268 | 27,0385 | 0,973386 | 34,0625708733 |
| 11 | 41,7743 | 1,5038748 | 41,9436 | 1,5099696 | -0,4052730985 |
| 12 | 39,6149 | 1,4261364 | 43,9375 | 1,58175 | -10,9115509568 |
| 13 | 43,6138 | 1,5700968 | | | |
| 14 | 61,5143 | 2,2145148 | 49,2815 | 1,774134 | 19,8861077831 |
| 15 | 41,1564 | 1,4816304 | | | |
| 16 | 41,3445 | 1,488402 | | | |
| 17 | 21,0267 | 0,7569612 | | | |
| 18 | 41,593 | 1,497348 | | | |
| 19 | 20,6793 | 0,7444548 | | | |
| 20 | | | 11,4178 | 0,4110408 | |
| 21 | 40,5381 | 1,4593716 | 9,17911 | 0,33044796 | 77,3568322146 |
| 22 | 20,9867 | 0,7555212 | | | |
| 23 | 61,8994 | 2,2283784 | | | |
| 24 | 41,5804 | 1,4968944 | 102,21 | 3,67956 | -145,8129310925 |
| 25 | | | 43,8735 | 1,579446 | |
| 26 | 41,0665 | 1,478394 | 74,9502 | 2,6982072 | -82,5093445996 |
| 27 | | | 13,8882 | 0,4999752 | |
| 28 | 41,0609 | 1,4781924 | 14,9529 | 0,5383044 | 63,5836038665 |
| 29 | | | 33,5585 | 1,208106 | |
| 30 | 20,9438 | 0,7539768 | 32,0706 | 1,1545416 | -53,1269397149 |
| 31 | 41,0853 | 1,4790708 | 54,6664 | 1,9679904 | -33,0558618289 |
| 32 | 20,8762 | 0,7515432 | 17,6104 | 0,6339744 | 15,6436516224 |
| 33 | 20,66 | 0,74376 | 40,4936 | 1,4577696 | -96 |
| 34 | | | 6,74242 | 0,24272712 | |
| 35 | 41,6366 | 1,4989176 | 27,3469 | 0,9844884 | 34,3200453447 |
| 36 | 41,7296 | 1,5022656 | | | |
| 37 | | | 31,1341 | 1,1208276 | |
| 38 | 40,5644 | 1,4603184 | 37,3952 | 1,3462272 | 7,8127619292 |
| 39 | | | 47,0639 | 1,6943004 | |
| 40 | 41,4671 | 1,4928156 | 59,994 | 2,159784 | -44,6785523945 |
| 41 | 41,8952 | 1,5082272 | | | |
| 42 | 41,2268 | 1,4841648 | 49,3293 | 1,7758548 | -19,6534778348 |
| 43 | 41,3983 | 1,4903388 | | | |
| 44 | 40,8897 | 1,4720292 | 44,8403 | 1,6142508 | -9,6616018215 |
| 45 | 21,2108 | 0,7635888 | | | |
| 46 | 20,6954 | 0,7450344 | | | |
| 47 | 41,4805 | 1,493298 | | | |

