

Universität Stuttgart



BOSCH

Spezialisierung einer Unfallerkennung am Zweirad mittels Smartphone auf Unabhängigkeit der Trageposition

Masterarbeit

des Studiengangs Medizintechnik an der
Universität Stuttgart

Cand. M.Sc. Oays Darwish

3480821

Prüfer:

Dr. Thomas Günther

Betreuer:

Dipl.-Ing. Nino Häberlen

Abgabedatum:

15.12.2022



Masterarbeit

Spezialisierung einer Unfallerkennung am Zweirad mittels Smartphone auf Unabhängigkeit der Trageposition

Unternehmensbeschreibung

Bosch.IO zählt weltweit über 800 Beschäftigte aus den Bereichen Digitalisierung und IoT und arbeitet mit 30.000 Fachleuten bei ganz Bosch zusammen. Bosch baut und liefert jährlich über eine Milliarde Geräte („Dinge“) an Kunden aus den unterschiedlichsten Branchen.

Bosch.IO GmbH
Postfach 30 02 40
70442 Stuttgart
GERMANY
Besucher:
Grönerstraße 5/1
71636 Ludwigsburg
Telefon 0711 811-0
www.bosch.io

18. März 2022

Aufgabenbeschreibung

Für die breite Anwendung einer Smartphone-basierten Unfallerkennung (u.a. Beschleunigungs-, GPS-Sensor) sollen verschiedene Trage- bzw. Transportpositionen auf ihre Eignung zur Detektion von Unfällen untersucht werden.

- Analyse Unfallarten beim Motorrad aus der Unfallstatistik
- Korrelation Unfallarten zu Verletzungsschwere
- Analyse der Übertragbarkeit statistischer Daten auf bestehenden Messdatensatz
- Analyse der Merkmale verschiedener Positionen: Worin unterscheiden sich verschiedene Positionen voneinander?
- Einfluss verschiedener Messorte auf die Qualität der Erkennung von Unfallklassen
- Vergleich Messung mit Smartphone am Fahrer (z.B. Jackentasche) vs. Smartphone am Fahrzeug (z.B. Tankrucksack)
- Spezialisierung der Help Connect-Unfallerkennung zu einer robusten Multipositions-Unfallerkennung

Das Ziel ist es, eine Unfallerkennung zu entwickeln, die weitgehend unabhängig von Trage- bzw. Messpositionen an Fahrer und Zweirad funktioniert.

Kontakt

Nino Haeberlen
Bosch.IO GmbH, IOB/PAC2
Telefon +49 173 1756137
Nino.Haeberlen@bosch.io

Kurzzusammenfassung

DE

Abstract

Englisch

Danksagung

An erster Stelle möchte ich meinen Eltern, meiner Frau und Familie danken, die mir mein Studium durch ihre Unterstützung ermöglicht haben und stets ein offenes Ohr für mich hatten. Ein besonderer Dank gilt Herrn Nino Häberlen für seine fachliche Unterstützung. Herrn Dr. Thomas Günter möchte ich dafür danken, dass er die Arbeit betreut hat. Darüber hinaus gilt mein Dank allen Personen, die beim Korrekturlesen meiner Abschlussarbeit tätig waren. Abschließend möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Arbeit unterstützt und motiviert haben.

Stuttgart, im November 2022

Oays Darwish

Inhaltsverzeichnis

Kurzzusammenfassung	v
Abstract	v
Danksagung	vii
Abbildungsverzeichnis	xi
Tabellenverzeichnis	xiii
Abkürzungsverzeichnis	xv
Symbolverzeichnis	xvii
1 Einleitung	1
2 Grundlagen	3
2.1 Unfall	3
2.1.1 Definition	3
2.1.2 Zeitliche Phasen eines Unfalls	4
2.1.3 Statistische Zahlen über Motorradunfälle	5
2.1.4 Mechanik der Motorradfahrt	10
2.2 Technische Grundlagen	12
2.2.1 Sensoren und Signale der Smartphones	12
2.2.2 Integrierte Aktivitätserkennung	14
2.2.3 Matlab/Simulink	14
2.3 Mathematische Grundlagen	15
2.3.1 Fast Fourier Transform (FFT)	16
2.4 Agile Softwareentwicklung	17
2.5 Unfallerkennungsalgorithmus	18
2.5.1 Kalibrierung	18
2.5.2 Übersicht der bereits erkennbaren Unfälle	19
2.5.3 1. Komponente: TipOver	21
2.5.4 2. Komponente: GroundHit	22
2.5.5 3. Komponente: CollisionHit	24
3 Unfallerkennung im Pocket-Mode	27
3.1 Grund des Pocket-Modes	27

3.2	Kritische Szenarien	28
3.3	Lauferkennung	29
3.3.1	Lauferkennung - Spitzendzähler	30
3.3.2	Frequenzbasierte Lauferkennung	34
3.4	Auf- und Absteigen	45
3.5	Anhalten	45
3.6	Verifikation des Algorithmus'	46
3.6.1	Versuchsplannung	46
3.6.2	Referenz-Aktivitätsdaten (Ground truth)	48
4	Ergebnisse	51
4.1	Verifikationsversuche	51
4.2	Lauferkennung	51
4.3	Verschiedene Fahrerpositionierung	52
4.4	Anhalten	54
4.5	Vor- und Nachteile	55
5	Ausblick	57
Literaturverzeichnis		xix
A	Anhang	xxiii
A.1	Entscheidungsfunktion	xxiii
Eidesstattliche Erklärung		xxv

Abbildungsverzeichnis

2.1	Einfache Darstellung des Regelkreises „Fahrer-Fahrzeug-Umfeld“ [H. Appel, 2002]	4
2.2	Beispiel der zeitlichen Phasen einer kritischen Fahrsituation [H. Appel, 2002]	5
2.3	Geschwindigkeitsüberschreitungen nach PKW- und Motorradfahrer an sieben Messstationen[d. Maire, 2020]	6
2.4	Unfallschwere über Geschwindigkeit aus der GAIDA-Datenbank[d. Maire, 2020]	7
2.5	Anzahl der Unfälle gegenüber Unfallgegner	8
2.6	Anteil der aktiven sowie passiven Unfälle	9
2.7	Anzahl der Unfälle über der Art der Ursache	9
2.8	Vereinfachte Darstellung der Kräfte bei stationärer Kurvenfahrt)	11
2.9	Verschiedene Techniken bei Kurvenfahrt mit gleicher Geschwindigkeit .	11
2.10	Beispiel von einer FFT (Zeitbereich und Frequenzbereich)[NTI-Audio, 2019]	16
2.11	Achsenrichtung in Sensorframe sowie in Bikeframe [Schnee u. a., 2020] .	19
2.12	Entscheidungsbaum des Unfallerkennungsalgorithmus'[Schnee u. a., 2021]	20
2.13	Definition der IDs aus der Abbildung 2.12[Schnee u. a., 2021]	21
2.14	Beispiel eines Fahrrads beim Umkippen mit einer ursprünglichen aufrechten Position sowie Neigung	22
2.15	Verlauf der Energie sowie des Energieschwellwerts bei einer Testfahrt ohne GroundHit	23
2.16	Verlauf der Energie sowie des Energieschwellwerts bei einer Echtfahrt mit GroundHit	24
2.17	Verlauf der Energie sowie des Energieschwellwerts bei einer Echtfahrt ohne CollisionHit	25
2.18	Verlauf der Energie sowie des Energieschwellwerts bei einer Echtfahrt ohne CollisionHit	26
3.1	Ergebnisse der Umfrage vom Spiegelinstitute über das Taschenmodus .	28
3.2	Die Use- und Edgecases mit der erwarteten Reaktion des Algorithmus'	29
3.3	Beispiel: Beschleunigungssignal beim Laufen	30
3.4	Testmodell der Lauferkennung - Spitzenzähler	31
3.5	Darstellung des im Testmodell der Lauferkennung generierten Sinussignal sowie jede Überschneidung der x-Achse	32
3.6	Skizze eines einfaches ideales Signal sowie eines komplexeres Signal .	33
3.7	Testbeispiel - Frequenzbasierte Lauferkennung - Sinussignal	35

3.8	Testbeispiel - frequenzbasierte Lauferkennung - Ausgabe des Spektrum-Analyzers im Fall eines komplexen Signals	35
3.9	Testbeispiel - Frequenzbasierte Lauferkennung - FFT	37
3.10	verschiedene Einzelsignale ($f = 23,8; f = 47,8; f = 1,1$) mit deren Summe f_g	38
3.11	Das Ergebnis der FFT - Spiegelung entfernt und Beträge	38
3.12	Ablaufschema des Testmodells der Frequenzbasierten Lauferkennung . .	39
3.13	Das Echtmodell der Frequenzbasierten Lauferkennung	41
3.14	Ein- und Ausgänge des Entscheidungsskripts	41
3.15	Entscheidungsbaum der Lauferkennung	42
3.16	Abgeschnittene Teile eines Beispieldesignals	44
3.17	Winkeländerung im Signal zwischen vertikaler sowie horizontaler Smartphone-Positionierung	46
3.18	Die Zusammenfassung der getesteten Szenarien	47
3.19	Beispiel der exportierten Tabelle mit der neuen Spalte	49
3.20	Die definierte Labels von Groundtruth	50
4.1	Ergebnis des Lauferkennungsmodells	52
4.2	Vergleich der Fahrt in vertikaler und normaler Position	53
4.3	Winkeländerung des Smartphones durch das Anhalten	54
4.4	Beinpositionen während einer Fahrt und gestreckt	55

Tabellenverzeichnis

3.1 Ausgangsmöglichkeiten der Entscheidungsfunktion	42
---	----

Abkürzungsverzeichnis

VSS	Verkehrssicherheitsscreening
GIDAS	German In-Depth Accident Study
MEMS	Mikro Elektronisch Mechanischen Systeme
FP	Frontalpanel (LabVIEW)
BD	Blockdiagramm (LabVIEW)
FFT	Fast-Fourier-Transformation
DFT	Diskrete-Fourier-Transformation
SF	Sensor Frame
BF	Bike Frame
NHTSA	National Highway Traffic Safety Administration

Symbolverzeichnis

f_s	Abtastfrequenz
B_l	Blocklänge
f_n	Bandbreite
D	Messdauer
d_f	Frequenzauflösung
λ	Neigungswinkel
F_q	Zentrifugalkraft
F_G	Gewichtskraft

1 Einleitung

Jedes Jahr wird weltweit das Leben von etwa 1,3 Millionen Menschen durch einen Verkehrsunfall beendet. Zwischen 20 und 50 Millionen weitere Menschen erleiden nicht-tödliche Verletzungen, wobei viele infolge ihrer Verletzung eine Behinderung erleben müssen. Verkehrsunfälle verursachen erhebliche wirtschaftliche Verluste für Einzelpersonen, ihre Familien und Nationen insgesamt. Diese Verluste ergeben sich aus den Behandlungskosten sowie aus Produktivitätsverlusten für diejenigen, die durch ihre Verletzungen getötet oder behindert wurden, und für Familienmitglieder, die sich von der Arbeit oder Schule freinehmen müssen, um sich um die Verletzten zu kümmern. Straßenverkehrsunfälle kosten die meisten Länder 3% ihres Bruttoinlandsprodukts. Verletzungen im Straßenverkehr sind die häufigste Todesursache für Kinder und junge Erwachsene im Alter von 5 bis 29 Jahren.[WHO, 2022]

Motorradfahren findet in Deutschland immer mehr Zuwachs. Rund 3,8 Millionen zweirädrige Kraftfahrzeuge waren am 1. Januar 2012 in Deutschland zugelassen. Dies entspricht 7,26 % aller zugelassener Kraftfahrzeuge in Deutschland [Hädrich, 2012]. Motorradfahrer stellen im Straßenverkehr eine besonders gefährdete Gruppe an Verkehrsteilnehmern dar. Die NHTSA schätzt, dass Motorradfahrer im Jahr 2018 etwa 27-mal häufiger bei Verkehrsunfällen in den USA ums Leben kamen als Insassen von PKWs[Venkatraman u. a., 2021].

Verzögerungen bei der Erkennung und Behandlung der an einem Verkehrsunfall beteiligten Personen erhöhen die Schwere der Verletzungen. Die Versorgung von Verletzungen nach einem Unfall ist äußerst zeitkritisch. Verzögerungen von Minuten können über Leben und Tod entscheiden. Die Verbesserung der Versorgung nach einem Unfall erfordert die Sicherstellung des Zugangs zu rechtzeitiger präklinischer Versorgung und die Verbesserung der Qualität sowohl der präklinischen als auch der stationären Versorgung.[WHO, 2022]

Da die Reaktionszeit der Rettung eine besonders große Rolle dabei spielt, Leben zu retten, ist eine automatische Unfallerkennung von großen Bedeutung. **** Connected life erklären ****

- ↴ Unfallerkennungsalgorithmus;
- ↴ Automatische Unfallerkennung
- ↴ kürzere Reaktionszeit; Genauere Informationen mitgeteilt; usw.
- ↴ kurze Versorgungszeit - ↴ Leben retten!!

- Geschwindigkeitsüberschreitung bei Motorrädern viel häufiger als bei Autos.

Diese Arbeit beschäftigt sich mit der Weiterentwicklung eines Unfallerkennungsalgorithmus mittels Smartphone.

- Pocket-Mode
- Edgecases
- Mögliche Maßnahmen
-

2 Grundlagen

In diesem Kapitel werden Grundlagen vorgestellt, die zum Verständnis dieser Arbeit dienen.

ODER:

In diesem Kapitel wird die Relevanz der vorliegenden Arbeit erörtert.

2.1 Unfall

In diesem Unterkapitel wird ein Unfall definiert, die Ablaufphasen eines Unfalls erläutert sowie eine Unfallstatistik präsentiert.

2.1.1 Definition

Straßenverkehrsunfälle können in der Regel nur unter Berücksichtigung des geschlossenen Regelkreises „Fahrer-Fahrzeug-Umfeld“ erklärt, analysiert und bewertet werden. Denn die Ursachen und Folgen von Unfällen lassen sich fast nie allein auf eine Komponente des Regelkreises zurückführen, sondern sind das Ergebnis des Zusammenspiels dreier Komponenten. Unfälle werden daher fast immer durch eine Kombination von Ursachen (z.B. Blendung entgegenkommender Fahrzeuge und Fußgänger in dunkler Kleidung) und deren Auswirkung auf das Zusammenspiel mehrerer Situationen (z.B. Tragen von Schutzhelmen, Auslösen von Sicherheitsairbags, Aufpralleinwirkung) verursacht. [H. Appel, 2002]



Bild 2.1 Einfache Darstellung des Regelkreises „Fahrer-Fahrzeug-Umfeld“

Abbildung 2.1: Einfache Darstellung des Regelkreises „Fahrer-Fahrzeug-Umfeld“ [H. Appel, 2002]

Die Abbildung 2.1 zeigt einen vereinfachten Regelkreis des Verhaltens zwischen den drei Komponenten (Fahrer-Fahrzeug-Umfeld). In dem Modell wurde die Ablenkung als ein Störgrößenbeispiel an den Fahrer und die nasse Straße als eine Störgröße ans Fahrzeug modelliert. Dieses Modell macht es leichter den Ablauf eines Unfalls zu verstehen und anschließend weitere Unfälle zu vermeiden. Durch eine Störung des Fahrers beziehungsweise des Fahrzeugs ändert sich der Ablauf einer Fahrt (oder eines Unfalls), da Störungen die Reaktionszeit sowie Reaktionsart der Fahrer stark beeinflussen. Diese sind im Fall eines Unfalls von großer Bedeutung.

2.1.2 Zeitliche Phasen eines Unfalls

Nach dem zeitlichen Unfallverlauf werden folgende Unfallphasen unterschieden:

- Pre-Crash-Phase (Einlaufphase): Die Einlaufphase beschreibt den Zeitraum vom Erkennen der kritischen Situation bis zum ersten Kontakt mit dem Hindernis beziehungsweise Unfallgegner.
- Crash (Kollisionsphase): Der Zeitraum vom ersten Kontakt zwischen den Unfallbeteiligten bis zur Lösung. Bei Mehrfachkollisionen werden mehrere Kollisionsphasen auftreten.
- Post-Crash-Phase (Folgephase): Die Folgephase ist der Zeitraum vom Lösen der Unfallbeteiligten bis zu ihrem Stillstand. Bei Mehrfachkollision treten auch mehrere Post-Crash-Phasen auf.

Die Einlaufphase (Pre-Crash-Phase) ist maßgeblich vom Fahrer, der Straßenumgebung und der aktiven Sicherheit vom Fahrzeug abhängig (z.B. Bremsverhalten, Fahrzeugbeladung, gefährliche Kreuzungen, ...usw.).

Die Folgen der Kollisionsphase werden für die betroffenen Verkehrsteilnehmer maßgeblich durch die Maßnahmen der passiven Sicherheit (z.B. Lederkleidung beim Motorradfahrer) beeinflusst. Der Ablauf der Folgephase hängt stark von den verschiedenen Parametern beim Fahrzeug, beim Insassen und bei der Umgebung (z.B. Straßennässe) ab.[H. Appel, 2002]

Beispiel der Ablaufphasen einer Fahrsituation

Die Abbildung 2.2 zeigt ein vereinfachtes Szenario einer kritischen Situation am Beispiel einer Kurve. Diese kritische Situation kann, muss aber nicht zwangsläufig zu einer Kollision führen. Zu einem bestimmten Zeitpunkt erkennt der Fahrer eine kritische Situation. Es ist zuerst unklar, ob es zu einem Unfall kommt. Nach dem Erkennen dieser Situation obliegt dem Fahrer die Entscheidung, welche Maßnahmen zu greifen sind, um eine Unfall-Situation zu vermeiden. Dabei wird der Fahrer auf bereits zurückliegende Erfahrungen zugreifen und eine zur Vermeidung dieser kritischen Situation geeignete Maßnahme ergreifen. Das Fahrzeug reagiert auf die Fahreraktionen, was zu "Fahrer-Fahrzeug-Interaktion" führt, die zu Unfällen führen.[H. Appel, 2002]

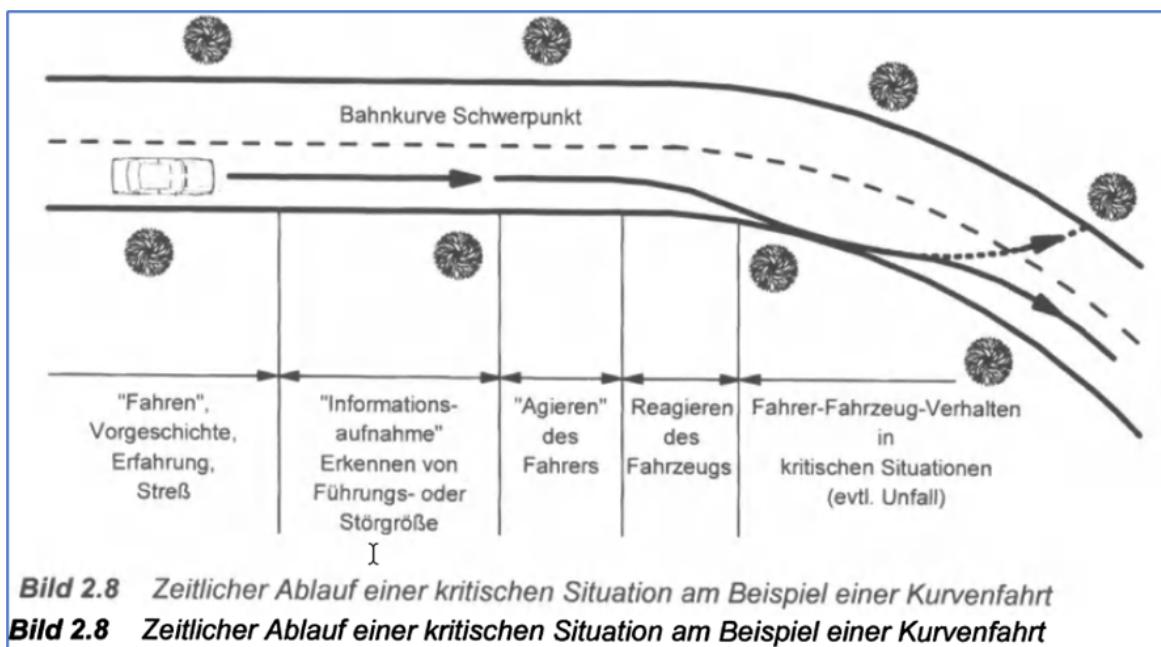


Abbildung 2.2: Beispiel der zeitlichen Phasen einer kritischen Fahrsituation [H. Appel, 2002]

2.1.3 Statistische Zahlen über Motorradunfälle

Das baden-württembergische Verkehrsministerium stellt ein Portal zur Verfügung, über das einzelne Verkehrsmessstellen abgefragt werden können. Die Messstationen

wurden nach zwei Kriterien ausgewählt. Einerseits müssen Unfallschwerpunkte in unmittelbarer Nähe zu Messstationen sein, um zuverlässige Aussagen über Verkehr und Störstellen zu treffen. Andererseits muss darauf geachtet werden, dass es keine Abzweigungen zwischen der Messstation und dem Unfallschwerpunkt gibt, da sonst das richtige Verkehrsaufkommen nicht erfasst werden kann.

