

Universität Stuttgart



**BOSCH**

## Masterarbeit

# Spezialisierung einer Unfallerkennung am Zweirad mittels Smartphone auf Unabhängigkeit der Trageposition

**eingereicht von:** Cand. M.Sc. Oays Darwish  
Matrikelnr.: 3480821

**am:** Institut für Mikrointegration  
Prof. Dr.-Ing. André Zimmermann

**Prüfer:** Herr Dr. Thomas Günther  
**Betreuer:** Herr Dipl.-Ing. Nino Häberlen

**Ausgabedatum:** 15.05.2022  
**Abgabedatum:** 15.02.2023





## 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](http://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](mailto:Nino.Haeberlen@bosch.io)



# **Eigenständigkeitserklärung**

Ich erkläre mit meiner Unterschrift, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen dieser Arbeit, die dem Wortlaut, dem Sinn oder der Argumentation nach anderen Werken entnommen sind (einschließlich das World Wide Web und anderer elektronischer Text- und Datensammlungen), habe ich unter Angabe der Quellen vollständig kenntlich gemacht.

Ort und Datum: Stuttgart, den \_\_\_\_\_ Unterschrift: \_\_\_\_\_



# **Kurzzusammenfassung**

Diese Abschlussarbeit beschäftigt sich mit der Weiterentwicklung eines Verfahrens zur smartphonebasierten Unfallerkennung am Zweirad im Taschenmodus (Pocket-Mode). Zu Beginn wird die Relevanz dieser Unfallerkennung mit aktuellen Unfallzahlen erläutert und der bisherige Unfallerkennungsalgorithmus vorgestellt. Danach wird eine Liste der Anwendungs- und potenzielle Grenzfälle anhand mehrerer Videoaufnahmen von Motorradfahrten und -Unfälle vorbereitet. Im Anschluss werden einzelne kritische Szenarien wie z.B. Laufen und während einer Fahrt Anhalten näher betrachtet. Im Rahmen dieser Arbeit wird ebenfalls eine Lauferkennung mittels Matlab/Simulink implementiert. Diese wird durch eine Verarbeitung der Beschleunigungssignale aus dem Smartphone sowie die Frequenzermittlung dieses Signals mit einer FFT und danach mit einer Entscheidungsfunktion erfolgen, die die ermittelte Frequenz sowie die gemessene Geschwindigkeit zum Entscheiden über den Laufen/Fahren-Status verwendet. Im Sinne der Lauferkennungsverifizierung und zur weiteren Betrachtung anderen Szenarien werden mehrere Versuche geplant, durchgeführt und ausgewertet. Bei der qualitativen Auswertung der Lauferkennung werden zwischen mehreren Methoden wie der Ausgabe der implementierten Aktivitätserkennung sowie der im Smartphone integrierten Aktivitätserkennung verglichen.

# **Abstract**

This master-thesis deals with the further development of a method for smartphone-based accident detection on two-wheelers in pocket mode. The relevance of this accident detection is first explained with statistical numbers and the previous accident detection algorithm is presented. A list of use cases and potential edge cases is prepared based on several videos of motorcycle rides and accidents. Subsequently, some individual critical scenarios such as walking and stopping are analyzed in more detail. In the context of this work, a walk detection module is implemented. This will be done by processing the acceleration signals from the smartphone and determining the frequency of this signal with an FFT and then with a decision script that uses the determined frequency and the measured speed to decide between walking/driving state. Several tests are planned, carried out and evaluated for the purpose of verifying the walking detection and for further consideration of other scenarios. In the evaluation of the walking detection, several methods such as the implemented activity detection and the activity detection integrated in the smartphone are compared.