| | | | | | |
|---|---|---|---|---|---|
| 48 | 41,1358 | 1,4808888 | | | |
| 49 | 41,2416 | 1,4846976 | | | |
| 50 | 41,7221 | 1,5019956 | | | |
| 51 | 40,3509 | 1,4526324 | 13,5225 | 0,48681 | 66,487736333 |
| 52 | 42,6322 | 1,5347592 | | | |
| 53 | 41,0023 | 1,4760828 | 33,2545 | 1,197162 | 18,8960131505 |
| 54 | 41,3752 | 1,4895072 | 82,6752 | 2,9763072 | -99,8182486127 |
| 55 | | | 27,9259 | 1,0053324 | |
| 56 | 20,8341 | 0,7500276 | 42,1621 | 1,5178356 | -102,3706327607 |
| 57 | 20,729 | 0,746244 | | | |
| 58 | | | 13,6898 | 0,4928328 | |
| 59 | 40,6073 | 1,4618628 | 28,4078 | 1,0226808 | 30,0426278034 |
| 60 | 41,6339 | 1,4988204 | 29,6265 | 1,066554 | 28,8404401221 |
| 61 | 40,6536 | 1,4635296 | 30,7222 | 1,1059992 | 24,4293248322 |
| 62 | 62,8873 | 2,2639428 | 60,7092 | 2,1855312 | 3,4634973993 |
| 63 | | | 47,2021 | 1,6992756 | |
| 64 | 40,0155 | 1,440558 | 21,8903 | 0,7880508 | 45,2954480139 |
| 65 | | | 42,8169 | 1,5414084 | |
| 66 | 42,0595 | 1,514142 | 52,641 | 1,895076 | -25,1584065431 |
| 67 | | | 39,0318 | 1,4051448 | |
| 68 | 20,7212 | 0,7459632 | 7,05745 | 0,2540682 | 65,94092041 |
| 69 | 42,0928 | 1,5153408 | 26,979 | 0,971244 | 35,9059031473 |
| 70 | 40,8897 | 1,4720292 | 37,4686 | 1,3488696 | 8,3666546832 |
| 71 | 20,9556 | 0,7544016 | 51,6689 | 1,8600804 | -146,5636870335 |
| 72 | 41,5013 | 1,4940468 | 43,3423 | 1,5603228 | -4,4360056191 |
| 73 | 41,3677 | 1,4892372 | 21,3618 | 0,7690248 | 48,3611610024 |
| 74 | 42,0809 | 1,5149124 | | | |
| 75 | 41,0732 | 1,4786352 | | | |
| 76 | 41,1564 | 1,4816304 | | | |
| 77 | | | 9,64102 | 0,34707672 | |
| 78 | 61,7284 | 2,2222224 | 21,2506 | 0,7650216 | 65,5740307541 |
| 79 | | | 21,7145 | 0,781722 | |
| 80 | 20,4792 | 0,7372512 | 92,9647 | 3,3467292 | -353,9469315208 |
| 81 | | | 49,2407 | 1,7726652 | |
| 82 | 42,3712 | 1,5253632 | 37,3258 | 1,3437288 | 11,9076164942 |
| 83 | | | 33,2972 | 1,1986992 | |
| 84 | 20,3504 | 0,7326144 | 30,5727 | 1,1006172 | -50,2314450822 |
| 85 | | | 39,4204 | 1,4191344 | |
| 86 | 21,0524 | 0,7578864 | 44,3218 | 1,5955848 | -110,5308658395 |
| 87 | | | 40,5789 | 1,4608404 | |
| 88 | 41,3199 | 1,4875164 | 36,8247 | 1,3256892 | 10,8790195523 |
| 89 | | | 14,8774 | 0,5355864 | |
| 90 | 41,3835 | 1,489806 | 120,892 | 4,352112 | -192,1260889002 |
| 91 | | | 34,9966 | 1,2598776 | |
| 92 | 41,5664 | 1,4963904 | 57,2795 | 2,062062 | -37,8024077139 |
| 93 | 20,885 | 0,75186 | 38,0593 | 1,3701348 | -82,2327028968 |
| 94 | 20,7789 | 0,7480404 | | | |
| 95 | | | 6,43293 | 0,23158548 | |
| 96 | 20,4444 | 0,7359984 | 22,0071 | 0,7922556 | -7,643657921 |
| 97 | | | 15,0826 | 0,5429736 | |

Diamond - Messreihe 3

| | | | | | |
|---|---|---|---|---|---|
| 98 | | | 12,4744 | 0,4490784 | |
| 99 | 20,7041 | 0,7453476 | 11,3689 | 0,4092804 | 45,0886539381 |
| Ø | 37,13063151 | 1,336702734 | 37,03558 | 1,33328088 | -20,3812236046 |