Die Abbildung 2.3 stellt dar, wie oft die Geschwindigkeit von Motorradfahrer sowie von PKW-Fahrer an verschiedenen Stationen überschritten wurde. Aus der Grafik ist deutlich zu erkennen, dass an sechs der sieben Stationen das Geschwindigkeitslimit regelmäßig überschritten wird. Die Motorradfahrer missachten die Geschwindigkeitsbegrenzung häufiger als die PKW-Fahrer. [d. Maire, 2020]

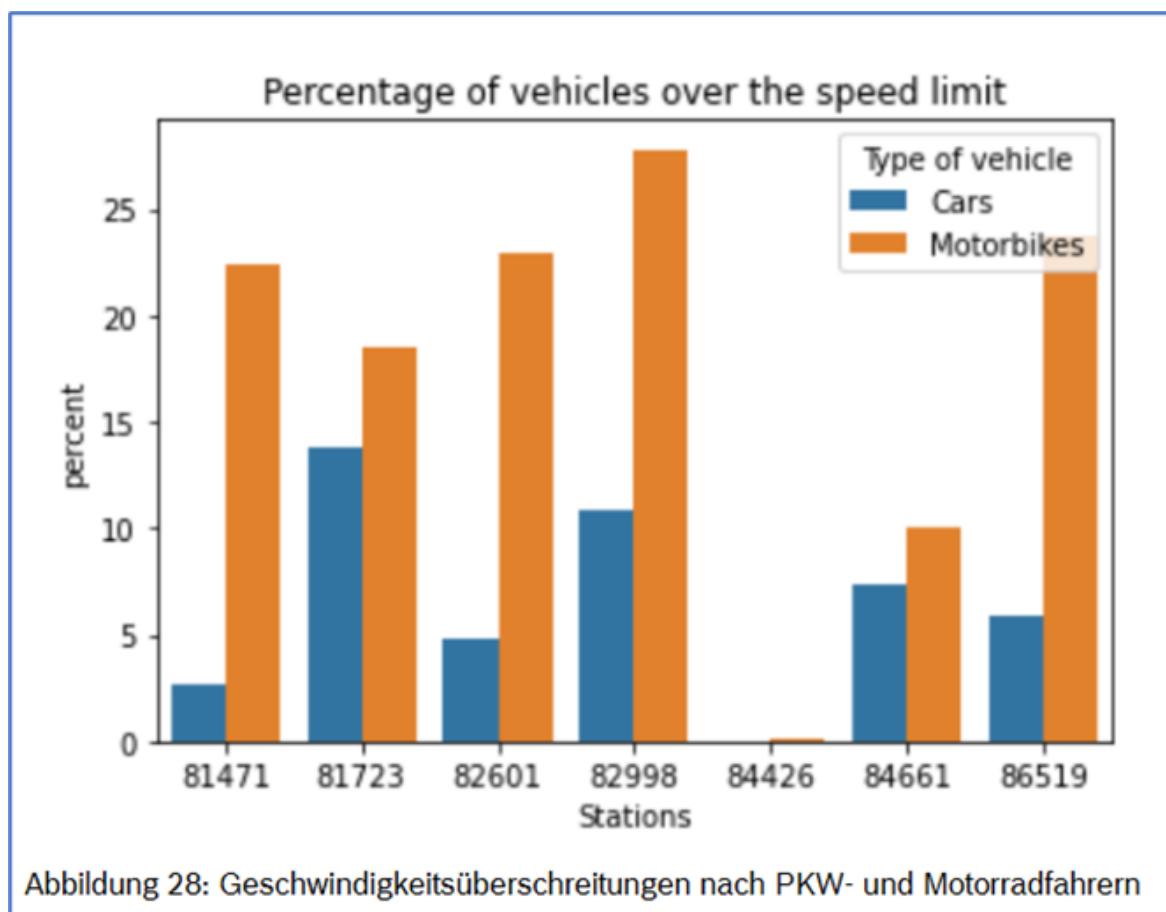


Abbildung 28: Geschwindigkeitsüberschreitungen nach PKW- und Motorradfahrern

Abbildung 2.3: *Geschwindigkeitsüberschreitungen nach PKW- und Motorradfahrer an sieben Messstationen*[d. Maire, 2020]

Die GIDAS-Daten verfügen über die Ausgangsgeschwindigkeit des Motorradfahrers, was die Ermittlung des Einflusses dieser Geschwindigkeit auf die Unfallfolgen ermöglicht. In der Abbildung 2.4 sind die Unfallschwere nach Motorradgeschwindigkeit als Histogramm dargestellt. Die Unfälle werden dabei nach Unfallschwere (leicht verletzt, schwer verletzt, tödlich verwundet) unterteilt. Die gestrichelten Linien repräsentieren die Mittelwerte der Histogramme. Die Grafik zeigt, dass die Verletzungsschwere stark

von der Geschwindigkeit abhängig ist. Bei einer 0km/h gibt es keine tödliche Unfälle, wobei die Unfälle bei einer Geschwindigkeit von 100 fast immer mit einer schweren Verletzung oder tödlichen Verkehrsteilnehmer enden. Die hohe Geschwindigkeit kommt immer mit hohen Kräfte zusammen, welche bei einem Unfall dem Fahrer bewirken. Im Fall eines Motorradfahrers ist das besonders wichtig zu betrachten, da diese Kräfte dem Fahrer direkt übertragen werden. [d. Maire, 2020]

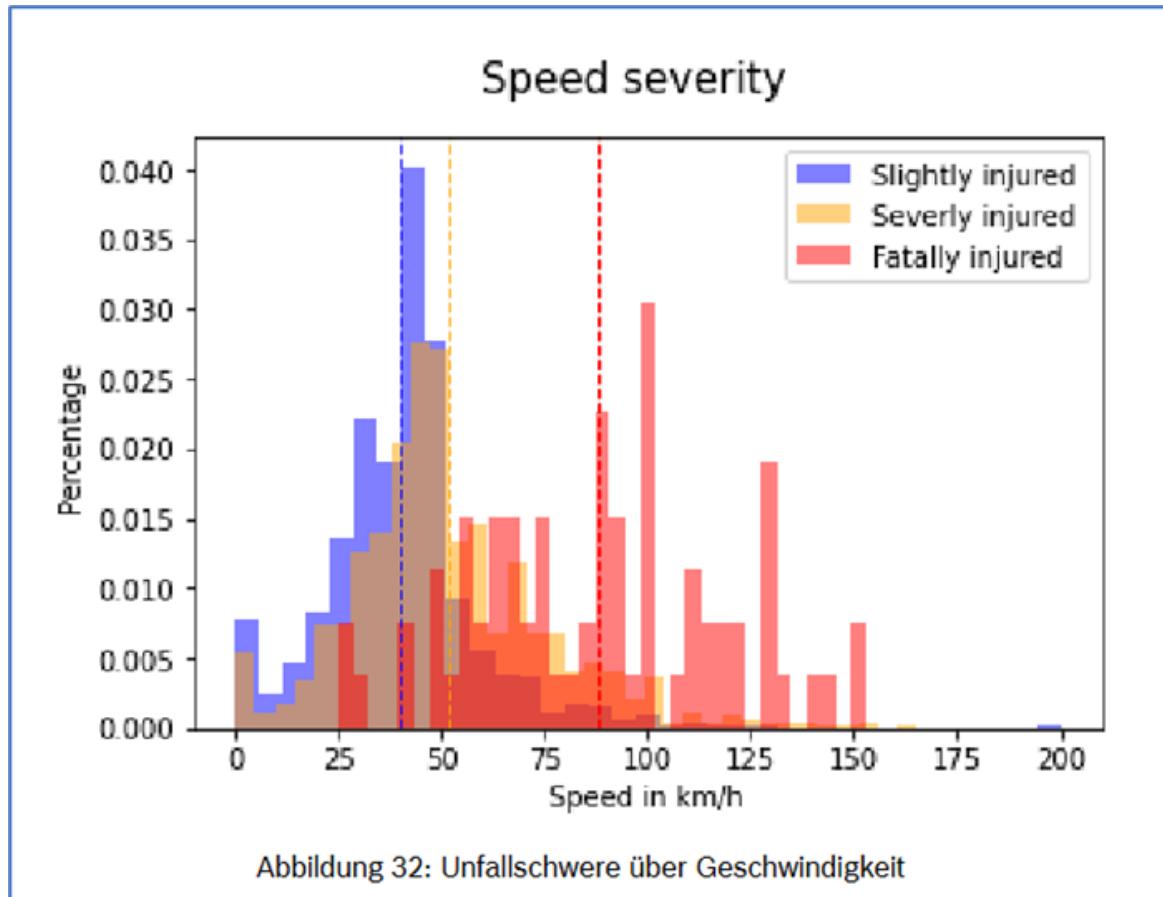


Abbildung 2.4: *Unfallschwere über Geschwindigkeit aus der GAIDA-Datenbank/d. Maire, 2020]*

Statistische Zahlen aus mehreren Youtube-Videos

Im Sinne der Verifizierung des angepassten Unfallerkennungsalgorithmus muss erstmals bekanntgegeben werden, welche Unfälle beziehungsweise Unfallarten am häufigsten vorkommen, damit diese tief betrachtet werden. Dazu wurden mehrere Videos von Motorradunfälle auf dem Plattform "Youtube" stichpunktartig angeschaut und die vorgestellten Unfälle statistisch analysiert. Es wurden insgesamt 32 Unfallsituationen ausgewertet. In der Auswertung wurden die Unfallgegner und der Ablauf des Unfalls betrachtet. [MostrandomComps, 2019] [Moto-Passion, 2018b] [Moto-Passion, 2018c] [Moto-Passion, 2018a] [Moto-Passion, 2018d]

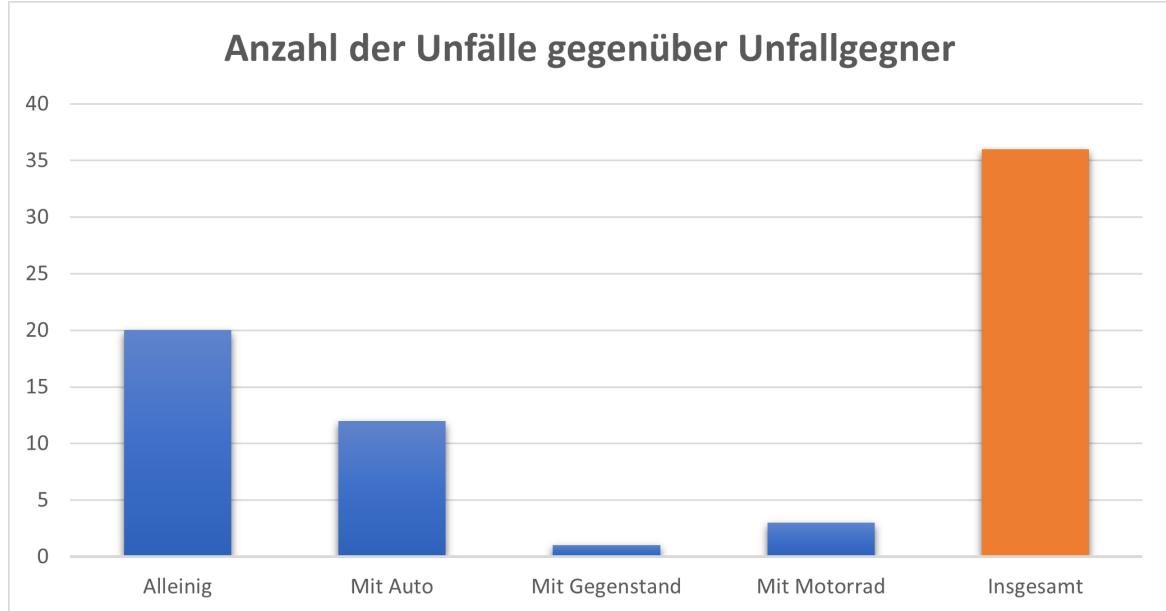


Abbildung 2.5: *Anzahl der Unfälle gegenüber Unfallgegner*

In der Abbildung 2.5 stellt die Grafik die Anzahl der Motorradunfälle gegenüber der Unfallgegner dar. Es wurde hier zwischen alleiniger Unfall, Unfall mit einem Gegner (Auto, Motorrad) oder Unfall wegen eines Gegenstands unterschieden. Diese Unterscheidung ist wegen der Verhaltensunterschied während eines Unfalls notwendig. Von insgesamt 36 Unfälle waren die alleinige Unfälle am meisten gefolgt von den Unfällen mit einem Auto. Die Unfälle mit einem anderen Motorrad oder wegen eines Hindernis sind am wenigsten. Unter alleinige Unfälle werden die zwei Szenarien (An einer Kurve rutschen und Kontrolle verlieren) am meisten aufgetreten, was sehr gut in der Abbildung 2.7 sichtbar sein kann. Das Szenario, in dem das Motorrad an ein Auto von hinten angestoßen hat, hatte die dritte Stelle besetzt. Die Hauptgründe der Unfällen mit einem Gegner waren vor Allem die hohe Geschwindigkeit und das schlechte Wetter (z.B. Nässe, Schnee), was zu Schwierigkeiten geführt hat, das Motorrad in kritischen Situationen zu kontrollieren.

Die Abbildung 2.6 zeigt den Anteil der aktiven sowie passiven Unfälle. Wenn das Motorrad angestoßen wird, wird von einem passiven Unfall gesprochen, da der Motorradfahrer keinen Einfluss darauf hat. Im Vergleich dazu könnte er bei einem aktiven Unfall das Ergebnis beeinflussen, in dem er langsamer fährt oder mehr Abstand mit den anderen Verkehrsteilnehmer hält.

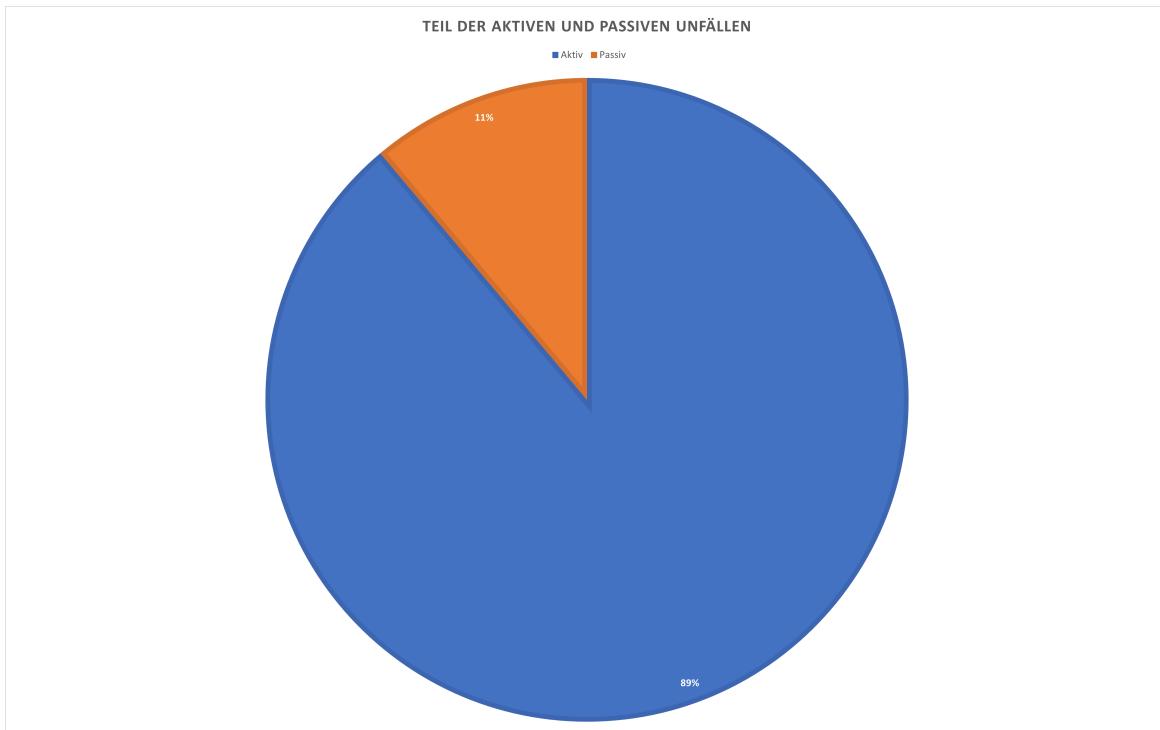


Abbildung 2.6: Anteil der aktiven sowie passiven Unfälle

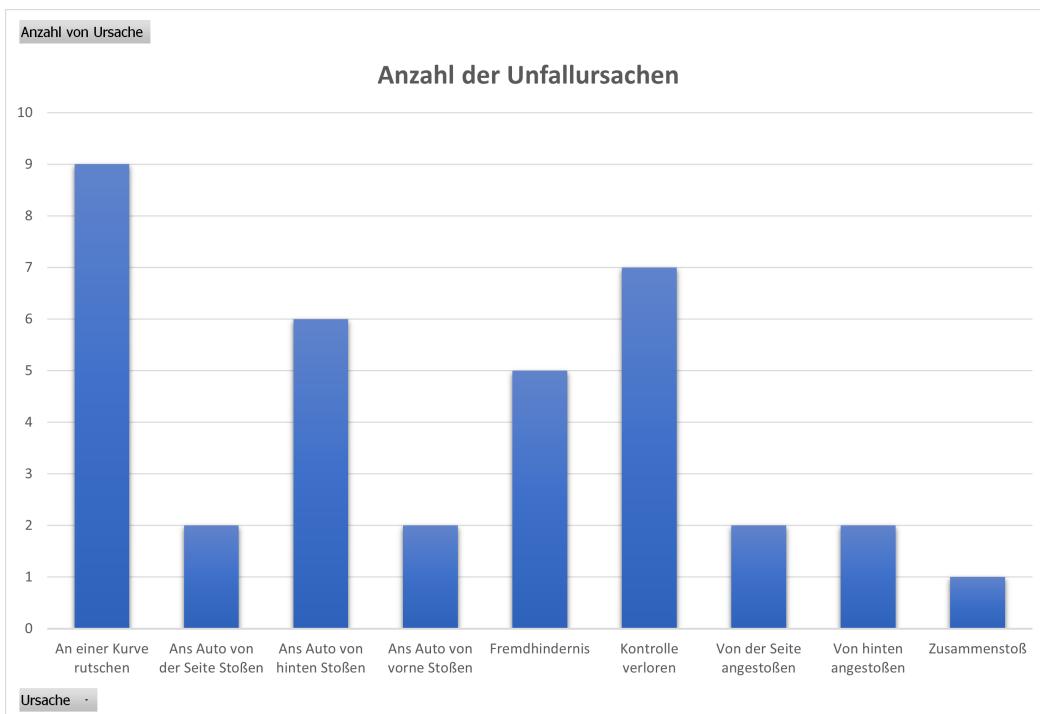


Abbildung 2.7: Anzahl der Unfälle über der Art der Ursache

Bilder aus den YT-Videos hinzufügen als Beispiele der Unfälle (vor Allem Alleiniger

Unfall)

2.1.4 Mechanik der Motorradfahrt

- Kinematik und Verletzungsbilder
- Zweiradfahrer (Bilder)

Kurvenfahrt

Der Fahrer bestrebt während der Fahrt immer das Gleichgewicht zu halten, damit er und das Motorrad nicht umkippt. Bei einer Geradeausfahrt und ab einer Geschwindigkeit von 25 km/h stabilisieren die Kreiselkräfte der Räder die Maschine. Bei einer Geschwindigkeit unter 25 km/h reichen die Kreiselkräfte nicht mehr aus, um das Gleichgewicht zu gewährleisten und der Fahrer muss mit kleinen Lenkausschlägen von bis zwei Grad nach links und rechts die Gleichgewichtslage halten.

Befährt ein Motorradfahrer eine Kurve, tritt im Vergleich zu einer Geradeausfahrt eine Instabilität im Gleichgewicht auf und versucht der Fahrer die Maschine wieder zum Gleichgewicht zu bringen. Während der gesamten Kurvenfahrt kann der Fahrer den Verlauf der Fahrlinie sowohl durch positives oder negatives Beschleunigen als auch durch die Veränderung des Lenkwinkels beeinflussen. [Hädrich, 2012]

Wirkende Kräfte am Fahrzeugschwerpunkt

Beim Einleiten einer Kurvenfahrt mit konstantem Bahnradius wirkt eine konstante Querbeschleunigung (a_q) auf die Einheit "Fahrer-Maschine". Diese Querbeschleunigung bewirkt eine Seitenkraft im Gesamtschwerpunkt der Einheit "Fahrer-Maschine". Die Abbildung 2.8 zeigt die wirkenden Kräfte und die daraus resultierenden Momente während der stationären Kurvenfahrt an einem vereinfachten Modell. Die Gewichtskraft $F_G = m_g \cdot g$ sowie die Fliehkraft beziehungsweise Zentrifugalkraft $F_a = m_g \cdot a_q$ sind in der Abbildung ersichtlich. Die Fliehkraft versucht das Motorrad nach Außen zum Kippen zu bringen, deswegen wird das Fahrzeug die Kurve unter Schräglage durchzufahren. Dadurch verschiebt sich der Schwerpunkt der Einheit zur Kurvenseite und wird ein Moment der Gewichtskraft um den Reifenaufstandspunkt resultiert, welches dem Moment der Fliehkraft entgegen wirkt und wieder ein Gleichgewicht sichert.

Der Schrägwinkel (λ) der Einheit lässt sich wie folgt berechnet:

$$\tan(\lambda) = \frac{F_a}{F_G}$$

$$\lambda = \arctan \left(\frac{F_a}{F_G} \right) \quad (2.1)$$

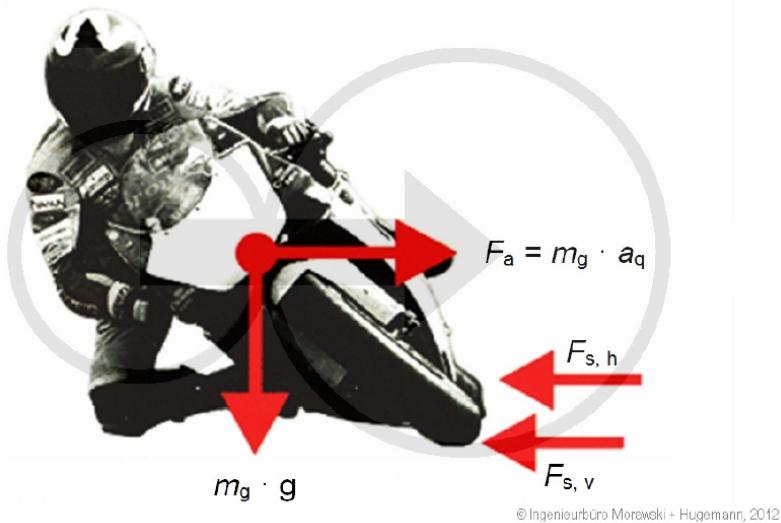


Abb. 3-2: Vereinfachte Darstellung der Kräfte bei stationärer Kurvenfahrt [ECK10a]

Abbildung 2.8: Vereinfachte Darstellung der Kräfte bei stationärer Kurvenfahrt)

Kurventechniken:

Die bisherigen Überlegungen und Berechnungen beziehen sich jeweils auf den Fall, dass der Fahrer während der Kurvenfahrt in einer Flucht mit der Maschinenachse bleibt. Tatsächlich gibt es jedoch verschiedene Techniken, eine Kurve durchzufahren, vgl. (Abbildung 2.9).

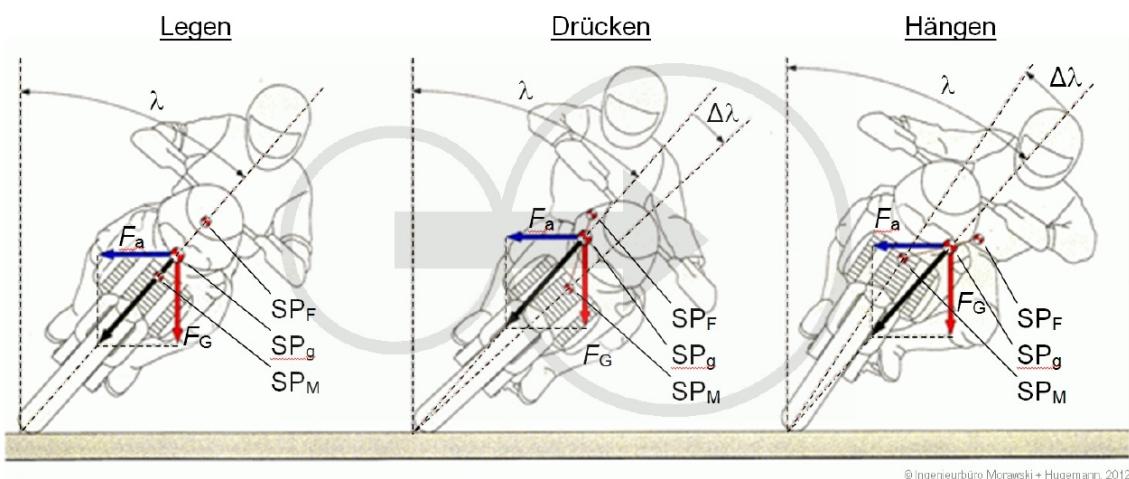


Abb. 3-6: Verschiedene Kurventechniken bei gleicher Geschwindigkeit [COC05]

Abbildung 2.9: Verschiedene Techniken bei Kurvenfahrt mit gleicher Geschwindigkeit

Der Fahrstil „Drücken“ hat im Vergleich zum Fahrstil „Legen“ keinen Vorteil im Zu-

sammenhang des Grips auf der Straße. Es ist durchaus vorstellbar, dass mit zunehmender Schräglage die Reifenaufstandsfläche (beziehungsweise Reifenlatsch) abnimmt. Fahrer von Renn- und Supersport-Maschinen bedienen sich der Tatsache des sich verändernden Reifenlatsches und „Hängen“ sich während der Kurvenfahrt von der Maschine. In diesem Fall liegt der Schwerpunkt des Fahrers SPF unterhalb des Schwerpunktes der Maschine SPM, so dass das Motorrad die Kurve mit deutlich geringerer Schräglage durchfahren kann, als beim „Legen“, (Abbildung 2.9). Gegenüber der anderen zwei Schräglagen kann durch das „Hängen“ mit gleichen Schräginkel höhere Seitenkräfte übertragen werden, da hier die Radaufstandsfläche größer ist.