# **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, Februar 2023

Oays Darwish



# Inhaltsverzeichnis

<b>Eigenständigkeitserklärung</b>	<b>v</b>
<b>Kurzzusammenfassung</b>	<b>vii</b>
<b>Abstract</b>	<b>vii</b>
<b>Danksagung</b>	<b>ix</b>
<b>Abbildungsverzeichnis</b>	<b>xiii</b>
<b>Tabellenverzeichnis</b>	<b>xv</b>
<b>Abkürzungsverzeichnis</b>	<b>xvii</b>
<b>Symbolverzeichnis</b>	<b>xix</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Grundlagen</b>	<b>3</b>
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 Android-integrierte Aktivitätserkennung . . . . .	14
2.2.3 Matlab/Simulink . . . . .	14
2.3 Mathematische Grundlagen . . . . .	16
2.3.1 Fast Fourier Transformation (FFT) . . . . .	16
2.4 Agile Softwareentwicklung . . . . .	18
2.5 Unfallerkennungsalgorismus . . . . .	18
2.5.1 Kalibrierung . . . . .	19
2.5.2 Übersicht der bereits erkennbaren Unfälle . . . . .	20
2.5.3 1. Feature: TipOver . . . . .	22
2.5.4 2. Feature: GroundHit . . . . .	22
2.5.5 3. Feature: Collision . . . . .	25

<b>3 Unfallerkennung im Pocket-Mode</b>	<b>27</b>
3.1 Begründung für den Pocket-Mode . . . . .	27
3.2 Kritische Szenarien . . . . .	28
3.3 Lauferkennung . . . . .	31
3.3.1 Spitzenzähler . . . . .	32
3.3.2 Frequenzbasierte Lauferkennung . . . . .	33
3.4 Auf- und Absteigen . . . . .	45
3.5 Anhalten . . . . .	45
3.6 Verifikation des Algorithmus . . . . .	46
3.6.1 Versuchsplanung . . . . .	46
3.6.2 Referenz-Aktivitätsdaten (Ground truth) . . . . .	50
<b>4 Ergebnisse</b>	<b>53</b>
4.1 Verifizierungsversuche . . . . .	53
4.2 Lauferkennung . . . . .	53
4.3 Verschiedene Fahrerpositionierung . . . . .	54
4.4 Anhalten . . . . .	56
<b>5 Ausblick</b>	<b>59</b>
<b>Literaturverzeichnis</b>	<b>xxi</b>
<b>A Anhang</b>	<b>xxv</b>
A.1 Matlab-Entscheidungsfunktion . . . . .	xxv

# Abbildungsverzeichnis

1.1	Umgang der Funktion „Help Connect“ im Fall eines Motorradunfalls . . . . .	2
2.1	Einfache Darstellung des Regelkreises ‚Fahrer-Fahrzeug-Umfeld‘[H. Appel, 2002] . . . . .	3
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 GAIDAS-Datenbank [d. Maire, 2020] . . . . .	6
2.5	Anzahl der Unfälle gegenüber Unfallgegner . . . . .	7
2.6	Beispiele verschiedener Unfallarten (Alleinunfall, Unfall mit einem Gegenstand sowie mit einem Motorrad) im Straßenverkehr . . . . .	8
2.7	Anzahl der Unfälle über der Art der Ursache . . . . .	9
2.8	Anteil der aktiven sowie passiven Unfälle . . . . .	9
2.9	Vereinfachte Darstellung der Kräfte bei stationärer Kurvenfahrt [Hädrich, 2012] . . . . .	11
2.10	Verschiedene Techniken bei Kurvenfahrt mit gleicher Geschwindigkeit [Hädrich, 2012] . . . . .	12
2.11	Beispiel von einer FFT (Zeitbereich und Frequenzbereich)[NTI-Audio, 2019] . . . . .	17
2.12	Achsenrichtung in Sensorframe sowie in Bikeframe . . . . .	19
2.13	Entscheidungsbaum des Unfallerkennungsalgorithmus anhand der drei Features sowie des Fahrzustands[Schnee u. a., 2021] . . . . .	21
2.14	Beispiel eines Fahrrads beim Umkippen mit einer ursprünglichen aufrechten Postion sowie mit einer ursprünglichen Neigung. Die Beispiele gelten ebenfalls für ein Motorrad . . . . .	23
2.15	Verlauf der Energie sowie des Energieschwellwerts bei einer Testfahrt ohne GroundHit . . . . .	24
2.16	Verlauf der Energie sowie des Energieschwellwerts bei einer Echtfahrt mit GroundHit . . . . .	24
2.17	Verlauf der Energie sowie des Energieschwellwerts bei einer echten Fahrt ohne Kollision . . . . .	25
2.18	Verlauf der Energie sowie des Energieschwellwerts bei einer echten Fahrt mit einer Kollision . . . . .	26
3.1	Ergebnisse der Umfrage vom Spiegelinstitute über das Taschenmodus . . . . .	28