| | |
|---|---|
| Prozentual: | 0,2559921633 |

Diamond - Messreihe 4

| | Sensor | | Algorithmus | | Prozentuale Abweichung |
|---|---|---|---|---|---|
| | cm / s | km / h | cm / s | km / h | |
| 1 | 20,9201 | 0,7531236 | | | |
| 2 | 20,25 | 0,729 | 50,1734 | 1,8062424 | -147,7698765432 |
| 3 | 41,8308 | 1,5059088 | 35,6572 | 1,2836592 | 14,7585033038 |
| 4 | 41,4124 | 1,4908464 | 41,1827 | 1,4825772 | 0,5546647864 |
| 5 | 41,7137 | 1,5016932 | 52,0569 | 1,8740484 | -24,7956906244 |
| 6 | 41,5312 | 1,4951232 | | | |
| 7 | | | 16,0554 | 0,5779944 | |
| 8 | 40,9601 | 1,4745636 | 9,01444 | 0,32451984 | 77,9921435739 |
| 9 | 41,4588 | 1,4925168 | | | |
| 10 | 41,8342 | 1,5060312 | | | |
| 11 | 20,4635 | 0,736686 | 32,4938 | 1,1697768 | -58,7890634544 |
| 12 | 41,8173 | 1,5054228 | 33,0057 | 1,1882052 | 21,0716617285 |
| 13 | 20,0523 | 0,7218828 | 7,10407 | 0,25574652 | 64,5722934526 |
| 14 | 21,2408 | 0,7646688 | 22,3781 | 0,8056116 | -5,3543181048 |
| 15 | 41,2244 | 1,4840784 | 4,2803 | 0,1540908 | 89,6170714431 |
| 16 | 41,7474 | 1,5029064 | 48,4817 | 1,7453412 | -16,1310644495 |
| 17 | 41,172 | 1,482192 | 75,5271 | 2,7189756 | -83,4428737977 |
| 18 | 41,5845 | 1,497042 | 58,0502 | 2,0898072 | -39,5957628443 |
| 19 | 61,9587 | 2,2305132 | 11,0926 | 0,3993336 | 82,0967838254 |
| 20 | 41,226 | 1,484136 | 55,9874 | 2,0155464 | -35,8060447291 |
| 21 | | | 2,72353 | 0,09804708 | |
| 22 | | | 3,407 | 0,122652 | |
| 23 | 20,5333 | 0,7391988 | 23,5979 | 0,8495244 | -14,9250242289 |
| 24 | | | 126,164 | 4,541904 | |
| 25 | 41,718 | 1,501848 | 42,9874 | 1,5475464 | -3,0428112565 |
| 26 | | | 34,0064 | 1,2242304 | |
| 27 | 20,6476 | 0,7433136 | 11,8846 | 0,4278456 | 42,4407679343 |
| 28 | | | 36,1948 | 1,3030128 | |
| 29 | 20,7506 | 0,7470216 | 54,6331 | 1,9667916 | -163,2844351489 |
| 30 | 41,4696 | 1,4929056 | 54,617 | 1,966212 | -31,7037058472 |
| 31 | 62,4094 | 2,2467384 | 47,835 | 1,72206 | 23,3528923528 |
| 32 | 41,2416 | 1,4846976 | 45,2727 | 1,6298172 | -9,7743540503 |
| 33 | 41,5228 | 1,4948208 | 48,4404 | 1,7438544 | -16,6597628291 |
| 34 | 41,9079 | 1,5086844 | | | |
| 35 | 41,3018 | 1,4868648 | | | |
| 36 | 41,3075 | 1,48707 | | | |
| 37 | 41,2465 | 1,484874 | | | |
| 38 | | | 47,5822 | 1,7129592 | |
| 39 | | | 64,7015 | 2,329254 | |
| 40 | | | 19,9542 | 0,7183512 | |
| 41 | 62,1309 | 2,2367124 | 32,7786 | 1,1800296 | 47,2426763494 |
| 42 | 41,3454 | 1,4884344 | 127,644 | 4,595184 | -208,7260009578 |
| 43 | 41,8605 | 1,506978 | | | |
| 44 | 40,6248 | 1,4624928 | 41,5935 | 1,497366 | -2,3845040468 |
| 45 | 41,8867 | 1,5079212 | 39,026 | 1,404936 | 6,8296141735 |
| 46 | 62,0243 | 2,2328748 | 42,0892 | 1,5152112 | 32,1407899807 |
| 47 | 20,8501 | 0,7506036 | | | |