2.2 Technische Grundlagen

2.2.1 Sensoren und Signale der Smartphones

Die meisten Smartphones haben einen Beschleunigungsmesser, und viele enthalten jetzt ein Gyroskop. Je nach Gerät können die softwarebasierten Sensoren ihre Daten entweder vom Beschleunigungs- und Magnetometer oder vom Gyroskop beziehen. Diese Sensoren sind nützlich zum Überwachen von Gerätebewegungen wie Neigung, Schütteln, Drehung oder Schwingen. Die Gerätsbewegung spiegelt normalerweise die direkte Benutzereingabe wider, kann aber auch die physische Umgebung widerspiegeln, in der sich das Gerät befindet (z.B. Das Smartphone bewegt sich mit der Person, die es am Körper hat und sich selbst bewegt).[Developers, 2022]

Beschleunigungssensoren

- Beispielsignal

Der Beschleunigungssensor ist ein elektromechanisches Gerät, das die Beschleunigungskraft misst, die durch Bewegung, Schwerkraft oder Vibration verursacht wird. Mathematisch gesehen ist die Beschleunigung ein Maß für die zeitliche Geschwindigkeitsänderung. Der Beschleunigungssensor im Smartphone misst die lineare Beschleunigung des Geräts. In der Ruheposition stellt die Figur die auf das Gerät wirkende Schwerkraft dar und misst gleichzeitig auch die Beschleunigung auf der X- und Y-Achse, die Null sein wird. Die meisten Smartphones verwenden heutzutage Beschleunigungssensoren, um die Bildschirmanzeige abhängig von der Position auszurichten, in der das Gerät gehalten wird. Mit den eingebauten Beschleunigungssensoren können Benutzer unter Anderem ein besseres Anzeigegerlebnis erzielen. [Sharma, 2020]

Der Beschleunigungssensor im mobilen Gerät liefert die XYZ-Koordinatenwerte, die zum Messen der Position und der Beschleunigung des Geräts verwendet werden. Die XYZ-Koordinate stellt die Richtung und Position des Geräts dar, an dem eine Beschleunigung aufgetreten ist. Die Drehrichtung und -position werden mit Gyroskopsensoren gemessen. Die vom Gerät bereitgestellten Beschleunigungsmesserwerte enthalten normalerweise auch die Schwerkraft. Das Signal des Beschleunigungssensor wird

in die Tief-/Hochpassfilter geleitet, um das Ergebnis basierend auf der verwendeten Anwendung zu verfeinern. [Sathish, 2021]

- Wird das Gerät auf die linke Seite geschoben (bewegt sich also nach rechts), ist der x-Beschleunigungswert positiv.
- Wenn das Gerät auf seinen Boden gedrückt wird, ist der y-Beschleunigungswert positiv.
- Wenn das Gerät mit einer Beschleunigung von $A \text{ m/s}^2$ in den Himmel geschoben wurde, ist der Wert der z-Beschleunigung gleich $A + 9,81$, da die Schwerkraft ($9,81 \text{ m/s}^2$) mitberechnet wird.

Im Allgemeinen ist der Beschleunigungssensor ein guter Sensor, wenn die Bewegung des Geräts überwacht werden soll. [Developers, 2022]

Gyroskop

- Beispieldaten

Gyroskop ist ein Gerät, das ein sich schnell drehendes Rad oder einen umlaufenden Lichtstrahl enthält. Gyroskop wird verwendet, um die Abweichung eines Objekts von seiner gewünschten Ausrichtung zu erkennen. Gyroskope werden zur automatischen Lenkung und zur Korrektur der Dreh- und Nickbewegung in Marschflugkörpern und ballistischen Flugkörpern verwendet.[Rogers, 2020] Der Gyroskopsensor im MEMS ist winzig (zwischen 1 und 100 Mikrometer). Wenn der Gyro gedreht wird, wird eine kleine Resonanzmasse bei einer Winkelgeschwindigkeitsänderung verschoben. Diese Bewegung wird in elektrische Signale mit sehr geringem Strom umgewandelt, die verstärkt und von einem Host-Mikrocontroller gelesen werden können.[sparkfun, 2022]

Global Positioning System (GPS)

GPS besteht aus drei Teilen: Satelliten, Bodenstationen und Empfängern. Die Position der Satelliten ist jederzeit bekannt. Die Bodenstationen verwenden Radar, um sicherzustellen, dass die Satelliten sich tatsächlich dort befinden, wo sie sich befinden sollen. Ein Empfänger in dem Smartphone oder im Auto wartet ständig auf ein Signal von diesen Satelliten und findet heraus, wie weit er von einigen von diesen Satelliten entfernt ist. Sobald die Entfernung zwischen einem Empfänger und vier oder mehr Satelliten berechnet wurde, ist genau bekannt, wo der Empfänger sich befindet. Der Basis-GPS-Dienst bietet Benutzern eine Genauigkeit von etwa 7,0 Metern, 95% der Zeit. GPS-Empfänger zeigen die Geschwindigkeit an und berechnen die Geschwindigkeit mithilfe von Algorithmen im Kalman-Filter.[Nasa, 2019][FAA, 2021][Deiss, 1999]

2.2.2 Integrierte Aktivitätserkennung

- Es gibt vom Google eine integrierte Aktivitätserkennung.
- Beispielsignal
- Das Ergebnis ist die Wahrscheinlichkeit pro Aktivität.
- Eine Darstellung hinzufügen. In der Abb. sind zwei Grafiken: 1. Acc-Signal bei verschiedenen Aktivitäten. 2. Eine Aktivitätsklassifizierung mit einer Phasenunterscheidung!

2.2.3 Matlab/Simulink

Matlab ist eine Hochleistungssprache für technisches Rechnen. Matlab integriert Berechnung, Visualisierung und Programmierung in einer benutzerfreundlichen Umgebung, in der Probleme und Lösungen in einer vertrauten mathematischen Notation ausgedrückt werden.

Simulink ist ein grafisches Softwarepaket zur Modellierung, Simulation und Analyse dynamischer Systeme und basiert auf Matlab. Die Software hat sich in den letzten Jahren zum weitesten verbreiteten Softwarepaket in Wissenschaft und Industrie entwickelt. Simulink unterstützt lineare und nichtlineare Systeme, die in kontinuierlicher Zeit, gesampelter Zeit oder einer Mischung aus beiden modelliert sind. Für die Modellierung bietet Simulink eine grafische Benutzeroberfläche (GUI) zum Erstellen von Modellen als Blockdiagramme. Mit dieser Schnittstelle können die gewünschten dynamischen Systeme einfach aufgebaut werden. Mithilfe von Scopes und anderen Anzeigeblocken können die Simulationsergebnisse während der Simulation analysiert werden. Die Simulationsergebnisse können zur Nachbearbeitung und Visualisierung in den MATLAB-Arbeitsbereich gestellt werden. [Blaabjerg, 2004][Karris, 2008]

Simulink und LabVIEW

LabVIEW ist eine von "National Instruments" entwickelte Software. Sie wird häufig von Ingenieuren, Wissenschaftlern und Studenten für die Datenerfassung, Instrumentensteuerung und industrielle Automatisierung verwendet. Die LabVIEW-Umgebung besteht aus zwei Hauptkomponenten: Frontpanel (FP) und Blockdiagramm (BD). Ein FP stellt die grafische Benutzeroberfläche bereit, während ein BD die Bausteine eines Systems enthält und einem Flussdiagramm ähnelt. LabVIEW-Systeme werden als virtuelle Instrumente (VIs) bezeichnet und ihr FP erscheint als Instrumententafel, die aus verschiedenen Bedienelementen und Anzeigen besteht.

Ähnlich wie LabVIEW bietet Simulink einen blockbasierten Programmieransatz für die Simulation, den Entwurf und die Analyse dynamischer Systeme. Es bietet eine interaktive grafische Umgebung zusammen mit einer Reihe von Bibliotheken zum Ent-

werfen und Simulieren von Systemen, einschließlich DSP-Systemen. Simulink-Blöcke werden als Modelle bezeichnet, und im Gegensatz zu LabVIEW werden die Codeimplementierung und Eingabe-/Ausgabeeinheiten in Simulink nicht explizit unterschieden. Simulink ist in MATLAB integriert und kann daher auf die Funktionen und Tools zugreifen, die in der MATLAB-Umgebung verfügbar sind. [N. Kehtarnavaz, 2006] [Kasnakoglu, 2015]

Wenn komplexe Simulationen ausgeführt werden sollen oder komplexe Simulationsmodelle von Steuerungen oder Anlagen zu erstellen/debuggen sind, wird Simulink verwendet, da LabVIEW keine effizienten Codegeneratoren für die dynamische Simulation hat. Simulink konzentriert sich hauptsächlich auf Simulation und Modellierung, was bei LabVIEW sicherlich nicht der Fall ist.

Der implementierte Algorithmus enthält einen verbreiteten und komplexen Entscheidungsbaum, welcher sich mit Simulink sowohl übersichtlicher als auch einfacher darstellen lässt als mit LabVIEW.

App-Entwicklung

Die Entwicklung mobiler Apps ist der Prozess zur Erstellung von Software für Smartphones und digitale Assistenten. Die Software kann auf dem Gerät vorinstalliert oder aus einem mobilen App Store heruntergeladen werden. Eine der bekannten Sprachen in der App-Entwicklung ist C. C ist eine leistungsstarke Programmiersprache, mit der Anwendungen in mehreren Bereichen erstellt werden können, von einfachen Taschenrechnern und Apps bis hin zu Videospielen. Sie ist eine Sprache auf niedriger Ebene, dies bietet Geschwindigkeit und eine weitaus bessere Möglichkeit zur Speicherverwaltung.

Für die Generierung eines C-Codes aus Simulink-Modellen wird der in Matlab/Simulink integrierte C/C++-Coder verwendet. *****Quelle*****

C/C++-Coder

Der C/C++-Code-Generator wird für Rapid Prototyping, Hardware-in-the-Loop-Tests, Simulationsbeschleunigung oder einfach als ausführbare Datei zur Ausführung außerhalb von MATLAB und Simulink verwendet. Diese Codegenerierung ist der Prozess der Generierung von Low-Level-Code direkt aus einer High-Level-Programmiersprache oder Modellierungsumgebung.

Der C/C++-Coder ist ein weiterer Vorteil von Simulink gegenüber LabVIEW und hierfür ist Simulink die richtige Entscheidung für die Implementierung der Software beziehungsweise des Unfallerkennungsalgorithmus'.

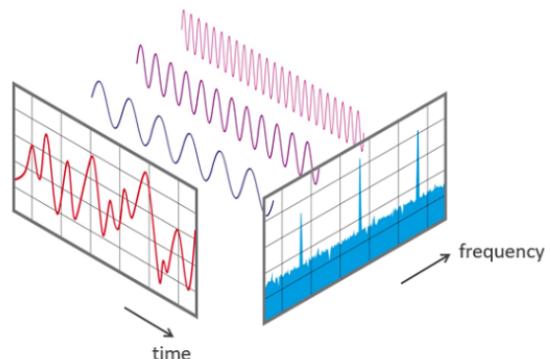
2.3 Mathematische Grundlagen

In dieser Arbeit wird eine FFT eingesetzt, für welche das folgende Hintergrundwissen zum Verständnis benötigt wird.

2.3.1 Fast Fourier Transform (FFT)

Die „Fast Fourier Transform“ ist ein wichtiges Messverfahren und wurde erstmals von Cooley und Tukey 1965 diskutiert, obwohl Gauß den kritischen Faktorisierungsschritt bereits 1805 beschrieben hatte. Dieses Verfahren wandelt ein Signal in einzelne Spektralkomponenten um und liefert dadurch Frequenzinformationen über das Signal. FFT wird zur Fehleranalyse, Qualitätskontrolle und Zustandsüberwachung von Maschinen oder Anlagen eingesetzt. Dieser Abschnitt erläutert die Funktionsweise einer FFT, die relevanten Parameter und deren Auswirkungen auf das Messergebnis. Die FFT ist ein optimierter Algorithmus zur Umsetzung der „Diskrete Fourier Transformation“ (DFT). Ein Signal wird über einen Zeitraum abgetastet und in seine Frequenzkomponenten zerlegt. Diese Komponenten sind einzelne sinusförmige Schwingungen mit unterschiedlichen Frequenzen, jede mit ihrer eigenen Amplitude und Phase. Diese Transformation ist an einem Beispiel in der Abbildung 2.10 dargestellt.

Das Diagramm zeigt ein kompliziertes Signal im Zeitbereich, das aus der Summe der drei periodischen Grundsignalen mit unterschiedlichen Frequenzen entsteht. Im Frequenzbereich sind u.A. die einzelnen Frequenzen der Grundsignale dargestellt. Dadurch lässt sich eine FFT-Transformation zwischen dem Zeit- sowie Frequenzbereich grafisch darstellbar.



View of a signal in the time and frequency domain

Abbildung 2.10: *Beispiel von einer FFT (Zeitbereich und Frequenzbereich)*[NTI-Audio, 2019]

Schritt für Schritt

Im ersten Schritt wird ein Ausschnitt des Signals abgetastet und zur weiteren Verarbeitung im Speicher abgelegt. Zwei Parameter sind hier relevant:

1. Die Abtastrate beziehungsweise Abtastfrequenz f_s des Messsystems (z.B. 48 kHz). Dies ist die durchschnittliche Anzahl von Abtastungen, die in einer Sekunde erhalten werden (Abtastungen pro Sekunde)
2. Die Blocklänge B_L ist die ausgewählte Anzahl von Proben (Samples). Dies ist immer eine ganzzahlige Potenz zur Basis 2 (z.B. $2^{10} = 1024$ Samples)

Aus den beiden Grundparametern f_s und B_L können weitere Parameter der Messung bestimmt werden.

Bandbreite: f_n (= Nyquist-Frequenz). Dieser Wert gibt die theoretische maximale Frequenz an, die durch die FFT bestimmt werden kann.

$$f_n = \frac{f_s}{2} \quad (2.2)$$

Beispielsweise können bei einer Abtastrate von 100 Hz Frequenzanteile bis 50 Hz bestimmt werden.

Messdauer: D ergibt sich aus der Abtastrate f_s und der Blocklänge B_L wie folgt:

$$D = \frac{B_L}{f_s} \quad (2.3)$$

Frequenzauflösung: d_f gibt den Frequenzabstand zwischen zwei Messergebnissen an.

$$d_f = \frac{f_s}{B_L} = \frac{1}{D} \quad (2.4)$$

In der Praxis ist die Abtastfrequenz f_s meist eine vom System vorgegebene Größe. Durch die Auswahl der Blocklänge B_L kann jedoch die Messdauer D und Frequenzauflösung d_f definiert werden. Es gilt:

- Eine kleine Blocklänge B_L führt zu schnellen Messwiederholungen mit grober Frequenzauflösung.
- Eine große Blocklänge B_L führt zu langsameren Messwiederholungen mit feiner Frequenzauflösung.

Spiegelfrequenzen

Wird die Nyquist-Frequenz (Gleichung 2.2) überschritten, wird das Signal an dieser gedachten Grenze reflektiert und fällt wieder in das Nutzfrequenzband zurück. Diese unerwünschten Spiegelfrequenzen wird vor der Abtastung mit einem analogen Tiefpassfilter (Anti-Aliasing-Filter) entgegengewirkt. Der Filter sorgt dafür, dass Frequenzen oberhalb der Nyquist-Frequenz unterdrückt werden. [NTI-Audio, 2019][Weisstein, 2022]

2.4 Agile Softwareentwicklung

Das Ziel der agilen Softwareentwicklung ist die kontinuierliche Bereitstellung funktionsfähiger Software, die in schnellen Iterationen erstellt wird. Die agile Softwareentwicklung ermöglicht eine kontinuierliche Bereitstellung funktionsfähiger Software,

die in schnellen Iterationen erstellt wird. Bei einer agilen Softwareentwicklung werden die Entwicklungsphasen in mehreren Sprinten geteilt. Die Länge einer Sprint wird am Anfang festgestellt. Nach jedem Sprint wird das Ergebnis ausgewertet. Dieses wird ggf. für die Anpassung des Entwicklungsvorgehens im nachfolgenden Sprinten benutzt [Brunskill, 2019]. Die Entwicklung des Unfallerkennungsalgorithmus' sowie das Pocket-Mode wird mittels der agilen Methode erfolgt.

2.5 Unfallerkennungsalgorithmus

In diesem Teil wird der Unfallerkennungsalgorithmus erläutert und näher betrachtet. Der Algorithmus ist sowohl für Fahrräder als auch für Motorräder entwickelt und bearbeitet die Signale der Beschleunigungssensor sowie Gyroskope im Smartphone und die über Gps gemessene Geschwindigkeit. In dem Unfallerkennungsalgorithmus werden drei Hauptkriterien (CollisionHit, GroundHit und TipOver) berücksichtigt. Die Modellierung der Merkmale GroundHit und Collision erfordert weitere unabhängige vorverarbeitete Signale. Zu diesem Zweck wird der ANOVA-Ansatz (Analysis of Variance) verwendet, um verschiedene statistische Eigenschaften (z. B. Mittelwert, Standardabweichung, Varianz, Extrema und Integral...usw.) über verschiedene Fenstergrößen des zu analysieren IMU-Beschleunigungsdaten. Der ANOVA-Ansatz erfordert, dass die vorverarbeiteten Daten normalverteilt sind. Der Anderson-Darling-Test wird auch angewandt, um diese notwendige Bedingung zu überprüfen. Das Ergebnis der ANOVA-Analyse liefert die spezifische Energie als optimalen Indikator unter denen aus den untersuchten statistischen Ansätzen zur Klassifizierung der Kollisions- und Bodentreffer-Ereignisse wie folgt:

$$\Delta e_{xy,u}(i) = \left(\int_{i-u}^i a_{Bf,x}(k) dk \right)^2 + \left(\int_{i-u}^i a_{Bf,y}(k) dk \right)^2 \quad (2.5)$$

$\Delta e_{xy,u}$ stellt die Änderung der massenspezifischen kinetischen Energie dar. Es beschreibt ein Ereignis in der XY-Ebene im Fahrradkoordinatensystem (also dem Fahrradrahmen (BF)) während des Zeitfensters u durch die Integration der Beschleunigungssignale $a_{Bf,x}$ und $a_{Bf,y}$. Da Kollisions- und Bodentrefferereignisse nur in dieser Ebene stattfinden, werden die Auswirkungen in der vertikalen z-Achse auf die Fahrbahnoberfläche oder Sprünge des Fahrradsystems zurückgeführt. Zusätzlich bietet die Varianz der x- und y-Beschleunigungssignale über ein Zeitfenster u_{GH} weitere Trennmöglichkeiten für GroundHit-Ereignisse. [Schnee u. a., 2021]

Die Signale aus dem Smartphone weisen keine Fahrtrichtung zu und müssen je nach Position unterschiedlich bearbeitet werden. Bevor die Signale zur Entscheidung verarbeitet werden, ist eine Kalibrierung notwendig.

2.5.1 Kalibrierung

Die Kalibrierung dient dazu, die Ausrichtung des Motorrads zu erkennen, damit die Richtung der Fahrt sowie diesbezügliche Bewegungen (Beschleunigung, Bremsen, Nei-

gung, ...usw.) richtig erkannt und gut ausgewertet werden.

In der Abbildung 2.11 ist der Unterschied zwischen den originalen Achsenrichtung (vom Smartphone) und diesen des Objekts anhand eines Fahrradbeispiel abgebildet. In der Abbildung sind die Achsen des Smartphones mit den Ziffern 'SF' (Sensor frame) vermerkt, sowie diese des Fahrrads mit den Ziffern 'BF' (Bike Frame).

Während der Kalibrierung wird der Stand des Fahrrads beziehungsweise Motorrads sowie die Richtung der Fahrt erkannt und die gesammelten Daten aus dem Smartphone so umgerechnet, dass sie fürs Koordinatensystem des Fahrrads (BF) geeignet sind.

Nachdem der Benutzer die App zum ersten Mal installiert, kalibriert sich der Algorithmus während der ersten Fahrt. Sollte der Benutzer die Lage des Smartphones nachkorrigieren, kalibriert sich der Algorithmus langsam nach. Die Nachkalibrierung erfolgt langsamer als die Erstkalibrierung, damit die Unfälle durch eine schnelle Nachkalibrierung nicht übersehen werden.

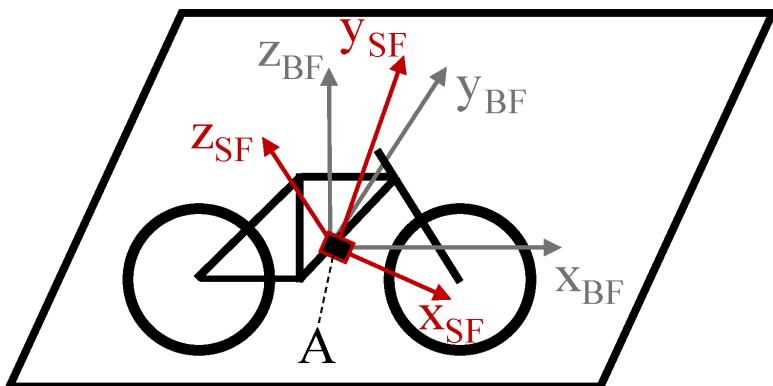


Abbildung 2.11: Achsenrichtung in Sensorframe sowie in Bikeframe [Schnee u. a., 2020]

2.5.2 Übersicht der bereits erkennbaren Unfälle

In der Abbildung 2.12 ist der Entscheidungsbaum abgebildet, wonach eine Unfallklassifizierung erfolgt wird. Die Einzelkomponenten (CollisionHit, GroundHit, TipOver) werden später separat erläutert.