3.2	Beispiel eines Beschleunigungssignal beim Laufen . . . . .	31
3.3	Testmodell der Lauferkennung - Spitzenzähler . . . . .	32
3.4	Darstellung des im Testmodell der Lauferkennung generierten Sinussignal sowie jede Überschneidung der x-Achse . . . . .	33
3.5	Skizze eines einfachen Signals sowie eines komplexeren Signals . . . . .	34
3.6	Testbeispiel - Frequenzbasierte Lauferkennung - Sinussignal . . . . .	35
3.7	verschiedene Einzelsignale ( $f = 23,8; f = 47,8; f = 1,1$ ) mit deren Summe $f_g$ . . . . .	36
3.8	Testbeispiel - frequenzbasierte Lauferkennung - Ausgabe des Spektrum-Analyzers im Fall eines komplexen Signals . . . . .	37
3.9	Ablaufschema des Testmodells der frequenzbasierten Lauferkennung . .	38
3.10	Testbeispiel - Frequenzbasierte Lauferkennung - FFT . . . . .	39
3.11	Das Ergebnis der FFT - Nach Entfernung der Spiegelung und Ermittlung der Beträge . . . . .	40
3.12	Das vollständige Echtmodell der Frequenzermittlung im Rahmen der Lauferkennung . . . . .	41
3.13	Ein- und Ausgänge des Entscheidungsskripts . . . . .	42
3.14	Entscheidungsbaum der Aktivitätserkennung . . . . .	42
3.15	Abgeschnittene Teile eines Beispieldesignals . . . . .	44
3.16	Winkeländerung im Signal zwischen vertikaler sowie horizontaler Smartphone-Positionierung . . . . .	46
3.17	Die Versuchsschritte des ersten Versuchsszenarios, indem die Versuchsperson sitzend und stehend fährt, absteigt, läuft und aufsteigen . . . .	49
3.18	Beispiel der exportierten Tabelle mit der neuen Spalte . . . . .	51
4.1	Ergebnis des Lauferkennungsmodells . . . . .	54
4.2	Zeitlicher Verlauf der Geschwindigkeit, Ground Truth der Fahreraktivität und -Position sowie der Winkeländerung . . . . .	55
4.3	Winkeländerung des Smartphones durch das Anhalten . . . . .	56
4.4	Beinpositionen während einer Fahrt und beim Anhalten . . . . .	57

# Tabellenverzeichnis

2.1	Ausgaben der Google-Aktivitätserkennung[Elbayoumy, 2018] . . . . .	14
2.2	Erklärung der verschiedenen Features-Zustände im Entscheidungsbaum in der Abbildung 2.13[Schnee u. a., 2021] . . . . .	20
3.1	Die Use- und Edgecases mit der erwarteten Reaktion des Algorithmus .	29
3.2	Ausgangsmöglichkeiten der Entscheidungsfunktion . . . . .	42
3.3	Die Zusammenfassung der getesteten Szenarien . . . . .	47
3.4	Die definierte Labels von Groundtruth . . . . .	51
4.1	Beschreibung der Klassen in der Abbildung 4.1 . . . . .	54



# **Abkürzungsverzeichnis**

API	Application Programming Interface
Anova	Analysis of Variance
BD	Blockdiagramm (LabVIEW)
BF	Bike Frame
DFT	Diskrete-Fourier-Transformation
DOF	Degrees of Freedom
DSP	Digitaler Signalprozessor
FFT	Fast-Fourier-Transformation
FP	Frontalpanel (LabVIEW)
GIDAS	German In-Depth Accident Study
GPS	Global Positioning System
IMU	Inertial Measurement Unit
MEMS	Mikro-Elektronisch-Mechanische Systeme
NHTSA	National Highway Traffic Safety Administration
SF	Sensor Frame
SPF	Schwerpunkt des Fahrers
SPM	Schwerpunkt der Maschine
VI	Virtuelle Instrumente
VSS	Verkehrssicherheitsscreening



# **Symboverzeichnis**

$B_l$	Blocklänge
$D$	Messdauer
$d_f$	Frequenzauflösung
$F_G$	Gewichtskraft
$f_n$	Bandbreite
$F_q$	Zentrifugalkraft
$f_s$	Abtastfrequenz
$\lambda$	Neigungswinkel