| | | | | | |
|---|---|---|---|---|---|
| 48 | 40,9577 | 1,4744772 | 22,8234 | 0,8216424 | 44,2756795426 |
| 49 | 20,7062 | 0,7454232 | 44,1877 | 1,5907572 | -113,4032318822 |
| 50 | | 0 | 2,83871 | 0,10219356 | |
| 51 | 20,688 | 0,744768 | 21,9891 | 0,7916076 | -6,2891531323 |
| 52 | 41,8655 | 1,507158 | 45,0614 | 1,6222104 | -7,6337318317 |
| 53 | 41,7186 | 1,5018696 | | | |
| 54 | 40,6241 | 1,4624676 | 60,5277 | 2,1789972 | -48,9945623411 |
| 55 | 41,4547 | 1,4923692 | 46,71 | 1,68156 | -12,6772115104 |
| 56 | 41,6364 | 1,4989104 | 40,1815 | 1,446534 | 3,4942982583 |
| 57 | 61,5997 | 2,2175892 | 44,0057 | 1,5842052 | 28,5618274115 |
| 58 | 21,0738 | 0,7586568 | | | |
| 59 | 41,2244 | 1,4840784 | | | |
| 60 | 20,5557 | 0,7400052 | 5,33969 | 0,19222884 | 74,0233122686 |
| 61 | 20,643 | 0,743148 | 89,7485 | 3,230946 | -334,7648113162 |
| 62 | | | 11,3957 | 0,4102452 | |
| 63 | 40,9975 | 1,47591 | 135,123 | 4,864428 | -229,5883895359 |
| 64 | | | 110,995 | 3,99582 | |
| 65 | 20,8028 | 0,7489008 | 46,4826 | 1,6733736 | -123,443959467 |
| 66 | | | 17,1404 | 0,6170544 | |
| 67 | 61,9575 | 2,23047 | 89,1761 | 3,2103396 | -43,9310817899 |
| 68 | | | 72,5972 | 2,6134992 | |
| 69 | 20,8224 | 0,7496064 | 31,7791 | 1,1440476 | -52,6197748578 |
| 70 | 41,7246 | 1,5020856 | 31,1013 | 1,1196468 | 25,4605196934 |
| 71 | 39,8414 | 1,4342904 | 5,0087 | 0,1803132 | 87,4284036204 |
| 72 | 41,924 | 1,509264 | 30,6134 | 1,1020824 | 26,978818815 |
| 73 | 41,5037 | 1,4941332 | 67,2859 | 2,4222924 | -62,1202447011 |
| 74 | 20,3268 | 0,7317648 | 118,821 | 4,277556 | -484,5533974851 |
| 75 | 21,3882 | 0,7699752 | | | |
| 76 | 20,4744 | 0,7370784 | 10,0489 | 0,3617604 | 50,9196850701 |
| 77 | | | 9,3923 | 0,3381228 | |
| 78 | 41,841 | 1,506276 | 42,129 | 1,516644 | -0,6883200688 |
| 79 | | | 31,852 | 1,146672 | |
| 80 | 40,7929 | 1,4685444 | 40,387 | 1,453932 | 0,9950260952 |
| 81 | 41,7339 | 1,5024204 | 74,4138 | 2,6788968 | -78,3054063962 |
| 82 | | | 30,1549 | 1,0855764 | |
| 83 | 41,1179 | 1,4802444 | 95,7511 | 3,4470396 | -132,8696261239 |
| 84 | 20,8195 | 0,749502 | 49,8535 | 1,794726 | -139,4557986503 |
| 85 | 41,4266 | 1,4913576 | | | |
| 86 | 41,1571 | 1,4816556 | 11,3808 | 0,4097088 | 72,3479059506 |
| 87 | 20,724 | 0,746064 | 38,1471 | 1,3732956 | -84,0720903301 |
| 88 | 41,4847 | 1,4934492 | 35,05 | 1,2618 | 15,5110197253 |
| 89 | 20,7673 | 0,7476228 | | | |
| 90 | 21,0747 | 0,7586892 | | | |
| 91 | 40,9456 | 1,4740416 | | | |
| 92 | 20,5369 | 0,7393284 | | | |
| 93 | 61,6376 | 2,2189536 | 32,237 | 1,160532 | 47,6991316988 |
| 94 | | | 40,0073 | 1,4402628 | |
| 95 | 41,9921 | 1,5117156 | 48,2615 | 1,737414 | -14,9299511099 |
| 96 | 20,8766 | 0,7515576 | 47,2698 | 1,7017128 | -126,4248009733 |
| 97 | 62,1982 | 2,2391352 | | | |