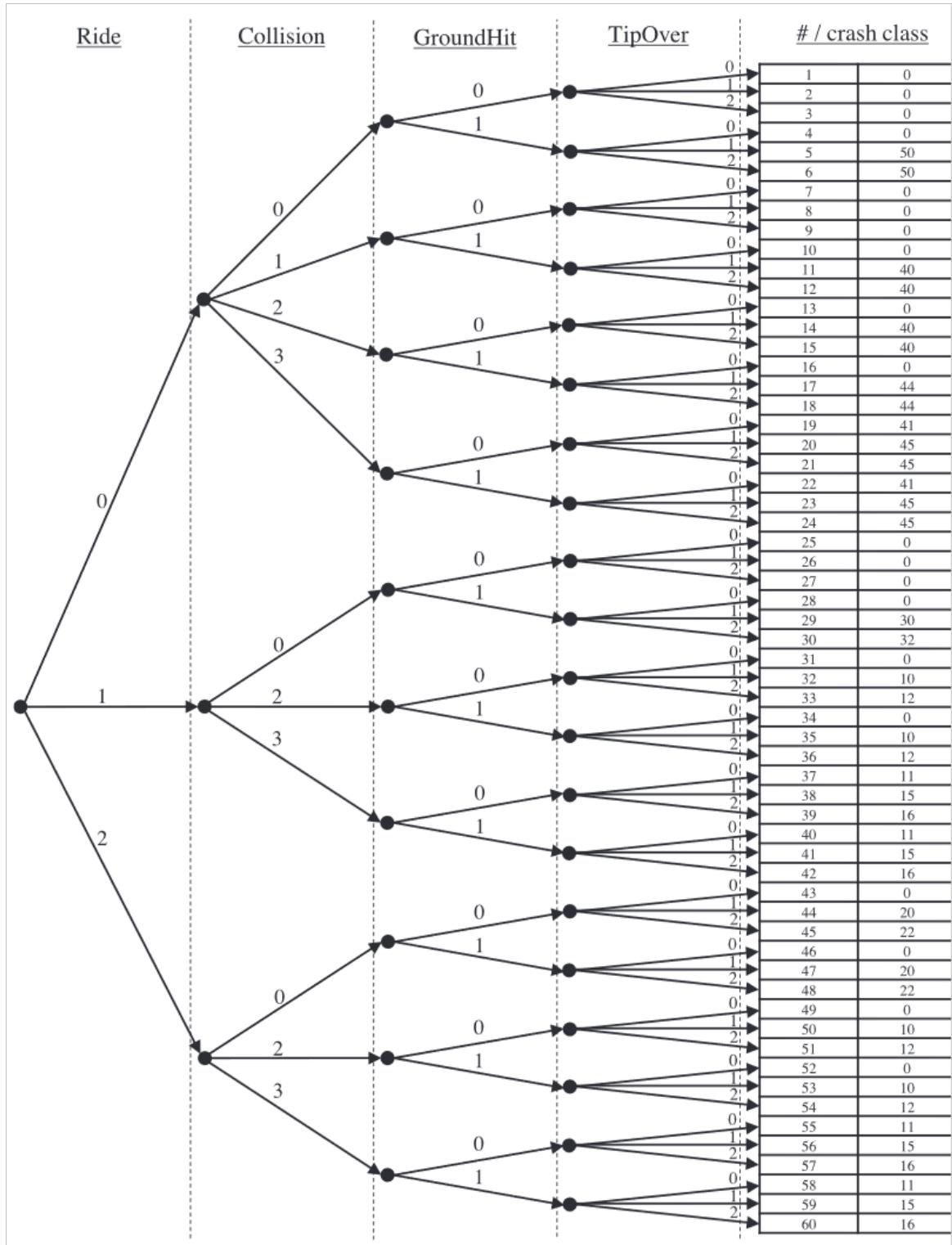


Abbildung 2.12: Entscheidungsbaum des Unfallerkennungsalgorithmus' [Schnee u. a., 2021]

Table 1
Definition of bicycle crash features and polytomous states.

#	Name	Description	States
1	Ride (Rd)	riding state, right before accident	0: standing 1: starting/stopping 2: riding
2	Collision (C)	collision with a rigid obstacle or further party involved	0: no collision 1: soft collision 2: medium collision 3: strong collision
3	GroundHit (GH)	impact when a bicycle is tipping over	0: false, 1: true
4	TipOver (TO)	orientation of bicycle, bicycle rolled over bicycle rotates over the handlebar	0: upright bicycle 1: TipOver (RollOver) 2: NoseOver

Abbildung 2.13: Definition der IDs aus der Abbildung 2.12 [Schnee u. a., 2021]

2.5.3 1. Komponente: TipOver

Diese Komponente kontrolliert den Neigungswinkel des Motorrads und stellt fest, wenn das Motorrad umkippt.

Bei Motorradunfällen auf Straßen, die sich im Anschluss an eine Kurve ereignen, stellt sich immer wieder die Frage nach der maximalen Geschwindigkeit, mit der die Kurve auf einem Motorrad durchfahren werden konnte. Um das Motorrad durch die Kurve zu bewegen, muss sich der Fahrer mit seiner Maschine in die Kurve legen. Während einer Fahrt in der Kurve wirken zwei Kräfte (F_q) und (F_G) am Motorrad (Abbildung 2.8). Wenn der Kraft (F_q) größer als (F_G) tritt, kippt das Motorrad um. Im normalen Fall soll

$$F_G \geq F_q$$

immer gültig sein.

D.h.

$$\frac{F_q}{F_G} \leq 1$$

Beim Einsetzen in der Gleichung 2.1 ergibt sich der maximale zulässige Winkelwert:

$$\lambda_{zul} \leq \arctan(1) \leq 45^\circ$$

D.h. Bei einer Neigung von über 45° , sollte das Motorrad umkippen. Sollte der Fahrer das Fahrstil "Hängen" verwenden, könnte er eine größere Neigung erfolgen, ohne zu rutschen.

Das Modell "TipOver" erkennt den Fall, in dem der Winkel über den Schwellwert liegt, und gibt dem Entscheidungsmodell eine Meldung weiter.

2.5.4 2. Komponente: GroundHit

Dieses Modell dient dazu, den Schlag zu erkennen, wenn das Motorrad am Boden ankommt. Dieses Modell geht von der Energie aus der Gleichung 2.5 aus. Es hat einige Spezifikationen, die die verschiedene Szenarien abdeckt. Zwei typische Szenarien sind in der Abbildung 2.14 dargestellt. Die Abbildung 2.14a zeigt der Fall, wenn ein Fahrrad nach einer ursprünglichen aufrechten Position umkippt, und die Abbildung 2.14b das Umkippen eines Fahrrads mit einer ursprünglichen Neigung (z.B. in einer Kurve). Die Energie vom GroundHit im zweiten Fall ist deutlich kleiner, deswegen wird der Schwellwert nach dem Neigungswinkel angepasst. Je größer die Motorradneigung ist, desto kleiner ist der GroundHit-Schwellwert.

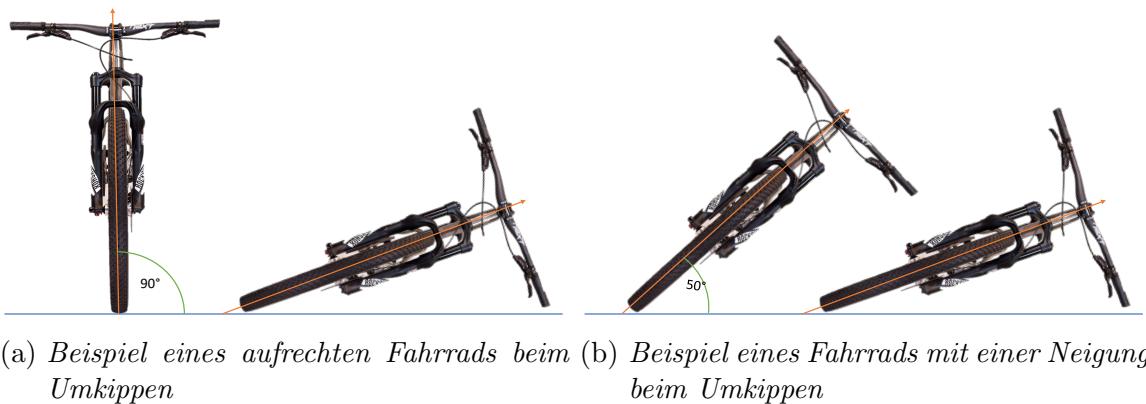


Abbildung 2.14: Beispiel eines Fahrrads beim Umkippen mit einer ursprünglichen aufrechten Position sowie Neigung

Nachdem das Modell den Bodenschlag erkennt, wird eine Meldung dem Entscheidungsmodell weitergegeben. Demnächst werden zwei Beispiele zum besseren Verständnis erläutert.

Beispiele 1: Kein GroundHit

In diesem Beispiel ist eine Testfahrt ohne GroundHit durchgeführt und schließlich analysiert. Nach der Fahrt und bei späterer Simulation wurde entdeckt, dass die Kalibrierung nicht richtig war und musste manuell in der Simulation angepasst werden. Eine Rotationsmatrix, die in der Regel durch die Kalibrierung gerechnet und umgesetzt wird, wurde hier manuell erstellt und verwendet. Um eine falsche Nachkalibrierung zu verhindern, wurde der zuständige Teil deaktiviert. Es ist wichtig zu wissen, dass die Position des Geräts sich während der Fahrt kaum verändert hat.

In der Abbildung 2.15 ist der Fall vorgestellt, in dem kein Groundhit erkannt wurde. Während dieser Fahrt fand ebenfalls kein Unfall statt. Die oberen Grafiken zeigen die Geschwindigkeit (blau) sowie die GroundHit-Auslösungen, die entweder im Feld (grün) oder durch die Simulation (orange) gefeuert wurde, über die Zeit. Die unteren

Grafiken stellen die kinetische Energie (blau) aus der Gleichung 2.5 sowie den Energieschwellwert (orange) über die Zeit dar. Die zwei rechte Grafiken veranschaulichen einen kleinen kritischen Bereich der jeweiligen Signalen, damit der Signalverlauf an der Stelle klar betrachtet wird.

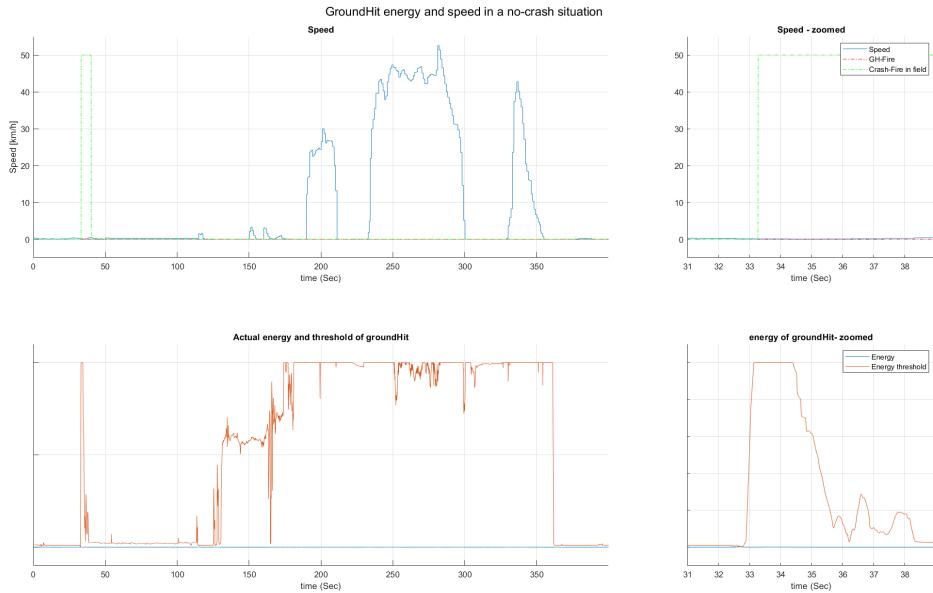


Abbildung 2.15: Verlauf der Energie sowie des Energieschwellwerts bei einer Testfahrt ohne GroundHit

An dieser Stelle hat die tatsächliche Energie (blau) den Schwellwert (orange) nicht überschritten, was eigentlich keinen GroundHit-Alarm auslösen soll. Während der Fahrt hat das Smartphone an der Stelle einen falschen Alarm ausgelöst, da die Kalibrierung nicht 100% richtig war. In den simulierten Daten wurde diese angepasst und hat dazu geführt, keinen Unfall zu erkennen.

Beispiele 2: Unfall mit GroundHit

In der Abbildung 2.16 sind die simulierten Daten einer echten Fahrt mit einem Unfall dargestellt.

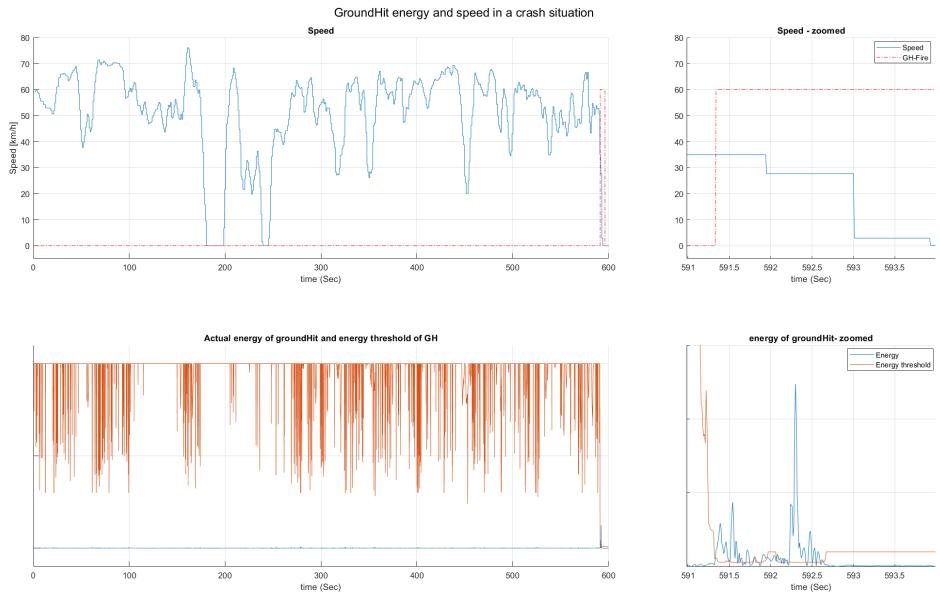


Abbildung 2.16: Verlauf der Energie sowie des Energieschwellwerts bei einer Echtfahrt mit GroundHit

In der Abbildung 2.16 sind die Daten eines Unfalls dargestellt und auf die Unfallphase gezoomt. Die oberen Grafiken stellen die Geschwindigkeit (blau) sowie die GroundHit-Auslösung (orange) über die Zeit dar. Die unteren Grafiken zeigen die kinetische Energie (blau) aus der Gleichung 2.5 sowie den Energieschwellwert (orange) im Laufe der Zeit. Die zwei rechte Grafiken veranschaulichen einen kleinen Bereich der jeweiligen Signale, in dem ein Unfall aufgetreten ist.

In der Abbildung ist zu erkennen, dass der Energieschwellwert bei der Sekunde 591 stark sinkt, da an der Stelle der Neigungswinkel ebenfalls größer wird. An dieser Stelle wird diesen Wert von der tatsächlichen kinetischen Energie überschritten, was eine Alarmauslösung feuern muss. Die obere rechte Grafik zeigt die Auslösung an der gleichen Stelle, an der den Schwellwert überschritten wird.

2.5.5 3. Komponente: CollisionHit

P.S: EnergyThreshold ändert sich nach Geschwindigkeit und Rollwinkel (Ab 45°).

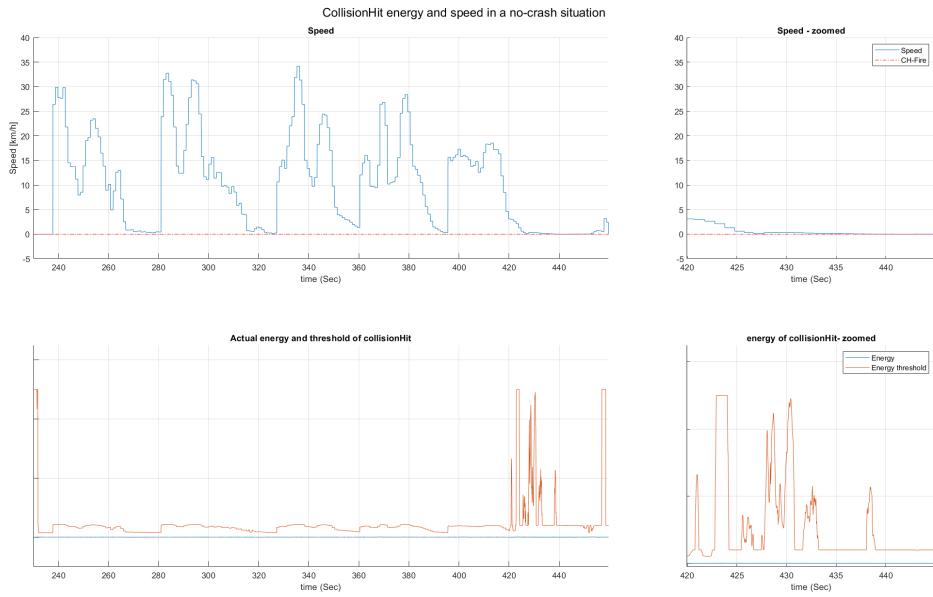


Abbildung 2.17: Verlauf der Energie sowie des Energieschwellwerts bei einer Echtfahrt ohne CollisionHit

In der Abbildung 2.17 ist der Fall vorgestellt, in dem keine Kollision erkannt wurde. Während dieser Fahrt fand ebenfalls kein Unfall statt. Die oberen Grafiken zeigen die Geschwindigkeit (blau) sowie die CollisionHit-Auslösung (orange) über die Zeit. Die unteren Grafiken stellen die kinetische Energie (blau) sowie den Energieschwellwert (orange) über die Zeit dar. Die zwei rechte Grafiken veranschaulichen einen kleinen Bereich der jeweiligen Signale, wo eine Überschneidung der Signale nicht klar ist.

Aus der Grafiken ist zu bemerken, dass der Schwellwert der Energie nicht überschritten wird, auch wenn der stark sinkt. Das entspricht ebenfalls den Erwartungen.

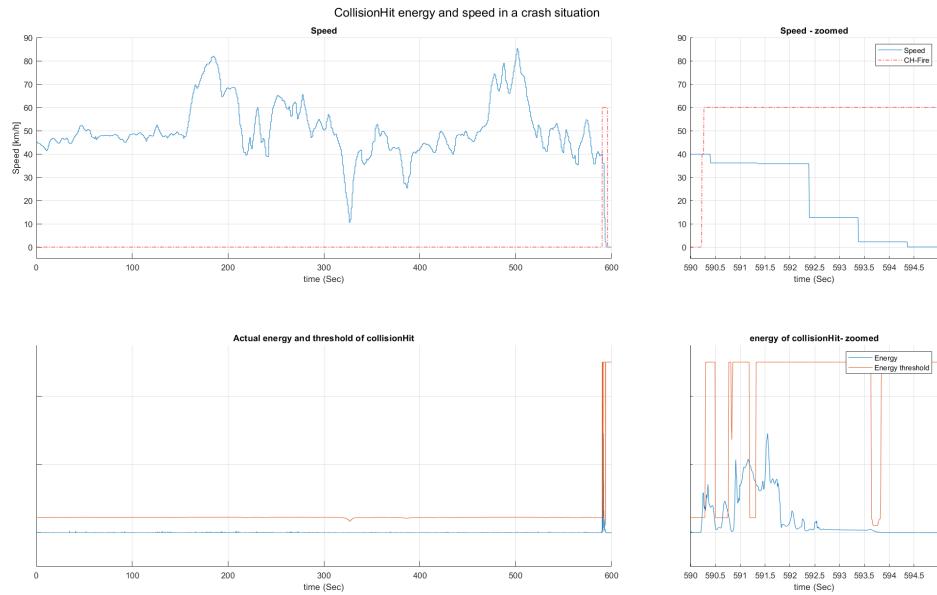


Abbildung 2.18: Verlauf der Energie sowie des Energieschwellwerts bei einer Echtfahrt ohne CollisionHit

Analog dazu ist in der Abbildung 2.18 der Fall vorgestellt, in dem eine Kollision erkannt wurde. Während dieser Fahrt fand ebenfalls auch ein Unfall statt. Die oberen Grafiken zeigen die Geschwindigkeit (blau) sowie die CollisionHit-Auslösung (orange) über die Zeit. Die unteren Grafiken stellen die kinetische Energie (blau) sowie den Energieschwellwert (orange) über die Zeit dar. Die zwei rechten Grafiken veranschaulichen einen kleinen Bereich der jeweiligen Signalen, wo eine Überschneidung der Signale nicht klar ist.

3 Unfallerkennung im Pocket-Mode

Dieses Kapitel beschäftigt sich mit der Weiterentwicklung der Unfallerkennung im Taschenmodus und Implementierung neuer Funktionen sowie die Verifizierung dieser Änderungen. Es wird zuerst erwähnt, warum ein Pocket-Mode wichtig ist und welche Szenarien zum Vergleich mit dem originalen Unfallerkennungsalgorithmus Unterschied machen würden. Schließlich werden ein Paar Szenarien näher betrachtet und Verifikationsversuche geplant, durchgeführt und ausgewertet werden.

3.1 Grund des Pocket-Modes

Die Anzahl der App-Nutzer (Unfallerkennungsalgorithmus) lag im November 2021 bei ca. 1190. Die Anzahl der Motorräder zum gleichen Zeitraum betrugt in Deutschland ca. 4,6 Millionen. Um die Anzahl der App-Nutzer zu erhöhen, sollten die Bedürfnisse der Benutzern bekannt sein, damit diese durch erweiterte beziehungsweise neue Funktionen abgedeckt werden.

In diesem Sinne wurde eine Umfrage vom Spiegel-Institute im Zeitraum zwischen November und Dezember 2021 in vier Länder (Deutschland, Frankreich, Italien, Spanien) mit 333 Befragten jeweils durchgeführt.

Die zwei wichtigsten relevante Fragen waren:

- Wofür nutzen Sie Ihr Smartphone während einer Fahrt mit dem Motorrad?
- Wo befindet sich aktuell Ihr Smartphone während der Fahrt normalerweise?

Die Ergebnisse der Umfrage aus den vier Ländern lagen sehr nah zu einander, deswegen wird demnächst nur das Umfrageergebnis der deutschen Nutzern erläutert.

In der Abbildung 3.1 ist das Ergebnis der Umfrage aus dem deutschen Markt dargestellt. 47% der Befragten nutzen kein Smartphone während einer Fahrt, weil die Strecke bekannt ist oder weil sie Ihre Smartphones nicht am Lenker befestigen wollen. 70% der Befragten haben Ihre Handys nicht am Motorrad oder am Lenker gehabt sondern in der (Jacken)-Tasche beziehungsweise im Rucksack.

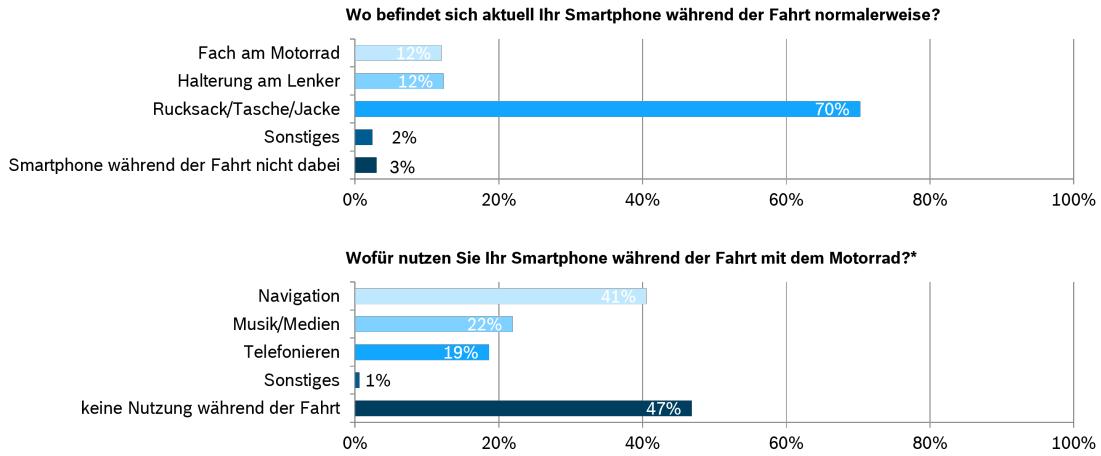


Abbildung 3.1: Ergebnisse der Umfrage vom Spiegelinstitute über das Taschenmodus

Aus diesem Grund ist die Entwicklung einer Unfallerkennung im Pocket-Mode wichtig, wo das Smartphone nicht mehr unbedingt am Lenker befestigt werden muss. Die Weiterentwicklung der Unfallerkennung wird mit agilen Methoden erfolgt.

3.2 Kritische Szenarien

Im Abschnitt 2.5 ist der Ablauf der aktuellen Unfallerkennungsalgorithmus sowie deren Parameter (z.B. TipOver) erläutert. Die Entwicklung des Pocket-Modes sollte auf keinen Fall zu Konflikten mit dem normalen Mode führen. Die aktuelle Zuverlässigkeit des Algorithmus' darf ebenso durch das Pocket-Mode nicht verringert werden, in dem ein im normalen Modus gut erkennbares Unfallszenario durch das Pocket-Mode übersehen wird.

Um solche Konflikte zu vermeiden wird eine Liste der Use- sowie Edgecases vorbereitet, in der die Erwarteten Reaktion des aktuellen Algorithmus' aufgelistet wird. Dadurch erfolgt eine Übersicht der möglichen Konflikten sowie der Fällen, wo ein falscher Alarm ausgelöst werden könnte, und gleich eine mögliche Gegenmaßnahme.

Die Abbildung 3.2 zeigt die erwähnte Liste. Da das Verhalten des Smartphones in der Hosentaschen (am Bein) und am Oberkörper unterschiedlich sein könnte, werden diese separat betrachtet. In der Spalte 'Beschreibung' ist eine nähere Erklärung des Szenarios erläutert. Die Spalte 'Erkennung durch den Algo' berichtet, ob der aktuelle Algorithmus das entsprechende Szenario richtig erkennen wird (IO: In Ordnung, NIO: Nicht In Ordnung). Unter 'Bemerkungen' ist eine weitere Erklärung des erwarteten Ergebnisses beschrieben. Bei den kritischen Szenarien, wo der Algorithmus den Fall nicht richtig erkennen würde, ist eine mögliche Gegenmaßnahme zum Korrigieren der Algorithmus-Entscheidung aufgeschrieben.

ID	Szenario-Name	Beschreibung	Algo-Reaktion (Erwartungen)		Bemerkungen	Mögliche Gegenmaßnahmen
			Handy am Körper	Handy am Motorrad		
1	Oberkörper bewegen	Umdrehen, Nach hinten schauen, Lenker mit einer Hand halten	-	IO	Kein Einfluss auf das Handy	-
2			IO	-	Kein GH, keine CH, Nur geringe Winkeländerung um die X-Y-Achsen	-
3		Nach vorne und hinten lehnen	IO	-	Kein Einfluss auf das Handy	-
4			IO	-	Kein GH, keine CH, Keine kritische Winkeländerung um die X-Y-Achsen	-
5		Seitliches Lehnen (rechts und links)	-	IO	Kein Einfluss auf das Handy	-
6			IO	-	Kein GH, keine CH, Keine Winkeländerung um die Z-Achse	-
7	Ab- und Aufsteigen		-	IO	Kein Einfluss auf das Handy	-
8			NIO	-	Bewegungsabhängig, enthält Winkeländerung (TO) und manchmal GroundHit. Falsch positiv bei GH, Kein GH -> kein Unfall	Testen
9	Laufen	Handy am Körper	NIO	-		Lauferkennungsmodul einbauen und die Unfallerkennung während des Laufen deaktivieren Erkennung durch das phone-lifting-Funktion
10	Handy in der Hand nehmen/nutzen	Eintippen, telefonieren, bewegen...etc.	NIO	-	Alles möglich (GH, CH, TO)	
11	Wheelie fahren	Nur auf das Hinterrad fahren	IO	-	Extremer Fall (Not intended use)	In AGB ausschließen
12	Auf der Motorradsitzbank stehen	-	IO	-	Extremer Fall (Not intended use)	In AGB ausschließen
13	Auf dem Motorrad (Fußraste) stehen	Stehend fahren	-	IO		Testen
14	Anhalten	Fahren, dann (stark) bremsen und Fuß runter	IO	-	Kein Einfluss auf das Handy	-
15	Normales Fahren	Beschleunigen, bremsen, Kurven fahren... ect.	-	IO	Kein GH, keine CH, Keine kritische Winkeländerung um die X-Y-Achsen	-
16	Handy in der Tasche rutscht	Mit Winkeländerung	-	IO	Im aktuellen Algo abgedeckt	Schnelles Nachkalibrierung oder in AGB bekannt machen (Handy befestigen) Fahren und Winkeländerung -> Unfall Testen ob GH oder CH erkannt werden
17		Keine Winkeländerung (gleiche Position)	-	IO	Keine Winkeländerung -> kein Unfall	
18	Motorrad abstellen	-	-	IO	Im aktuellen Algo abgedeckt	
19	Hanging off	In der Kurve (Hanging off)	-	IO	Keine ausreichende Winkeländerung für ein TipOver	

Abbildung 3.2: Die Use- und Edgecases mit der erwarteten Reaktion des Algorithmus'

Nach einer internen Statistik ist das Laufen ein häufiger Grund von den falschen Alarmauslösungen, deswegen eine Lauferkennung zur Verbesserung der Zuverlässigkeit sehr wichtig.

3.3 Lauferkennung

In der bereits bestehenden Version des Algorithmus' ist davon ausgegangen, dass das Smartphone am Lenker befestigt wird. Wenn die Person das Handy nach einer Fahrt in die Hosen- beziehungsweise Jackentasche einsteckt und fängt an zu laufen, wird öfters einen falschen Alarm (falsch-positiv) ausgelöst, da das Laufen im bisherigen Algorithmus nicht berücksichtigt wurde. Wenn die Unfallerkennung im Pocket-Mode verwendet wird, ist stark zu erwarten, dass die Person nach einer Fahrt oder während einer Pause (z.B. Tanken) vergisst (oder ignoriert), die Unfallerkennung zu deaktivieren, und mit dem Smartphone an sich läuft. Das führt dazu, dass die Anzahl der falschen Alarmangaben im Pocket-Mode wesentlich steigt.

Diese Arbeit beschäftigt sich im Teil mit der Implementierung der Lauferkennung. Das Ziel dahinter ist das Laufen zu erkennen und die Unfallerkennung temporär zu deaktivieren, damit die falsche Alarne verhindert werden. In diesem Kapitel werden die Entwicklungsschritte der Lauferkennung erläutert.

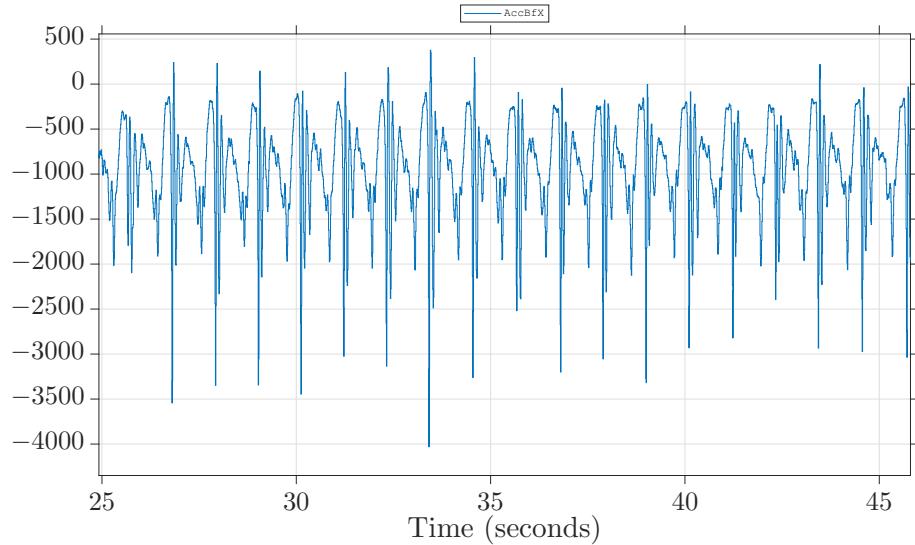


Abbildung 3.3: Beispiel: Beschleunigungssignal beim Laufen

In der Abbildung 3.3 ist ein Beispieldesign aus dem Beschleunigungssensor im Smartphone während des Laufens abgebildet. Die Person kann bis zu 2 Schritte pro Sekunde im Schnitt zurücklegen. In der Grafik können die Peaks innerhalb einer Sekunde aufgezählt werden und kann die durchschnittliche Anzahl der Schritte ermitteln. Wenn diese unter 2 pro Sekunde liegt, ist vom Laufen auszugehen, da ein Motor so wenige Umdrehungen pro Sekunde nicht schafft. Im nächsten Abschnitt werden diese Peaks aufgezählt, um die Anzahl der Schritte beziehungsweise Umdrehungen zu ermitteln.

3.3.1 Lauferkennung - Spitzenzähler

Wie bereits erwähnt wurde, kann die Person bis zu vier Schritte pro Sekunde laufen. D.h. aus einem typischen Laufsignal (z.B. Abbildung 3.3) soll maximal 4 Schritte (zwei Bewegungen pro Fuß) pro Sekunde aufgezählt werden. Es soll ein Modell implementiert werden, das die Anzahl der Schritte beziehungsweise Spitzen aufzählt und der Mittelwert pro Sekunde zurückgibt. Zur Vereinfachung der Implementierung ist eine Testumgebung (Abbildung 3.4) aufgebaut, in der ein bekanntes Sinussignal generiert, dargestellt und verarbeitet wird.

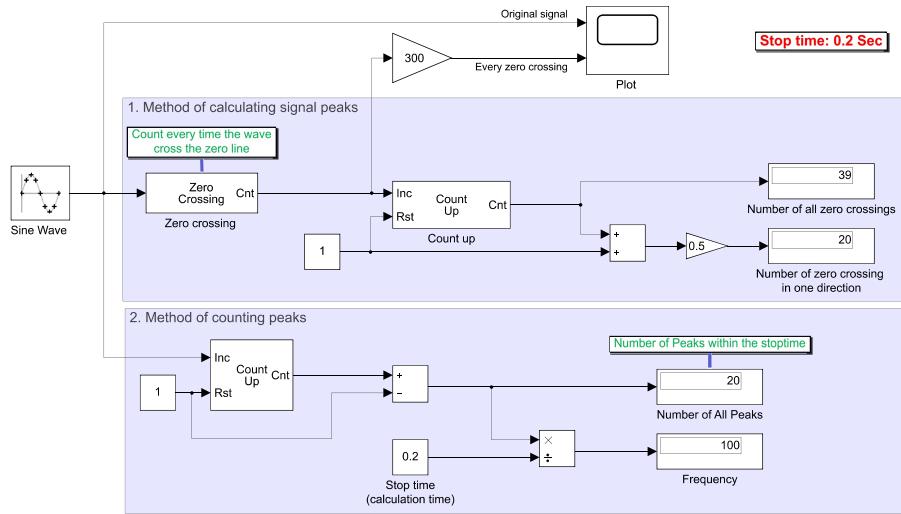


Abbildung 3.4: Testmodell der Lauferkennung - Spitzenzähler

Das generierte Sinussignal hat eine Amplitude von 325 und eine Frequenz von 100 Hz und lässt sich mithilfe eines Scopes (Simulink-Block) in der Abbildung 3.5 (blau) darstellen sowie wie oft das Signal die Nulllinie überschneidet (rot). Aus der Grafik ist die Anzahl der Peaks einfach zu ermitteln und diese beträgt in diesem Fall 100 Hz umgerechnet.

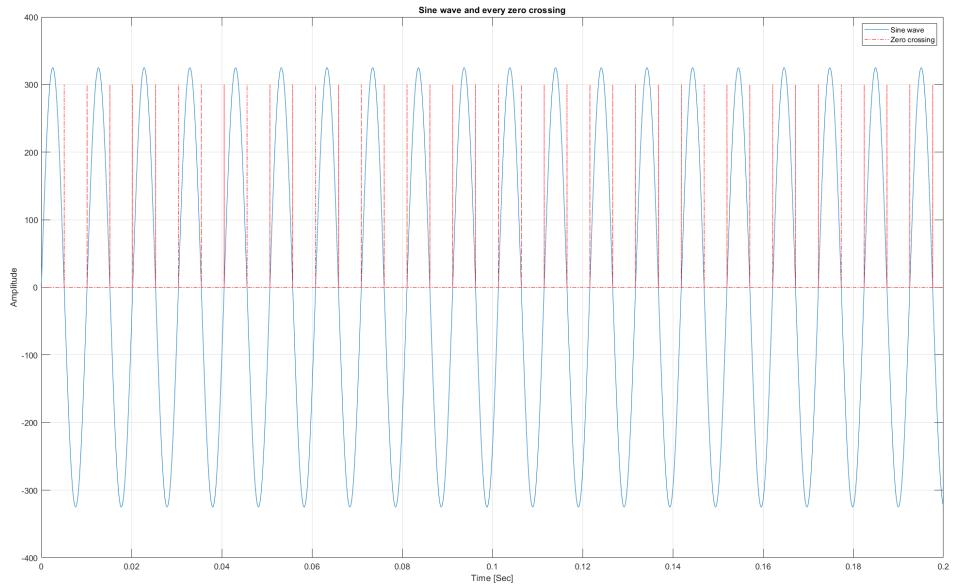


Abbildung 3.5: Darstellung des im Testmodell der Lauferkennung generierten Sinussignal sowie jede Überschneidung der x-Achse

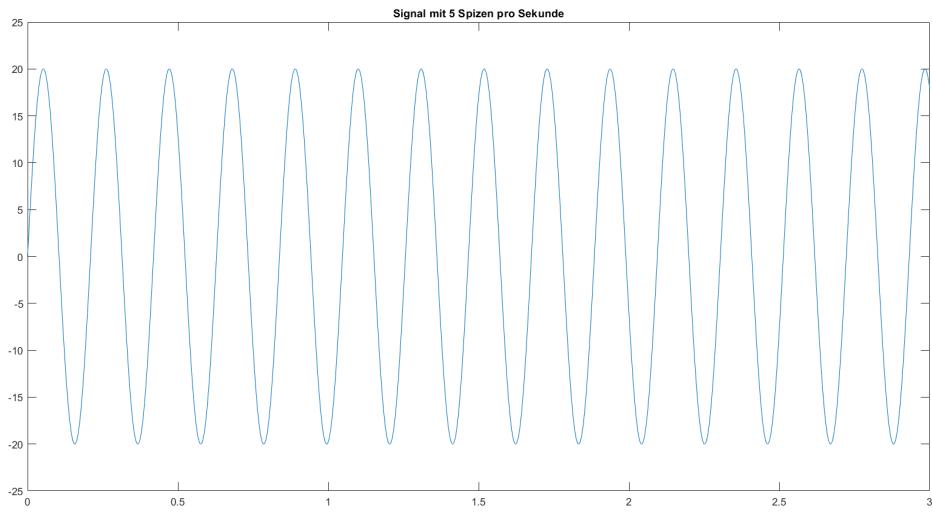
In dem Modell ist die Funktion 'Zero Crossing' verwendet. Diese zählt wie oft das Signal die x-Achse überquert. Diese Methode liefert das richtige erwartete Ergebnis, wenn das Signal um die x-Achse dargestellt ist. Diese Funktion hilft allerdings nicht, wenn das Signal ein Offset hat, in dem dieses z.B. um die Linie $y = 300$ (Verschiebung auf der y-Achse), da in diesem Fall das Signal die x-Achse (amplitudenabhängig) nicht mehr überschneidet. Das führt dazu, dass das Ergebnis nicht mehr zuverlässig ist. Eine andere Methode hat sich ergeben, dass die Funktion 'Counter up' in dem Modell verwendet wird. Dieses Block zählt wie oft das Signals in die positive Richtung geht. Die neue Implementierung hat ein zuverlässiges Ergebnis im Vergleich zum vorherigen Modell geliefert.

Beim Einsetzen des gleichen Vorgehens beziehungsweise Modell auf das richtige Laufsignal (Abbildung 3.3) wird eine Frequenz von ca. 11 Hz beim Laufen zurückgegeben, was eigentlich nicht wahr sein kann, da der Mensch keine 11 Schritte pro Sekunde zurücklegen kann.

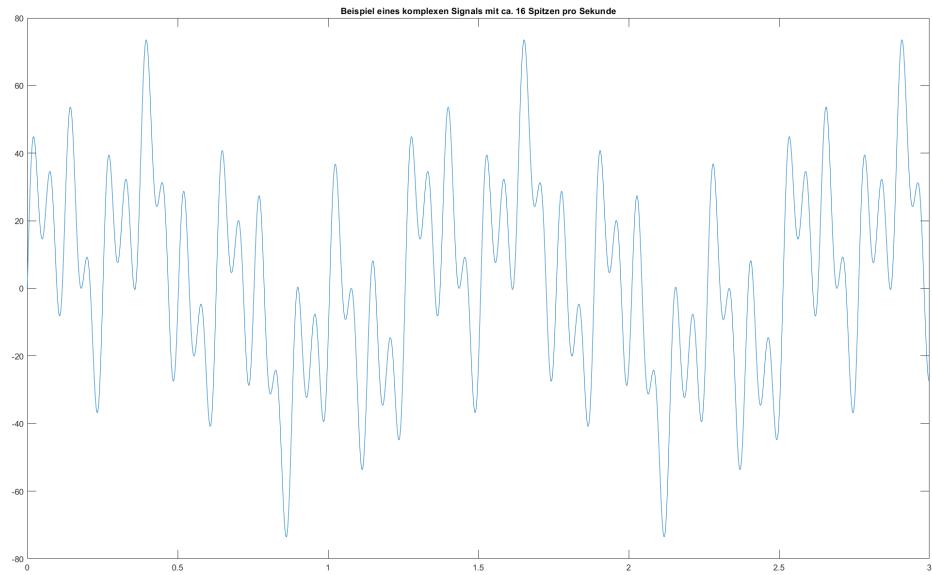
Nach weiteren Auswertungen und Forschungen wird der Grund des Fehlers entdeckt. Es liegt an den Unterschied zwischen dem einfachen generierten Sinussignal und dem echten Laufsignal. Das echte Signal hat im Vergleich zum Generierten viele Störungen (Rauschen). Diese lassen sich durch das benannte Modell nicht ausfiltert oder ignorieren, was zu einem falschen Ergebnis führt. Die Abbildung 3.6 stellt ein gutes Beispiel dieser Unterschied dar. Die Rauschen erhöhen die Anzahl der Spitzen.

Die Abbildung 3.6 zeigt zwei sinusförmige Signale. Die obere Grafik stellt ein einfaches Signal mit einer Frequenz von ungefähr $f = 5,5\text{Hz}$ und die Untere ein komplexes Signal dar. Das Modell hat für das untere Signal 19 Spitzen pro Sekunde geliefert, was

die Frequenz nicht entsprechen könnte.



(a) Beispiel eines einfaches Signal



(b) Beispiel eines komplexes Signal

Abbildung 3.6: Skizze eines einfachen ideales Signal sowie eines komplexeres Signal

Da der Spitzenzähler nicht zuverlässig funktioniert, ist eine bessere Idee notwendig.

3.3.2 Frequenzbasierte Lauferkennung

Das Laufsignal stellt ein wiederholtes Pattern dar, was auch durch eine Frequenzermittlung erkannt wird. Das Modell muss diese Frequenz ermitteln und auswerten. Analog zum Unterabschnitt 3.3.1 wird hier nochmal eine neue Hypothese festgelegt, die durch einem Testmodell überprüft werden soll.

Die Hypothese: Die Frequenz während des Laufens sollte kleiner als 2Hz sein und beim Fahren über 7Hz . Wenn eine Frequenz von über 7 Hz ermittelt wird, ist eine Laufaktivität ausgeschlossen, da ein Mensch auf keinen Fall 14 Schritte zurücklegen kann. Die Transformation vom Zeitbereich zum Frequenzbereich wird durch eine FFT erfolgt.

Die App der Unfallerkennung hat eine Abtastrate von $f_s = 100 \text{ Hz}$. D.h. es werden 100 Messwerte pro Messsekunde aufgenommen.

Bezogen auf die Nyquist-Frequenz (Gleichung 2.2) lässt sich die Bandbreite beziehungsweise minimale erkennbare Frequenz $f_n = 50 \text{ Hz}$ berechnen.

Spectrum Analyzer

In der Abbildung 3.7 sind die Eigenschaften des generierten Sinussignals mit einer Frequenz von 100 Hz zu sehen. Es wird in diesem Modell ein Block "Spectrum Analyzer" (rot markiert) als Referenz verwendet, was die einzelnen Frequenzen eines komplexen Signals zurückgibt. Der Benutzer kann die Spezifikationen vom 'SSpectrum Analyzer' einstellen. Die Ausgabe des 'SSpectrum Analyzer's ist in der Abbildung 3.8 veranschaulicht. In der oberen Grafik werden die Intensität der Frequenz(en) (auch Spektrum genannt) abgebildet und in der unteren Grafik eine 3-D-Darstellung "Frequenz-Zeit-Intensität" (Spektrogramm), wobei die Farbe die Intensität repräsentiert. Wenn die Abbildung näher betrachtet wird, ist eine Frequenz von 100 Hz gut sichtbar.