Diamond - Messreihe 4

| | | | | | |
|---|---|---|---|---|---|
| 98 | 20,799 | 0,748764 | | | |
| 99 | 41,4597 | 1,4925492 | 16,9756 | 0,6111216 | 59,0551788846 |
| 100 | 20,7038 | 0,7453368 | | | |
| 101 | 20,6926 | 0,7449336 | 29,2894 | 1,0544184 | -41,5452867209 |
| 102 | 41,2202 | 1,4839272 | 58,4163 | 2,1029868 | -41,7176529954 |
| 103 | 41,6482 | 1,4993352 | | | |
| 104 | 20,6897 | 0,7448292 | 63,0261 | 2,2689396 | -204,6254899781 |
| | | | | | |
| Ø | 36,72450357 | 1,306528221 | 43,0559605 | 1,550014578 | -34,5826981286 |
| | | | | Prozentual: | -17,2404152891 |

Diamond - Messreihe 5

| | Sensor | | Algorithmus | | Prozentuale Abweichung |
|---|---|---|---|---|---|
| | cm / s | km / h | cm / s | km / h | |
| 1 | 42,5951 | 1,5334236 | | | |
| 2 | 40,7762 | 1,4679432 | | | |
| 3 | 42,2493 | 1,5209748 | 31,3122 | 1,1272392 | 25,8870561169 |
| 4 | 40,5032 | 1,4581152 | 48,572 | 1,748592 | -19,9213889273 |
| 5 | 42,3037 | 1,5229332 | 42,2589 | 1,5213204 | 0,1059009023 |
| 6 | | | 8,0766 | 0,2907576 | |
| 7 | 40,213 | 1,447668 | 39,4355 | 1,419678 | 1,9334543556 |
| 8 | 21,5167 | 0,7746012 | | | |
| 9 | 20,1503 | 0,7254108 | | | |
| 10 | 42,4033 | 1,5265188 | 12,0911 | 0,4352796 | 71,4854740079 |
| 11 | | | 43,8723 | 1,5794028 | |
| 12 | 40,1731 | 1,4462316 | 128,359 | 4,620924 | -219,5147997043 |
| 13 | 42,8305 | 1,541898 | | | |
| 14 | | | 45,8501 | 1,6506036 | |
| 15 | 59,8231 | 2,1536316 | 86,3023 | 3,1068828 | -44,262500606 |
| 16 | 42,8296 | 1,5418656 | 48,1068 | 1,7318448 | -12,3213852102 |
| 17 | 60,7691 | 2,1876876 | | | |
| 18 | | | 17,4841 | 0,6294276 | |
| 19 | 20,9569 | 0,7544484 | 35,7349 | 1,2864564 | -70,5161545839 |
| 20 | 40,644 | 1,463184 | 31,8836 | 1,1478096 | 21,5539809074 |
| 21 | 21,519 | 0,774684 | | | |
| 22 | 40,9991 | 1,4759676 | | | |
| 23 | 40,5484 | 1,4597424 | 41,6973 | 1,5011028 | -2,8334040307 |
| 24 | 42,4686 | 1,5288696 | | | |
| 25 | | | 5,54566 | 0,19964376 | |
| 26 | 40,1334 | 1,4448024 | 28,3229 | 1,0196244 | 29,4281072623 |
| 27 | 42,2743 | 1,5218748 | 41,6861 | 1,5006996 | 1,3913890946 |
| 28 | 61,0679 | 2,1984444 | 34,1113 | 1,2280068 | 44,1420124157 |
| 29 | 21,2413 | 0,7646868 | 10,9227 | 0,3932172 | 48,5780060542 |
| 30 | 40,402 | 1,454472 | 36,9285 | 1,329426 | 8,597346666 |
| 31 | 42,4572 | 1,5284592 | 46,3466 | 1,6684776 | -9,1607548307 |
| 32 | 20,2156 | 0,7277616 | 45,8194 | 1,6494984 | -126,6536734007 |
| 33 | 42,3227 | 1,5236172 | 33,4467 | 1,2040812 | 20,9721969534 |
| 34 | 40,8421 | 1,4703156 | | | |
| 35 | 41,9869 | 1,5115284 | 13,9484 | 0,5021424 | 66,779162072 |
| 36 | 40,7433 | 1,4667588 | | | |
| 37 | 63,7302 | 2,2942872 | | | |
| 38 | | | 8,05212 | 0,28987632 | |
| 39 | 40,8196 | 1,4695056 | 41,0701 | 1,4785236 | -0,6136757832 |
| 40 | | | 52,4437 | 1,8879732 | |
| 41 | 39,9777 | 1,4391972 | 32,1624 | 1,1578464 | 19,5491486504 |
| 42 | 43,5332 | 1,5671952 | | | |
| 44 | 40,4005 | 1,454418 | 38,3035 | 1,378926 | 5,1905298202 |
| 45 | 42,6883 | 1,5367788 | | | |
| 46 | 62,2955 | 2,242638 | | | |
| 47 | 39,9908 | 1,4396688 | 42,0009 | 1,5120324 | -5,0264060734 |
| 48 | 42,9159 | 1,5449724 | | | |