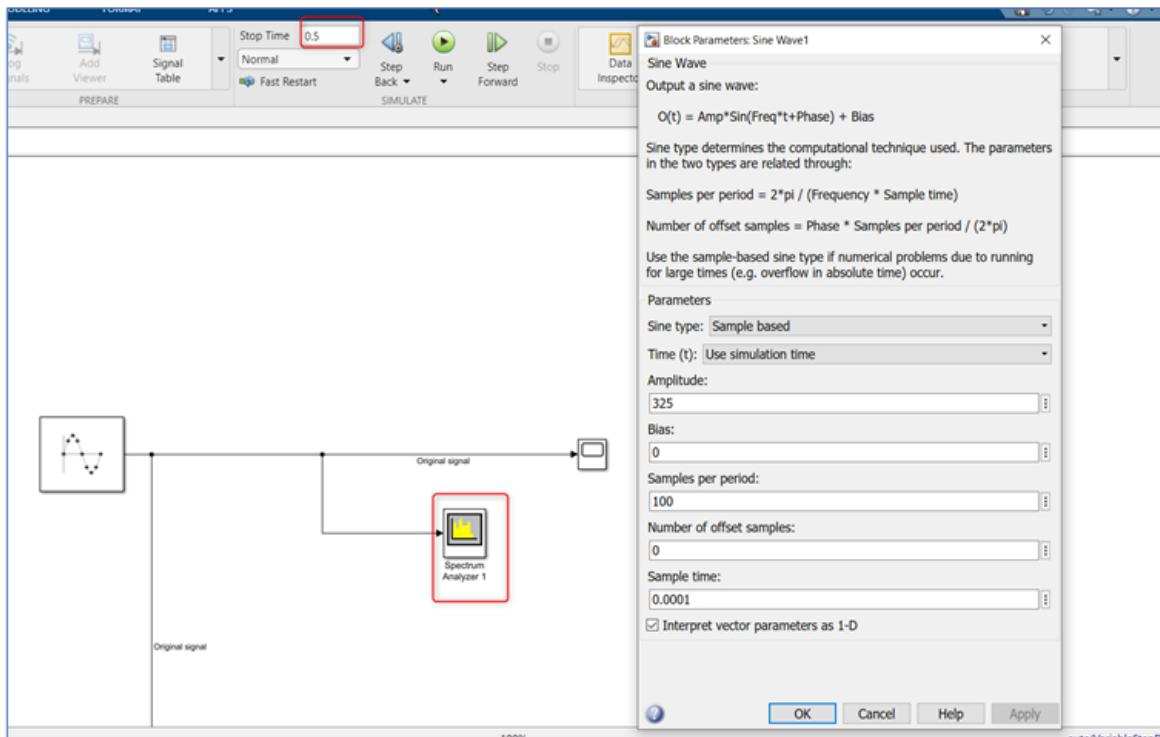


Abbildung 3.7: Testbeispiel - Frequenzbasierte Lauferkennung - Sinussignal

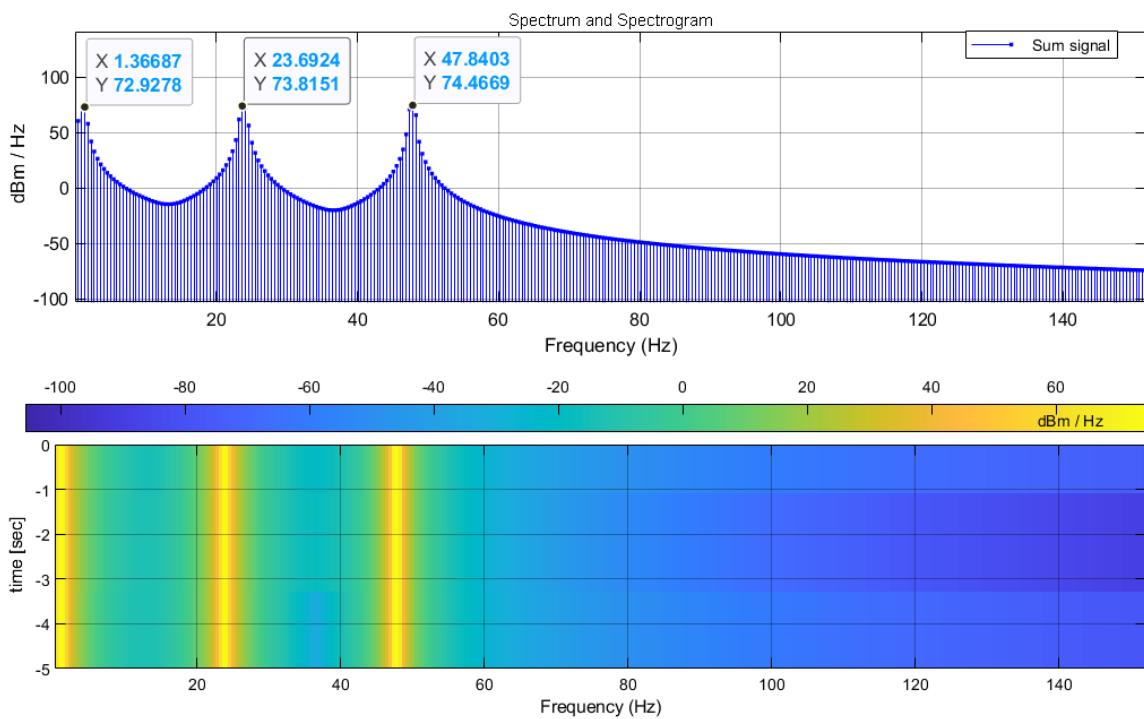


Abbildung 3.8: Testbeispiel - frequenzbasierte Lauferkennung - Ausgabe des Spektrum-Analyzers im Fall eines komplexen Signals

Testmodell

Ein Testmodell ist in der Abbildung 3.9 veranschaulicht. Das Ziel ist die Funktionalität des Prinzips zu überprüfen, bevor dieses mit einem Echtesignal verwendet wird. In dem Modell sind drei Sinussignale mit verschiedenen Frequenzen generiert, die zusammen summiert werden, um ein komplexes Signal zu erstellen. Die drei Sinussignale sowie deren Summe sind in der Abbildung 3.10 dargestellt.

Das Testmodell erstellt zuerst ein komplexes Signal mit bekannten Einzel- beziehungsweise Grundfrequenzen und konvertiert dieses mit dem 'SZero-Order-Hold"-Block zu einem diskreten Signals, da die FFT nicht auf ein kontinuierliches Signal anwendbar ist. Danach wird das FFT-Fenster durch das 'Buffer'-Block ermittelt und dann die FFT für das entsprechende Fenster verwendet. Das Ergebnis der FFT wird weiterbearbeitet, in dem der Betrag gebildet und Spiegelung entfernt wird. Das Resultat ist eine 2-D-Matrix, wobei die x-Werte die Frequenzen auf einer Skala von 1-512 sind und die y-Werte die Intensität jeder Frequenz darstellen. Das Resultat ist in der Abbildung 3.11 dargestellt. Die X-Werte (beziehungsweise Indexe) werden extrahiert und in den Skala von 1-100 umgerechnet, in dem diese mit 100/512 multipliziert. Eine vereinfachte Ablaufschema ist in der Abbildung 3.12 gezeigt. Das Endergebnis des Modells ist eine sortierte Liste der tatsächlichen Grundfrequenzen. Die ersten drei Werte haben eine wesentliche große Intensität und sind somit die gesuchten Frequenzen mit minimaler Abweichung.

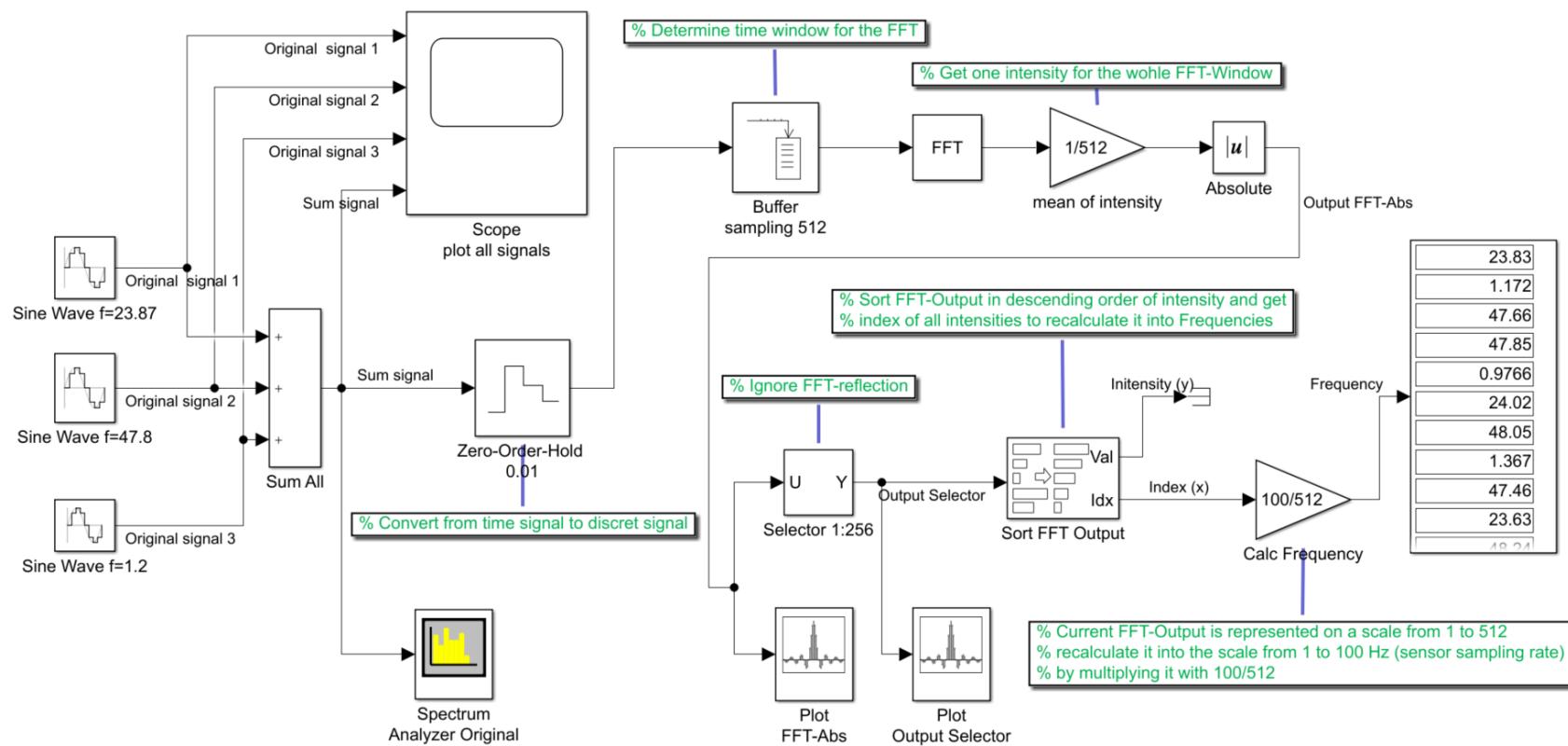


Abbildung 3.9: Testbeispiel - Frequenzbasierte Lauferkennung - FFT

3 Unfallerkennung im Pocket-Mode

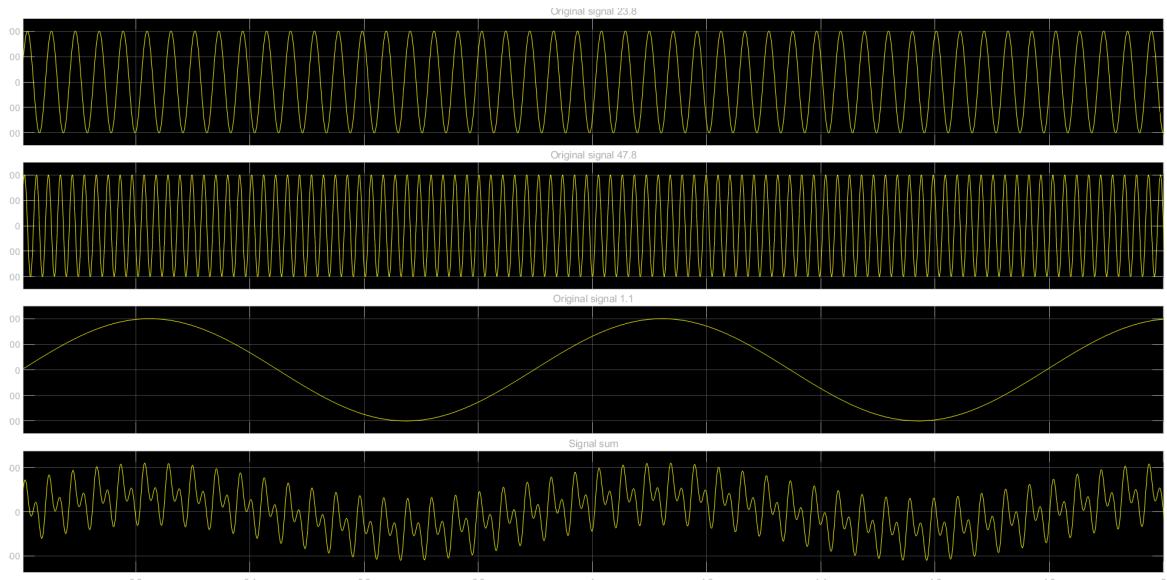


Abbildung 3.10: verschiedene Einzelsignale ($f = 23,8; f = 47,8; f = 1,1$) mit deren Summe f_g

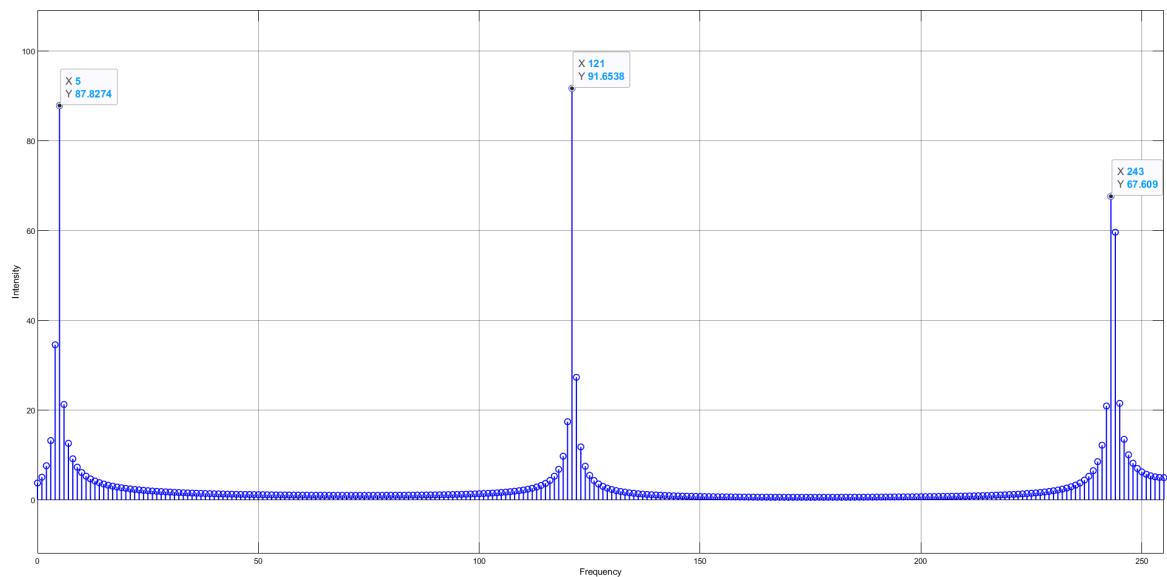


Abbildung 3.11: Das Ergebnis der FFT - Spiegelung entfernt und Beträge

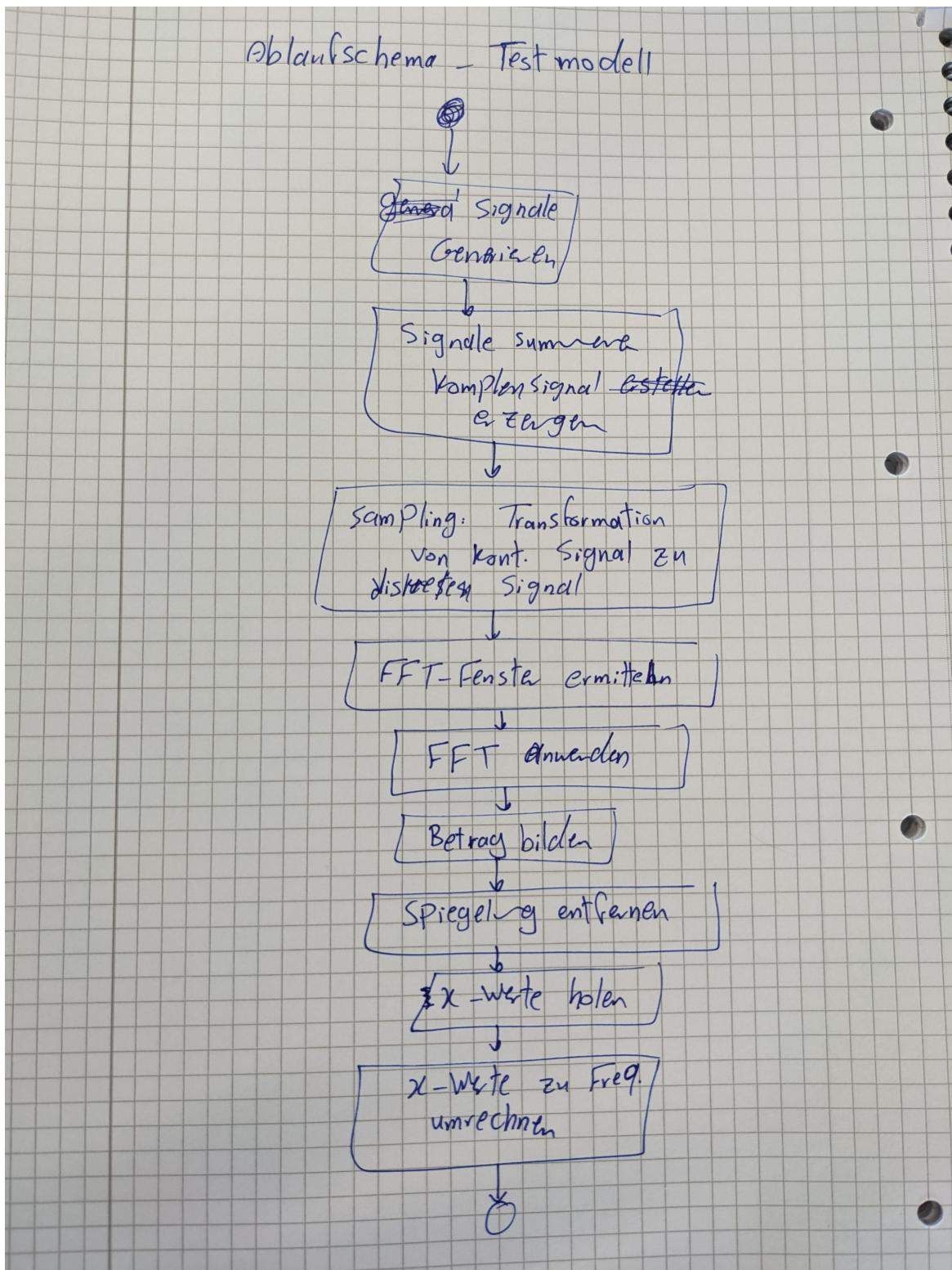


Abbildung 3.12: Ablaufschema des Testmodells der Frequenzbasierten Lauferkennung

Nachdem die Ergebnisse des Testmodells für Richtigkeit geprüft wurden, wird das

Modell mit einem Echtesignal getestet.

Anwendung auf ein Echtesignal

Zum Anwenden auf ein Echtesignal wird ein Untermodell 'MotionDetection' erstellt, das die Bewegung (Laufen oder fahren) erkennt und zurückgibt. In der Abbildung 3.13 ist einen Teil des genannten Modells sichtbar.

Im ersten Teil (1) wird der Betrag aller drei Komponenten der Beschleunigung (X,Y,Z) ausgerechnet und dieser für die Lauferkennung verwendet, um diese unabhängig von der Laufrichtung zu bewahren. Der erste Teil kann wie folgt mathematisch beschrieben werden:

$$Acc_g = \sqrt{Acc_X^2 + Acc_Y^2 + Acc_Z^2}$$

In dem zweiten Teil (2) wird eine FFT durchgeführt, um danach die Frequenzen ermitteln zu können. Das Block 'Buffer' stellt das FFT-Fenster (B_L) ein, in dem eine bestimmte Anzahl der Proben (Messungen) gesammelt wird. In diesem Modell wurde das Fenster auf $B_L = 2,56$ Sekunden (d.h. 256 Proben) mit einer Überlappung von 50% eingestellt. Die minimale erkennbare Frequenz lässt sich durch $f_{min} = \frac{1}{B_L} = 0,3906$ Hz berechnen (siehe Unterabschnitt 2.3.1).

Eine größeres FFT-Fenster B_L hätte eine bessere Frequenzermittlung gesichert und würde allerdings zu größeren Rechenaufwand und längeren Rechenzeiten führen. Die Überlappung dient dazu die Frequenzen am FFT-Fensterrand besser zu berücksichtigen. Danach wird die FFT-Spiegelung mit der Funktion 'Select' vernachlässigt. Der Ausgang dieses Teils ist eine 2D-Liste auf ein Skala von 1 bis 256, die sortiert werden soll.

Der dritte Teil sortiert die entsprechende Liste nach Intensität. Das Block 'Sort FFT Output' ergibt die Sortierten Intensitäten sowie deren Indizes aus dem ursprünglichen Matrix. Diese Indexe entsprechen die gesuchten Frequenzen auf die Skala (1-256). Mit einer Umrechnung in die Skala (1-100) lassen sich die tatsächliche Frequenzen berechnen. Die 'Select'-Blöcke dienen dazu eine Rechenzeit zu verkürzen, in dem nur die ersten zehn Frequenzen (beziehungsweise die Frequenzen mit den zehn größten Intensitäten) ausgesucht werden, da nur diese später für die Entscheidung relevant sind.

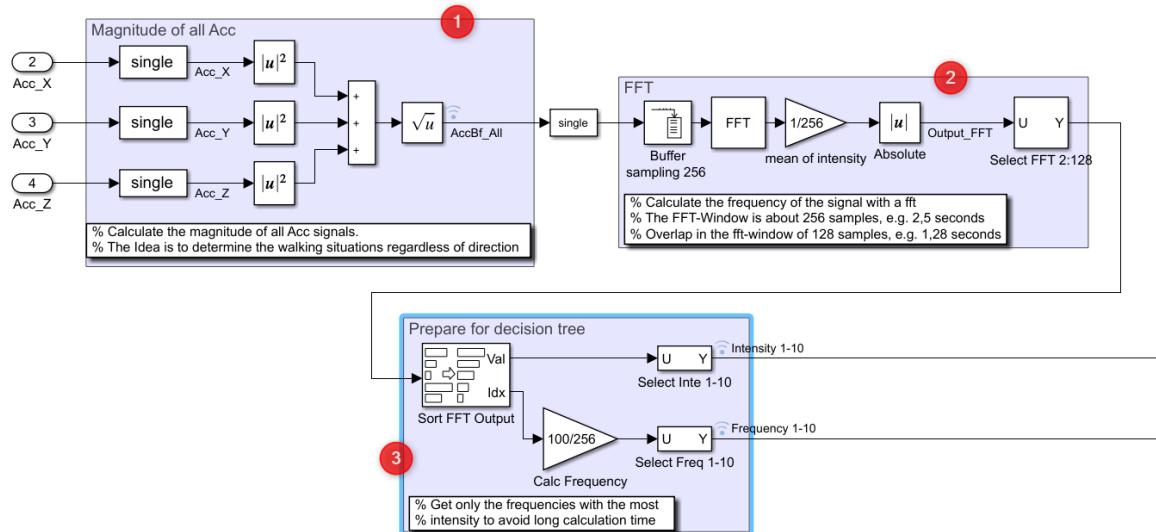


Abbildung 3.13: Das Echtmodell der Frequenzbasierten Lauferkennung

Die vom dritten Teil ausgegangenen Daten werden zu einer Matlabdatei (Abbildung 3.14) geleitet, wo eine Entscheidung getroffen werden muss.

Entscheidungskriterien - Matlabskript

Die Abbildung 3.14 zeigt die Matlab-Funktion, die die Entscheidung übers Laufen trifft. Die Eingänge der Funktion sind die zehn Frequenzen mit den höchsten zehn Intensitäten sowie die aktuelle Geschwindigkeit. Die durch die Funktion letzte erkannte Aktivität sowie deren Zeitpunkt werden auch in die Funktion geleitet. Nach dem Durchlauf liefert die Matlab-Funktion eine Id-Zahl, die eine Aktivität entspricht. Der Aktivität-ID-Zusammenhang ist in der Tabelle 3.1 abgebildet.