| | | | | | |
|---|---|---|---|---|---|
| 49 | 61,4877 | 2,2135572 | 1,96403 | 0,07070508 | 96,8058164478 |
| 50 | | | 41,6272 | 1,4985792 | |
| 51 | 20,1268 | 0,7245648 | 11,7051 | 0,4213836 | 41,8432140231 |
| 52 | 43,0567 | 1,5500412 | | | |
| 53 | 20,4461 | 0,7360596 | 61,7014 | 2,2212504 | -201,7758888003 |
| 54 | 39,9707 | 1,4389452 | 18,8214 | 0,6775704 | 52,9120080459 |
| 55 | 43,2213 | 1,5559668 | | | |
| 56 | 41,3314 | 1,4879304 | | | |
| 57 | | | 6,2993 | 0,2267748 | |
| 58 | | | 11,7591 | 0,4233276 | |
| 59 | 39,9629 | 1,4386644 | 3,61679 | 0,13020444 | 90,9496307826 |
| 60 | 43,1538 | 1,5535368 | | | |
| 61 | 41,131 | 1,480716 | 6,60231 | 0,23768316 | 83,9480926795 |
| 62 | | | 34,4672 | 1,2408192 | |
| 63 | 40,0378 | 1,4413608 | 40,6644 | 1,4639184 | -1,5650210551 |
| 64 | 43,2811 | 1,5581196 | | | |
| 65 | | | 47,5913 | 1,7132868 | |
| 66 | 40,8228 | 1,4696208 | 32,3798 | 1,1656728 | 20,6820698237 |
| 67 | 40,6823 | 1,4645628 | 6,08888 | 0,21919968 | 85,033097932 |
| 68 | 42,6726 | 1,5362136 | | | |
| 69 | 41,1392 | 1,4810112 | 31,9383 | 1,1497788 | 22,3652866366 |
| 70 | 60,5477 | 2,1797172 | 25,017 | 0,900612 | 58,6821629889 |
| 71 | 42,302 | 1,522872 | 30,8497 | 1,1105892 | 27,0727152381 |
| 72 | | | 7,84884 | 0,28255824 | |
| 73 | | | 4,85126 | 0,17464536 | |
| 74 | 40,317 | 1,451412 | 49,3496 | 1,7765856 | -22,4039487065 |
| 75 | 21,5755 | 0,776718 | | | |
| 76 | 41,1474 | 1,4813064 | 14,6853 | 0,5286708 | 64,3105032153 |
| 77 | 41,0462 | 1,4776632 | | | |
| 78 | 20,9599 | 0,7545564 | | | |
| 79 | 41,3462 | 1,4884632 | | | |
| 80 | 41,3273 | 1,4877828 | | | |
| 81 | | | 11,2398 | 0,4046328 | |
| 82 | 40,1668 | 1,4460048 | 21,4963 | 0,7738668 | 46,4824183156 |
| 83 | 42,9452 | 1,5460272 | | | |
| 84 | 40,9002 | 1,4724072 | 12,5581 | 0,4520916 | 69,295749165 |
| 85 | 62,0058 | 2,2322088 | | | |
| 86 | 20,9909 | 0,7556724 | | | |
| 87 | 41,4207 | 1,4911452 | | | |
| 88 | 20,6963 | 0,7450668 | | | |
| 89 | 40,1109 | 1,4439924 | 4,43279 | 0,15958044 | 88,9486648268 |
| 90 | 42,85 | 1,5426 | | | |
| 91 | 41,0023 | 1,4760828 | 3,93746 | 0,14174856 | 90,3969777305 |
| 92 | 40,2238 | 1,4480568 | 15,378 | 0,553608 | 61,7689029878 |
| 93 | 42,6461 | 1,5352596 | 28,6781 | 1,0324116 | 32,7532881084 |
| 94 | 61,6964 | 2,2210704 | | | |
| 95 | 20,9774 | 0,7551864 | | | |
| 96 | 41,2244 | 1,4840784 | | | |
| 97 | 41,7061 | 1,5014196 | | | |
| 98 | 41,4514 | 1,4922504 | | | |