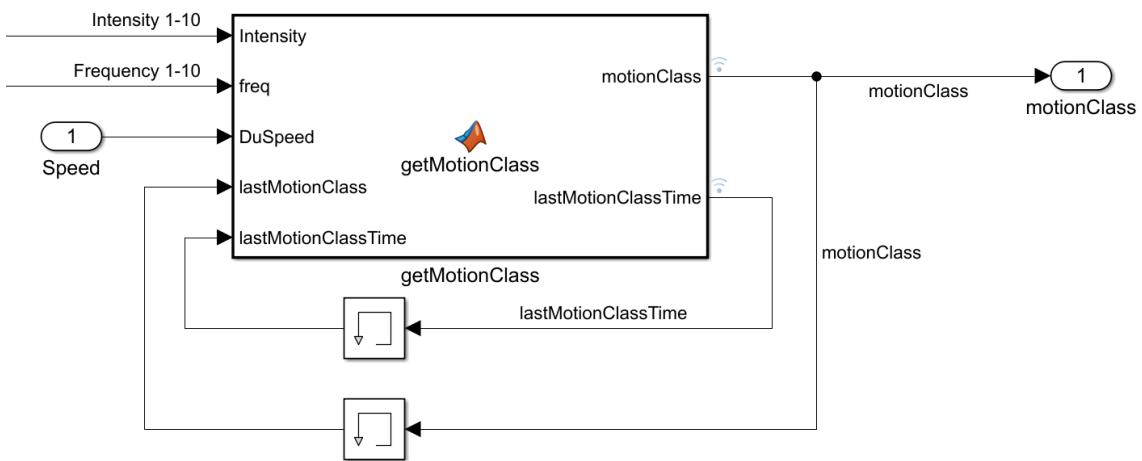


Abbildung 3.14: Ein- und Ausgänge des Entscheidungsskripts

Tabelle 3.1: Ausgangsmöglichkeiten der Entscheidungsfunktion

ID	Aktivität
-1	Konflikt/Fehler
0	Keine Bewegung
1	Laufen
2	Fahren

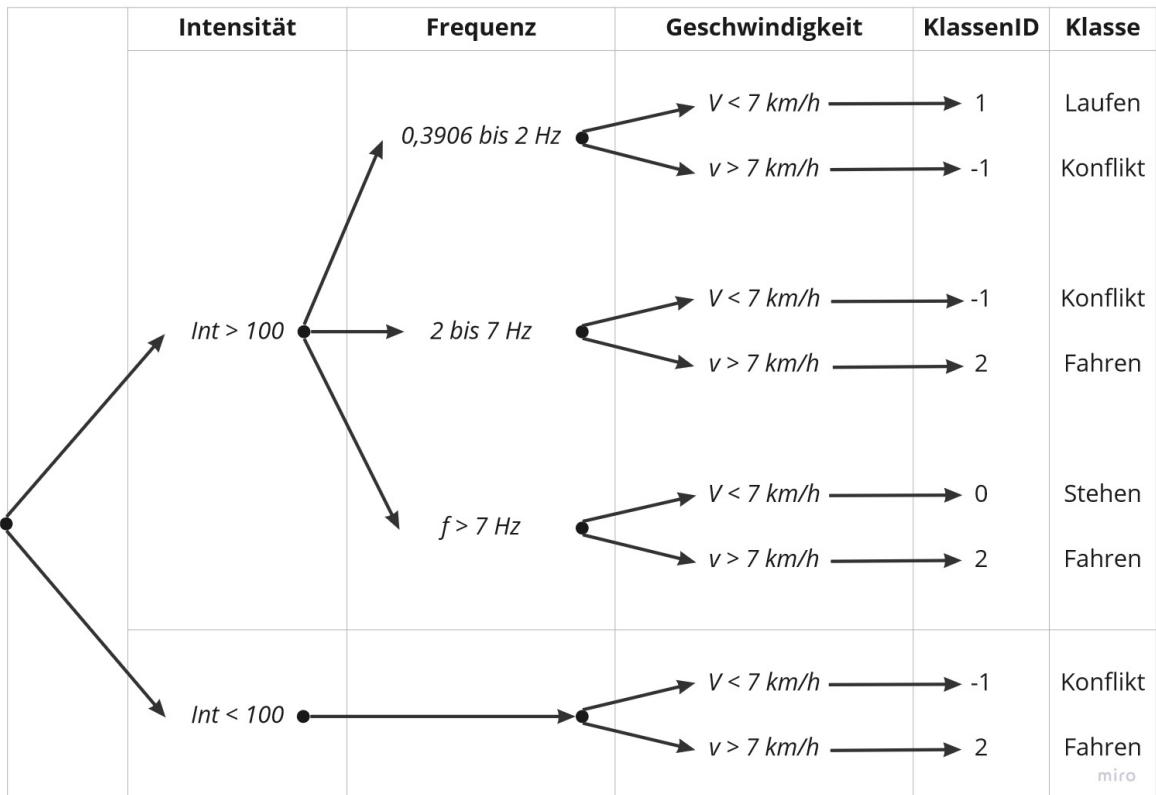


Abbildung 3.15: Entscheidungsbaum der Lauferkennung

Die Abbildung 3.15 zeigt das vereinfachte Entscheidungsbaum, wonach eine Lauferkennung erfolgt wird. Die Entscheidung erfolgt in einer Matlab-Funktion innerhalb vom Simulink-Modell (siehe Abschnitt A.1).

Die drei Hauptkriterien sind die Frequenz mit ihrer Intensität sowie die gemessene Geschwindigkeit. Wenn die Intensität einen Zulässigen Wert hat, wird die dazugehörige Frequenz berücksichtigt und danach die gelieferte Aussage mit der Geschwindigkeit nachgeprüft. In dem Entscheidungsbaum sind vier Klassen definiert.

- Stehen beziehungsweise keine Bewegung
- Laufen
- Fahren

- Konflikt: Wenn die Entscheidungskriterien (Frequenz und Geschwindigkeit) verschiedene Aussagen liefern

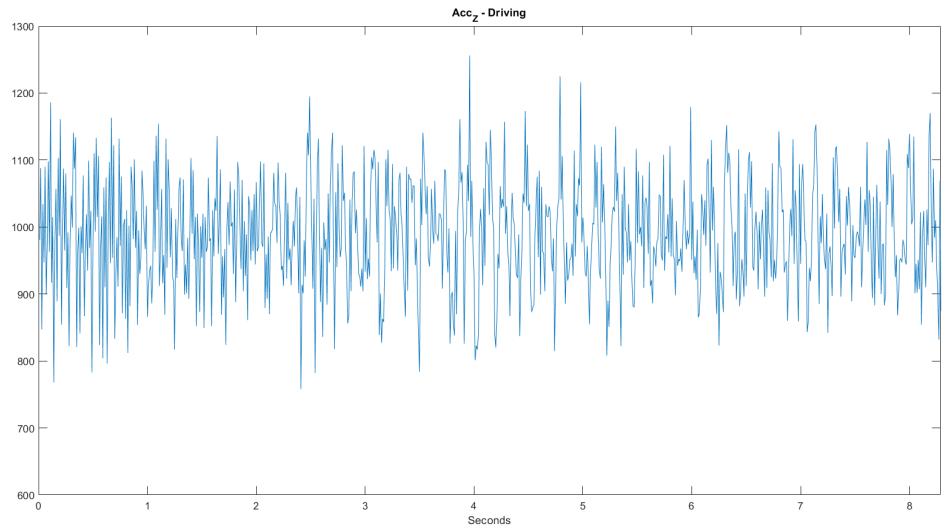
Der Entscheidungsbaum fängt bei der Intensität an. Wenn die größte Intensität kleiner als der Schwellwert (100) ist, kann die dazugehörige Frequenz für die Entscheidung nicht vertrauend sein und werden bis zu vier nachfolgenden Intensitäten untersucht. Sollte immer noch keine zulässige Intensität ergeben, wird sofort nach Geschwindigkeit geschaut und diese für die Entscheidung verwendet. Z.B. Bei einer Geschwindigkeit von 30 km/h ist von einer Fahr auszugehen und bei 3 km/h vom Laufen.

Kommt eine zulässige Intensität vor, wird die dazugehörige Frequenz berücksichtigt. Eine Frequenz unter 2 Hz bedeutet 'laufen' und über 7 Hz entspricht 'fahren'. Es wird danach mit der Geschwindigkeit nachgeprüft. Eine Geschwindigkeit von über 7 km/h bedeutet auf jeden Fall 'Fahren' und darunter 'laufen'. Wenn die Frequenz- und Geschwindigkeitsüberprüfung verschiedene Aussagen zurückgeben, ist von einem 'Konflikt' auszugehen. In diesem Fall werden bis zu größten nachfolgenden Frequenzen überprüft.

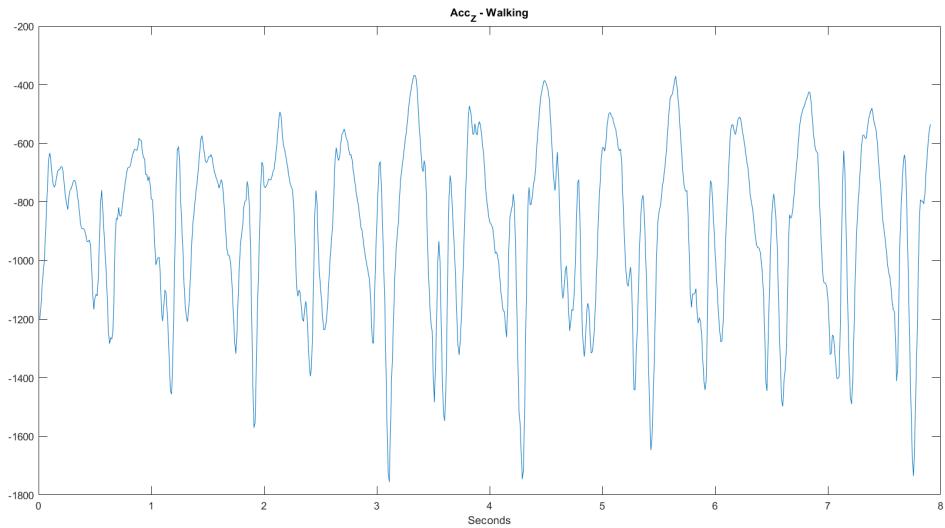
Testbeispiel

Die Signale in der Abbildung 3.16 sind aus einem Echtesignal ausgeschnitten, in dem der Fahrer nach einer Fahrt gelaufen ist. Das Fahren ist in der Abbildung 3.16a sowie das Laufen in der Abbildung 3.16b abgebildet.

Das Lauferkennung-Modell wurde mit dem entsprechenden Signal getestet. Das Modell hatte die richtigen Frequenzen erkannt. Für das erste Teilsignal aus der Abbildung 3.16a hat eine Frequenz von durchschnittlich 20 Hz geliefert. Das Teilsignal aus der Abbildung 3.16b hat eine Frequenz von etwa 1.8 Hz. Die Frequenzermittlung der Lauferkennung war sehr nah zur Realität. Nachdem die Frequenzen ermittelt wurden, sind diese ins Entscheidungsskript geleitet. Das Skript entscheidet dann mit einer Geschwindigkeitsprüfung, welche Aktivität (Fahren oder Laufen) durchgeführt wurde.



(a) Beispieldesignal - Laufen



(b) Beispieldesignal - Fahren

Abbildung 3.16: Abgeschnittene Teile eines Beispieldesignals

Sinnvolle Werte auswählen

Nachdem das Testmodell gute Ergebnisse geliefert hatte, sollen die ausgewählte Schwellwerte diskutiert werden.

- **Intensität:** Der Entscheidungsbaum sucht nach einer Intensität von über 100, damit die dazugehörige Frequenz vertrauend zur Entscheidung verwendet wird. Nach mehreren Testungen wurde eine Intensität von 100 ausgewählt. Ein niedriger Intensitätswert führt zu einer möglichen falschen Entscheidung, da die Rau-

schen beziehungsweise die kleinen Frequenzen aus dem Frequenzbereich fälschlicherweise in das Entscheidungsskript eingeleitet wurde.

- **Frequenzbereich:** das entspricht der Frequenzbereiche vom Laufen und vom Fahren.
Der Lauffrequenzbereich beträgt in der Regel kleiner als 2 Hz. Das Motorrad schafft wesentlich mehr als 7 Hz (420 Umdrehungen pro Minute).
- **Geschwindigkeit:** In dem Skript ist ein Wert von 7 km/h als Schwellwert zwischen Fahren und Laufen ausgewählt. Der Wert ist in der Straßenverkehrsordnung als Schrittgeschwindigkeit anerkannt [Bussgeldkataloge, 2022]. Erfahrungsmäßig läuft der Mensch in einer Geschwindigkeit zwischen 5 und 10 km/h.

3.4 Auf- und Absteigen

In der Umgangsphase zwischen Fahren und Laufen steigt die Person ab oder auf. Beim Auf- und Absteigen entsteht eine starke Winkeländerung in der Beinposition, da der Fahrer sein Bein über das Motorrad hebt und manchmal nach hinten streckt. Wenn das Smartphone in der Hosentasche ist, bekommt es die Winkeländerung ganz klar mit. Eine theoretische Betrachtung dieser Bewegung schlägt einen Fehlalarmauslösung vor, da die Winkeländerung zu der Sitzposition über 45° liegt, was in der Regel durch das Modell 'TipOver' als ein Umkippen erkannt wird. Aus diesem Grund wird dieses Szenario für die Richtigkeit getestet.

3.5 Anhalten

In diesem Abschnitt wird ein Szenario aus der Tabelle analysiert, in dem der Fahrer während einer Fahrt an eine Ampel für kurze Zeit anhält und sein Fuß runter setzt. In diesem Fall ist das Smartphone in der Hosentasche des bewegenden Beins platziert. Die Annahme, dass in so einem Fall im Pocket-Mode ein falscher Alarm ausgelöst werden könnte. Der Grund ist die Winkeländerung von ca. 90° zwischen den zwei Positionen, was das Modell 'TipOver' aktiviert und zu einer Alarmauslösung führt. Die erste Position ist das Bein während einer Fahrt mit der horizontalen Beinstellung. Wenn der Fuß am Boden ist, steht das Bein in einer vertikalen Postion.

In der Abbildung 3.17 ist ein Beispielsignal eines Smartphones abgebildet. Die Grafik stellt das Signal in zwei Smartphone-Positionen (Vertikal und Horizontal) dar und zeigt einen Winkelunterschied von ca. 90°. Diese Winkeländerung aktiviert das "TipOver"-Modell und schlägt einen Unfall vor. Dieses Szenario wird ebenfalls für die Richtigkeit getestet und analysiert.

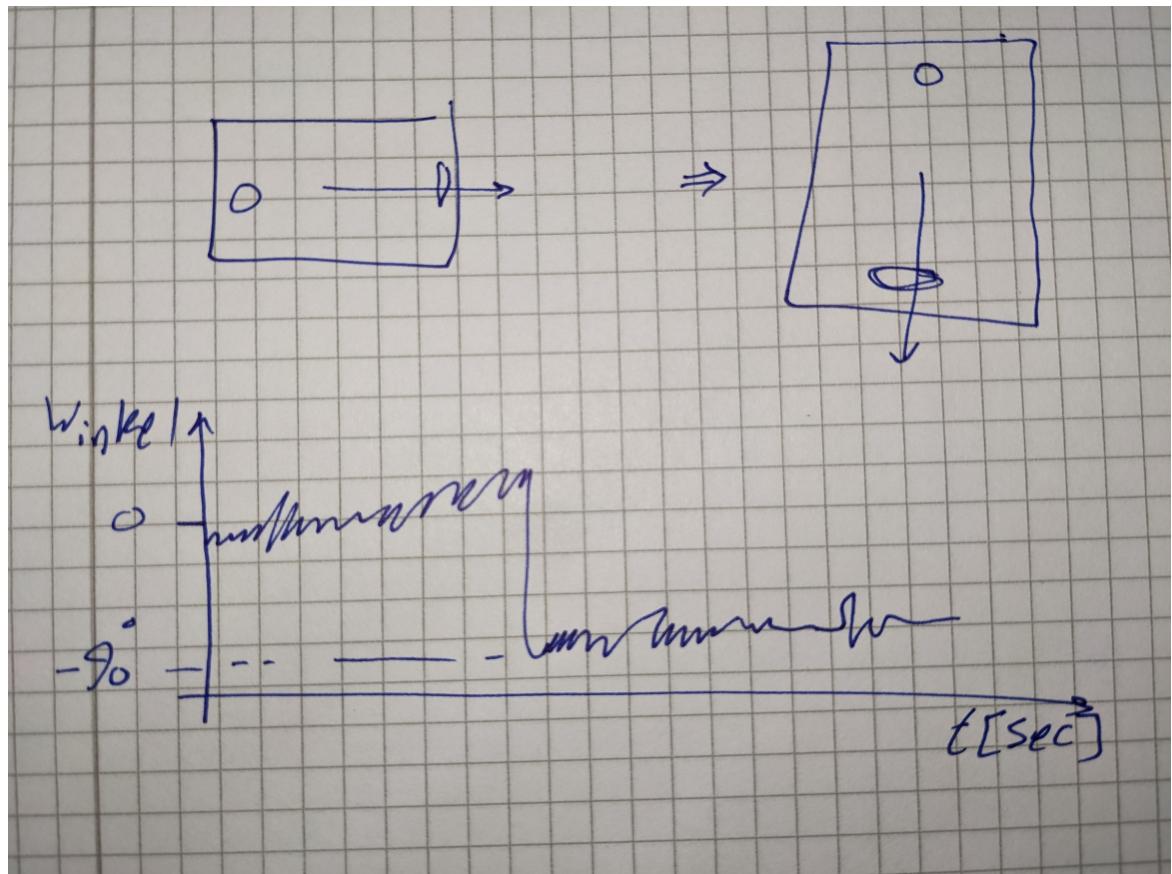


Abbildung 3.17: Winkeländerung im Signal zwischen vertikaler sowie horizontaler Smartphone-Positionierung

3.6 Verifikation des Algorithmus'

Die Verifizierung dient dazu, die Annahmen und Hypothesen aus den letzten Abschnitten zu prüfen, sowie die implementierte Lauferkennung zu testen.

Aufgrund dieser Verifizierung wird eine Versuchsplanung durchgeführt.

3.6.1 Versuchsplanung

In diesem Abschnitt wird auf die zu testenden Szenarien sowie deren Durchführung eingegangen.

Die Abbildung 3.2 zeigt eine Liste der Use- und Edgescases, in der einige Szenarien getestet werden sollen. Es wurden diesbezüglich 5 verschiedene Szenarien geplant. Die Tabelle (Abbildung 3.18) fasst diese zusammen.

Sz-Nr.	Anz. Durchläufe	Beschreibung	Smartphones				
			iPhone SE	Samsung A5	Samsung S10	Pixel 3	Huawei P30
1	2	Fahren, Auf der Fußraste stehen, Auf- und Absteigen, Laufen	Lenker	Hosentasche	Tankrucksack	Brusttasche	Rucksack
2	1	Anhalten	Lenker	X	X	Hosentasche	Hosentasche
3	1	Handy im Tankrucksack rutschen lassen (eine Ebene)	Lenker	X	X	Tankrucksack	Tankrucksack
4	1	Handy im Tankrucksack rutscht (Mehrere Ebenen)	Lenker	X	X	Tankrucksack	Tankrucksack
5	2	Crash simulieren; Rucksack während einer Fahrt fallen lassen	Lenker	X	X	Rucksack	Rucksack

Abbildung 3.18: Die Zusammenfassung der getesteten Szenarien

Der Versuchsablauf wurde bereits vor dem Testen vorbereitet und möglichst detailliert geplant, damit während des Testens kein Zeitverlust entsteht. Demnächst werden die allgemeine Schritte des Versuchsablaufs aufgelistet.

- 1 Befestigung und Einstellung der Kameras am Lenker und an der Brust
- 2 Befestigung der Smartphones (inklusive das Referenzhandy am Lenker) in der geplanten Stellen, Einschaltung und Prüfung der Signalaufnahme
- 3 Eine Kalibrierungsfahrt
- 4 Prüfung der Smartphoneskalibrierung
- 5 Start der Videoaufnahme und Dokumentation von Datum sowie Uhrzeit
- 6 Durchführung des Testversuchs (Variiert je nach Versuchsszenario)
- 7 Fahrtende und absteigen
- 8 Datenexport und -Übertragung
- 9 Datenexistenz im Zielordner prüfen
- 10 Löschung der Daten im Smartphone
- 11 Vorbereitung des nächsten Versuchs beziehungsweise Durchlaufs

Das erste Szenario ist das Wichtigste für diese Arbeit, weil dadurch die implementierte Lauferkennung sowie den Unterschied der Aktivitätserkennung zwischen den verschiedenen Bewegungsarten (z.B. Laufen, Fahren) getestet und verifiziert werden. Die einzelne Testschritte dieses Szenario werden hiermit erläutert:

- Szenario 1: Fahren mit verschiedener Fahrerpositionierung und Laufen sowie der Übergang dazwischen (Auf- und Absteigen)
 - 1 Start der Videoaufnahme und Dokumentation von Datum sowie Uhrzeit
 - 2 Normale Fahrt für ca. eine Minute
 - 3 Fahrt in stehender Fahrerposition für ca. 10-20 Sekunden
 - 4 Normale Fahrt für ca. eine Minute
 - 5 Anhalten und Absteigen
 - 6 Laufen für 30-40 Sekunden
 - 7 Wieder Aufsteigen und für ca. eine Minute normal weiterfahren
 - 8 Fahrt in stehender Fahrerposition für ca. 10 Sekunden
 - 9 Normale Fahrt für ca. eine Minute
 - 10 Fahrtende und absteigen

3.6.2 Referenz-Aktivitätsdaten (Ground truth)

Die Lauferkennung ergibt die Aussage (Fahren oder Laufen) und soll demnächst getestet werden. Für die Testung muss der tatsächliche Verlauf einer Fahrt bekannt gegeben werden. Zu diesem Zweck wurden alte Crashtests mit Videoaufnahmen optisch manuell mithilfe eines intern entwickelten LabVIEW-Tool ausgewertet werden. Die Crashtests wurden mit mehreren am Motorrad befestigten Smartphones. Verschiedene Unfallszenarien wurden geprüft, wie z.B. gegen die Wand fahren oder während der Fahrt an einer Kurve rutschen. Diese Crashtests wurden mit Videos aufgenommen, die nur einen kleinen Teil der aufgezeichneten Signal abgedeckt haben. Damit die Videos dem richtigen Signalteil zugeordnet werden können, ist eine Synchronisierung zwischen dem Signal und dem dazugehörigen Video erforderlich.

Der Benutzer kann mit dem internen Tool die Video-Signal-Synchronisierung unkompliziert erfolgen. Danach können bestimmte Labels (z.B. Fahren oder Stehen) für die entsprechenden Zeitfenster schnell und einfach eingegeben werden. Am Ende wird eine Tabelle exportiert, welche die gleiche originalen Daten sowie eine neue zusätzliche Spalte mit den Label-Ids der Aktivitäten beinhaltet. Diese Tabelle kann mit den Ergebnissen der Lauferkennung verglichen werden.

Die erwähnte Labels muss der Benutzer vorher definieren und in das Tool importiert. In der Abbildung 3.20 ist die Tabelle der definierten Labels sichtbar. Die Tabelle hat sieben verschiedene Klassen, die sich im Tool mit den F-Tasten einfach eingeben lassen. 'Undefined' entspricht unbekannter Eingabe, wenn das Motorrad nicht im Video sichtbar ist. 'Post-Crash' repräsentiert der Motorradstand nach einem Aufprall beziehungsweise einem Sturz bis zum Ruhestand.

In der Abbildung 3.19 ist ein verkürztes Beispiel der exportierten Tabelle, in der die neue Spalte 'GroundTruthId' sichtbar ist.

Timestamp	accTime	accX	accY	accZ	speed	GroundTruthID	GroundTruth
3736765632	39839,867	460	-484	1307	28,244		2 Drive
3736765632	39839,879	-1	-493	1484	28,244		2 Drive
3736765632	39839,887	317	-395	1534	28,244		2 Drive
3736765632	39839,898	66	4	1079	28,244		2 Drive
3736765632	39839,91	214	170	826	28,244		2 Drive
3736765632	39839,918	-85	8	919	28,244		2 Drive
3736765632	39839,93	164	-137	1012	28,244		2 Drive
3736765632	39839,937	-74	-172	953	28,244		2 Drive
3736765632	39839,949	364	-301	1038	28,244		2 Drive
3736765632	39839,957	13	-79	1055	28,244		3 Crash
3736765632	39840,504	14	29	946	28,244		3 Crash
3736765632	39840,516	370	-107	1455	28,244		3 Crash
3736765632	39840,523	-36	74	1086	28,244		3 Crash
3736765632	39840,535	503	322	1010	28,244		3 Crash
3736765632	39840,547	248	0	1173	28,244		3 Crash
3736765632	39840,555	212	-281	1416	28,244		3 Crash
3736765632	39840,566	19	-184	1148	28,244		3 Crash
3736765632	39840,574	448	-163	1072	25,231		3 Crash
3736765632	39840,586	157	-91	1083	25,231		3 Crash
3736765632	39840,594	92	57	1113	25,231		3 Crash
3736765632	39840,605	293	191	878	25,231		3 Crash
3736765632	39840,613	166	83	825	25,231		4 PostCrash
3736765632	39840,625	232	-285	1155	25,231		4 PostCrash
3736765633	39840,633	78	-281	1193	25,231		4 PostCrash
3736765633	39840,645	336	-228	1286	25,231		4 PostCrash
3736765633	39840,656	249	-214	1441	25,231		4 PostCrash
3736765633	39840,664	380	-454	1851	25,231		4 PostCrash
3736765633	39840,676	11	-14	1323	25,231		4 PostCrash
3736765633	39840,684	376	-22	1095	25,231		4 PostCrash
3736765633	39840,695	108	35	1139	25,231		4 PostCrash
3736765633	39840,703	406	-98	1528	25,231		4 PostCrash
3736765633	39840,715	-26	-238	1341	25,231		4 PostCrash

Abbildung 3.19: Beispiel der exportierten Tabelle mit der neuen Spalte

Label Enum	Class	F-Key Shortcut
-1	UNDEFINED	Escape
0	No Movement	F12
1	Starting/Stopping	F1
2	Driving	F2
3	Impact	F3
4	Post-Crash	F4
5	Lying	F5
6	Standing	F6

Abbildung 3.20: Die definierte Labels von Groundtruth

Demnächst wird die Versuchsplanung sowie das Versuchsvorgehen erläutert.

Datenauswertung

4 Ergebnisse

- Auswertung (Qualitative Auswertung)
- Lauferkennung
- Rechenzeit: kein wesentlicher Unterschied (55 Sec und 57 Sec)

4.1 Verifikationsversuche

- Alle geplanten Szenarien wurden mind. einmal getestet.
- Die erste sowie letzte Szenarien wurden jeweils zweimal getestet.

Nach der Durchführung der Verifikationsveruchen werden die Ergebnisse analysiert und diskutiert. Die Lauferkennung ist ein großer Teil dieser Arbeit und wird demnächst mit den tatsächlichen Daten sowie die integrierte Aktivitätserkennungsfunktion verglichen.

4.2 Lauferkennung

In der Abbildung 4.1 werden die verschiedenen Methoden der Aktivitätserkennung sowie die Geschwindigkeit (oberste Grafik) dargestellt.

Die zweite Grafik zeigt die tatsächlichen Aktivitätsdaten (GroundTruth) und die Dritte zeigt die Ausgabe der im Rahmen dieser Arbeit implementierten Lauferkennung. Die letzte Grafik stellt die Ausgabe der im Smartphone integrierten Aktivitätserkennung dar. Diese drei Grafiken unterscheiden zwischen drei Klassen:

**** Die Liste der Klassen als Tabelle hinzufügen. ****

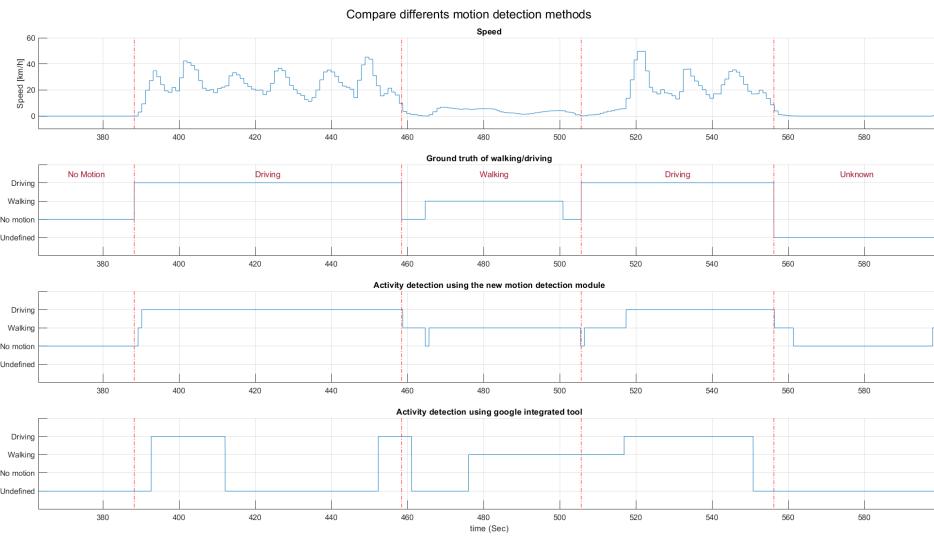


Abbildung 4.1: *Ergebnis des Lauferkennungsmodells*

Aus der Grafiken ist zu bemerken, dass die Klassifizierung durch das implementierte Modell sehr gute Ergebnisse liefert, die mit der Wahrheit mehr als 85% übereinstimmt. Im Vergleich zu der Google-Aktivitätserkennung hat das Modell eine wesentlich bessere Aussage getroffen.

4.3 Verschiedene Fahrerpositionierung

Die Abbildung 4.1 zeigt vier verschiedene Grafiken mit der gleichen x-Achse (Zeit [Sec]). In der ersten Grafik ist die Geschwindigkeit (blau) sowie die Alarmauslösung (lila) über die Zeit abgebildet. Die zweite Grafik stellt die tatsächliche Daten der Fahrerposition (Sitzen, Stehen oder unbekannt) über die Zeit dar. In der dritten Grafik ist der tatsächliche Verlauf Fahreraktivität (Fahren, Laufen oder unbekannt) dargestellt. Die Kurve aus der letzten Grafik zeigt den Verlauf des Gerätewinkels im Vergleich zur ursprünglichen Platzierung während der Kalibrierung.

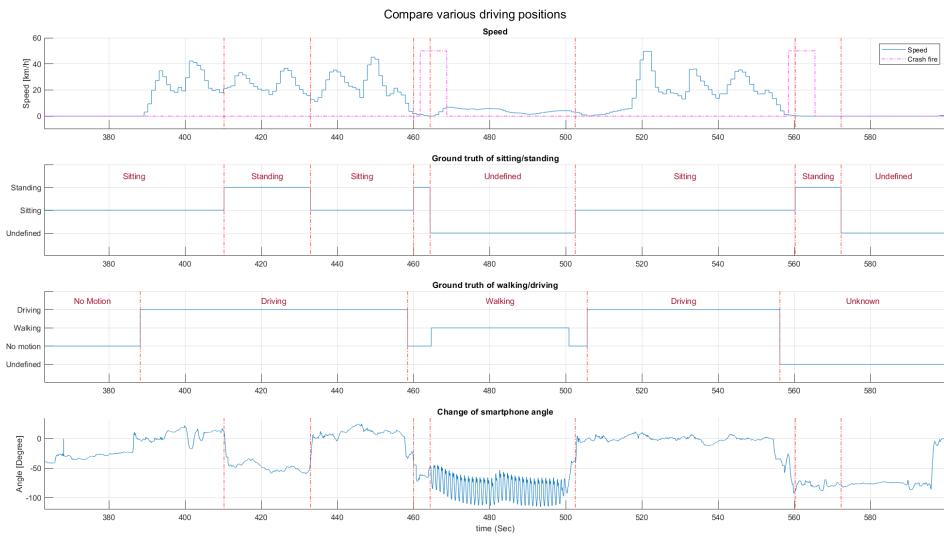


Abbildung 4.2: Vergleich der Fahrt in vertikaler und normaler Position

Aus der Abbildung 4.1 ist Folgendes zu verstehen:

- In der Zeitbereich zwischen 388 – 458 s findet eine Fahrt statt, während dieser die Fahrerposition sich verändert hat. Der Fahrer ist für ca. 10 s gestanden und wieder gesessen.
- In der Zeitbereich zwischen 465 – 505 s ist die Versuchsperson gelaufen, deswegen lautet die Klassifizierung in der zweiten Grafik 'Ünbekannt‘, da die gedachte Klassen so ein Fall nicht abdecken.
-

4.4 Anhalten

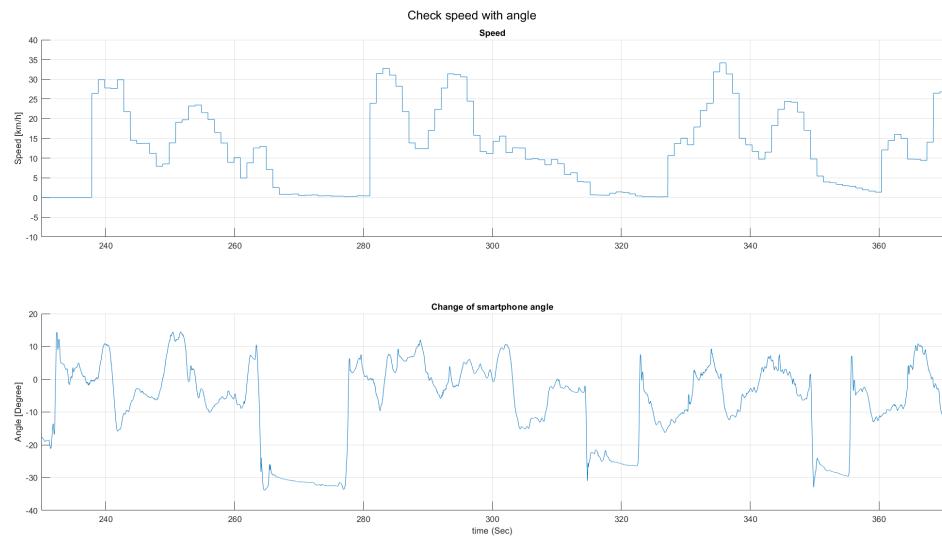


Abbildung 4.3: Winkeländerung des Smartphones durch das Anhalten

Winkeländerung nicht über 45° beträgt –> kein Fehlalarm.

Beurteilung:

Zu diesem Zweck wurden mehreren Tests durchgeführt, während Diesen keine falsche Alarmauslösungen aktiviert wurden. Grund ist, dass die Winkeländerung nicht 90° beträgt sondern nur ca. 20° - 30° (Abbildung 4.4). Die Person hat sein Bein beim Sitzen nicht genau horizontal sondern leicht nach Unten geneigt. Und wenn der Fahrer sein Fuß runter setzt, ist diese auch nicht genau vertikal sondern bisschen gebogen mit einem Winkel von ca. 10° - 20° zum Vertikallinie. D.h. die Winkeländerung ist nicht über 45° und sollte zu keinen Alarmauslösungen führen..



(a) Beinposition während einer Fahrt



(b) Beinposition Beim Stehen

Abbildung 4.4: Beinpositionen während einer Fahrt und gestreckt

4.5 Vor- und Nachteile

- Rechenzeit: kein wesentlicher Unterschied (55 Sec und 57 Sec)

5 Ausblick

- ...
- Quantitative Auswertung
- Aktivitätserkennung immer aktiviert haben und beim Motorradfahren Unfallerkennung automatisch aktivieren (App im Hintergrund laufen)
- Zuverlässiger machen (andere Messwerte verwenden und auswerten) z.B. App-Nutzung, Bildschirmaktivität.
- Phone-lifting-Funktion einbauen und nutzen -; Flase-Alarm unterdrücken, wenn das Handy benutzt wird.
- Unfallerkennung auf anderen Transportmitteln erweitern. Erstmal Mofas und dann auch Scooter, oder Stützerkennung auch beim Laufen.
- Ermittlung einer Unfallschwere besser und automatisch machen. Ziel ist eine Zwischenphase (der Agent ruft an und checkt nach) zu stoppen und Zeit zu sparen. Bei einem schweren Unfall das Krankenwagen automatisch anrufen.

Literaturverzeichnis

- [Blaabjerg 2004] BLAABJERG, F. Iov; A. D. Hansen; P. Sorensen; F.: Wind Turbine Blockset in Matlab/ Simulink / Institute of Energy Technology, Aalborg University. Version: 2004. <https://www.osti.gov/etdeweb/servlets/purl/20613486>. 2004. – Forschungsbericht
- [Brunskill 2019] BRUNSKILL, Vicki-Lynn: *Sprint (software development)*. <https://www.techtarget.com/searchsoftwarequality/definition/Scrum-sprint>. Version: August 2019. – Aufgerufen am 26.09.2022
- [Bussgeldkataloge 2022] BUSSGELDKATALOGE: *Schrittgeschwindigkeit*. <https://www.bussgeldkataloge.de/schrittgeschwindigkeit/>. Version: September 2022. – Aufgerufen am 18.10.2022
- [Deiss 1999] DEISS, J. Yeazel; J. Mehaffey; S. Penrod; A.: *GPS Speed*. <http://www.gpsinformation.net/main/gpsspeed.htm>. Version: Oktober 1999
- [Developers 2022] DEVELOPERS, Android: *Motion sensors*. https://developer.android.com/guide/topics/sensors/sensors_motion#sensors-motion-accel. Version: August 2022. – Aufgerufen am 01.08.2022
- [FAA 2021] FAA: *Satellite Navigation - GPS - How It Works*. https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks. Version: Februar 2021. – Aufgerufen am 01.08.2022
- [H. Appel 2002] H. APPEL, D. V. G. Krabbel K. G. Krabbel: *Unfallforschung, Unfallmechanik und Unfallrekonstruktion*. 2002
- [Hädrich 2012] HÄDRICH, Christian: *Messung der Schräglage von Motorrädern bei Kurvendurchfahrt*, RWTH Aachen University, Diplomarbeit, 2012
- [Karris 2008] KARRIS, S. T.: *Introduction to Simulink® with Engineering Applications*. Orchard Publications, 2008
- [Kasnakoglu 2015] KASNAKOGLU, C. A. Cansalar; E. Mavis; C.: Simulation Time Analysis of MATLAB/Simulink and LabVIEW for Control Applications. (2015). <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7125143>
- [d. Maire 2020] MAIRE, F. d.: *Analyse der Verkehrssicherheit von Motorradfahrern zur prädiktiven Bestimmung von Normalfahrsituationen über kritische Situationen bis hin zu Unfällen*. 2020

- [MostrandomComps 2019] MOSTRANDOMCOMPS: *Motorcycle Crash Compilation.* https://www.youtube.com/watch?v=dMsYT_7xNfg&ab_channel=MostrandomComps. Version: November 2019. – Aufgerufen am 06.09.2022
- [Moto-Passion 2018a] MOTO-PASSION: *Animals Vs Bikers.* https://www.youtube.com/watch?v=KiRydLwtAyc&ab_channel=MotoPassion. Version: Juli 2018. – Aufgerufen am 06.09.2022
- [Moto-Passion 2018b] MOTO-PASSION: *CRAZY Driver Vs Biker.* https://www.youtube.com/watch?v=HFig7J40mT8&ab_channel=MotoPassion. Version: Dezember 2018. – Aufgerufen am 06.09.2022
- [Moto-Passion 2018c] MOTO-PASSION: *Motorcycle Crashes and Mishaps.* https://www.youtube.com/watch?v=jLm0b4Mq-pQ&ab_channel=MotoPassion. Version: Juli 2018. – Aufgerufen am 06.09.2022
- [Moto-Passion 2018d] MOTO-PASSION: *Motorcycle Crashes on the Road.* https://www.youtube.com/watch?v=S_lizETnMpk&ab_channel=MotoPassion. Version: Juli 2018. – Aufgerufen am 06.09.2022
- [N. Kehtarnavaz 2006] N. KEHTARNAVAZ, C. G.: DSP System Design using LabVIEW and Simulink: A Comparative Evaluation. (2006). <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1660510>
- [Nasa 2019] NASA: *How Does GPS Work?* <https://spaceplace.nasa.gov/gps/en/>. Version: Juni 2019. – Aufgerufen am 01.08.2022
- [NTI-Audio 2019] NTI-AUDIO: *Fast Fourier Transformation FFT - Basics.* <https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft>. Version: Dezember 2019
- [Rogers 2020] ROGERS, K.: *gyroscope.* Encyclopedia Britannica. <https://www.britannica.com/technology/inertial-guidance-system>. Version: Februar 2020. – Aufgerufen am 01.08.2022
- [Sathish 2021] SATHISH: *How to Measure Acceleration in Smartphones using Accelerometer?* <https://blog.contus.com/how-to-measure-acceleration-in-smartphones-using-accelerometer/>. Version: November 2021. – Aufgerufen am 01.08.2022
- [Schnee u. a. 2021] SCHNEE, Jan ; STEGMAIER, Jürgen ; LI, Pu: A probabilistic approach to online classification of bicycle crashes. In: *Accident Analysis and Prevention* 160 (2021), 106311. <http://dx.doi.org/https://doi.org/10.1016/j.aap.2021.106311>. – DOI <https://doi.org/10.1016/j.aap.2021.106311>. – ISSN 0001-4575

[Schnee u. a. 2020] SCHNEE, Jan ; STEGMAIER, Jürgen ; LIPOWSKY, Tobias ; LI, Pu: Auto-Correction of 3D-Orientation of IMUs on Electric Bicycles. In: *Sensors* 20 (2020), Nr. 3, 589. <http://dx.doi.org/10.3390/s20030589>. – DOI 10.3390/s20030589. – ISSN 1424-8220

[Sharma 2020] SHARMA, S.: *What Is Accelerometer? How to Use Accelerometer in Mobile Devices?* <https://www.credencys.com/blog/accelerometer/>. Version: Juli 2020. – Aufgerufen am 01.08.2022

[sparkfun 2022] SPARKFUN: *Gyroscope*. <https://learn.sparkfun.com/tutorials/gyroscope/how-a-gyro-works>. Version: März 2022. – Aufgerufen am 01.08.2022

[Venkatraman u. a. 2021] VENKATRAMAN, V. ; RICHARD, C. M. ; MAGEE, K. ; JOHNSON, K.: *Countermeasures that work: A highway safety countermeasures guide for State Highway Safety Offices*. NHTSA, 2021 <https://www.nhtsa.gov/book/countermeasures/countermeasures-work/motorcycle-safety>. – Aufgerufen am 07.11.2022

[Weisstein 2022] WEISSTEIN, E. W.: *Fast Fourier Transform*. <https://mathworld.wolfram.com/FastFourierTransform.html>. Version: Januar 2022. – Aufgerufen am 25.07.2022

[WHO 2022] WHO: *Road traffic injuries*. <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>. Version: Juni 2022. – Aufgerufen am 01.08.2022

A Anhang

A.1 Entscheidungsfunktion

Screenshot vom Skript mit einer Erklärung (??)

```
1 function [motionClass, lastMotionClassTime, frequency, Inten] =
2     getMotionClass(Intensity, freq, DuSpeed, lastMotionClass,
3         lastMotionClassTime)
4 % status = -1;
5 for iFq = 1:5
6     if Intensity(iFq) > 100
7         lastMotionClassTime = 0;
8         % lastClassTime = 0;
9         if freq(iFq) >= 7
10            % high frequency (most likely driving), check with speed
11            if DuSpeed < 5
12                % high frequency but very low speed => sitting on
13                % motorbike and standing still
14                motionClass = 0;
15            else
16                % there is movement
17                motionClass = 2;
18            end
19            break
20        elseif freq(iFq) >= 0.390625000 && freq(iFq) <= 2
21            % walking, (most likely walking) check with speed
22            if DuSpeed <= 7 && DuSpeed > 0
23                % walking speed
24                motionClass = 1;
25                break
26            elseif DuSpeed == 0
27                % no speed, probably standing
28                motionClass = 0;
29                break
30            else
31                % speed > 7 kmh => not walking => conflict => check next
32                % frequency
33                motionClass = -1;
34                continue;
35            end
36        elseif freq(iFq) > 2 && freq(iFq) < 7
37            % undeclared. Maybe walking and maybe driving. Double check
38            % with the speed. If the speed is bigger than 7 km/h than
39            % he must be driving.
40            % If the speed's smaller than 7 km/h it should stay
```

```

        undeclared , he might be driving with no GPS-Signal (e.g.
        no speed) or walking
    35   if DuSpeed >= 7
        motionClass = 2;
        break;
    37   else
        motionClass = -1;
        continue;
    38 end
    39
    40
    41
    42
    43 elseif freq(iFq) < 0.3906 % undeclared f < 0.3906 Hz
        % not intended to get here
        % (sensor couldn't sense frequencies smaller than 0.396)
        motionClass = -2;
        break
    44
    45
    46
    47
    48 else
        % not intended to get here either ; all usecases were defined
        % in the
        % previous elseifs
        motionClass = lastMotionClass;
        break
    49
    50
    51
    52
    53 end
    54 else
        if DuSpeed >= 7
            motionClass = 2;
            break;
        elseif DuSpeed < 7 && DuSpeed > 0.1
            motionClass = 1;
            break;
        % if lastMotionClassTime <=100
        %     lastMotionClassTime = lastMotionClassTime + 1;
        %     motionClass = lastMotionClass;
        % else
        %     lastMotionClassTime = 0;
        %     motionClass = 1;
        % end
        % continue
        break;
    55
    56
    57
    58
    59
    60
    61
    62
    63
    64
    65
    66
    67
    68
    69
    70
    71
    72
    73
    74 end
    75 end
    76 frequency = freq(iFq);
    77 Inten = Intensity(iFq);
    78 end
    79
    80 % statusAll = [statusAll status];

```

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Master-Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort und Datum: Stuttgart, den _____ Unterschrift: _____