Diamond - Messreihe 5

| | | | | | |
|---|---|---|---|---|---|
| 99 | 39,959 | 1,438524 | 6,40173 | 0,23046228 | 83,9792537351 |
| Ø | 40,39009277 | 1,45404334 | 29,4349695 | 1,059658902 | 16,6056581389 |

| | |
|---|---|
| Prozentual: | 27,1232931629 |

# C. Rechenzeitanalysen

## *Rechenzeitanalyse für Notebook mit verschiedenen Testbildern:*

Rechner – Spezifikation

- Acer Aspire 5680
- Hauptspeicher: 3072 MB
- Betriebssystem: Windows 8 Professional x86

Ausführungs – Spezifikation

- Virtuelle Maschine
- Ausführung in: Oracle VM VirtualBox Manager
- Hauptspeicher: 1536 MB
- Betriebssystem: Linux Echtzeitbetriebssystem

Testimages_22

| | |
|---|---|
| Durchschnittliche Zeit für Berechnung: | 27,6482444 Millisekunden |
| davon | |
| doppelte Erkennung: | 30,9899655 Millisekunden |
| einfache Erkennung: | 26,7352539 Millisekunden |

Testimages_Diamond

| | |
|---|---|
| Durchschnittliche Zeit für Berechnung: | 8,4001496 Millisekunden |
| davon | |
| doppelte Erkennung: | 7,5046617 Millisekunden |
| einfache Erkennung: | 8,6986456 Millisekunden |

## *Rechenzeitanalyse für UNI-Rechner mit verschiedenen Testbildern:*

Rechner – Spezifikation

- HP Compaq dc 7800
- Hauptspeicher: 2048 MB
- Betriebssystem: Windows 7 Professional x64

Ausführungs – Spezifikation

- Virtuelle Maschine
- Ausführung in: Oracle VM VirtualBox Manager
- Hauptspeicher: 1024 MB
- Betriebssystem: Linux

Testimages_22

| | |
|---|---|
| Durchschnittliche Zeit für Berechnung: | 2,1988131 Millisekunden |
| davon | |
| doppelte Erkennung: | 3,4938564 Millisekunden |
| einfache Erkennung: | 1,8734368 Millisekunden |

Testimages_Diamond

| | |
|---|---|
| Durchschnittliche Zeit für Berechnung: | 6,9072342 Millisekunden |
| davon | |
| doppelte Erkennung: | 8,2361483 Millisekunden |
| einfache Erkennung: | 6,4642628 Millisekunden |

*Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.*

Hamburg, 12. April 2013 Bastian Meiners