# 1 Einleitung

Jedes Jahr wird weltweit das Leben von etwa 1,3 Millionen Menschen durch einen Verkehrsunfall beendet. Zwischen 20 Millionen und 50 Millionen weitere Menschen erleiden 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 von Verkehrsteilnehmern dar. Die National Highway Traffic Safety Administration (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 und 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. Bosch hat ein neues Notrufsystem namens „Help Connect“ auf den Markt gebracht, das automatisch Hilfe rufen kann, wenn es Unfälle auf Motorrädern erkennt. Der Dienst nutzt einen Unfallerkennungsalgorithmus, um festzustellen, ob ein Motorrad wirklich in einen Unfall verwickelt oder nur umgestürzt ist. Wenn „Help Connect“ entscheidet, dass ein Motorrad einen Unfall hatte, übermittelt es Informationen über die Unfallstelle und den Fahrer wie z.B. den aktuelle Standort, den Schweregrad des Aufpralls und die optional hinterlegte Gesundheitsdaten an das Bosch Service Center. Im Call-Center versuchen speziell geschulte Agenten, den Fahrer zu erreichen. Wenn der Fahrer nicht reagiert und die Sensoren einen schweren Unfall erkannt haben, wenden sie sich mit allen Informationen an die lokale Rettungsdienste, damit sie der Person so schnell wie

möglich helfen können.[Gibbons, 2021][Moon, 2020][Bosch, 2021]

Die Abbildung 1.1 fasst die Ablaufschritte im Fall einer Unfallerkennung zusammen. Nachdem ein Motorradunfall durch das Smartphone erkannt wird, wird eine Benachrichtigung an das Bosch Call-Center mit der Angabe der Unfallschwere weitergeleitet. Der Agent versucht den Fahrer telefonisch zu erreichen, der den Unfall bestätigt oder dementiert. Wenn die Person Hilfe benötigt, gibt der Agent die notwendige Informationen an die lokale Rettungsdienste weiter.

Diese Arbeit beschäftigt sich mit der Weiterentwicklung des Unfallerkennungsalgorithmus, damit dieser auch im Taschenmodus verwendbar und funktionsfähig bleibt. Darunter fällt die Ermittlung der potenziellen Verhaltensunterschiede des Algorithmus zwischen dem originalen Modus (Am Lenker) und dem Taschenmodus. Anpassungs-ideen werden vorgeschlagen und bei bestimmten Szenarien entwickelt und verifiziert.

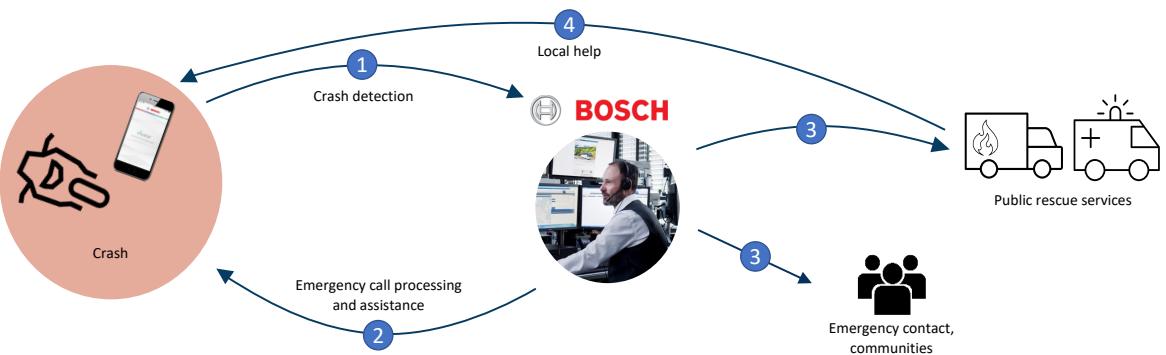


Abbildung 1.1: *Umgang der Funktion „Help Connect“ im Fall eines Motorradunfalls*

# 2 Grundlagen

In diesem Kapitel wird die Relevanz der vorliegenden Arbeit erörtert und wichtige Begriffe wie „Unfall“ näher erklärt.

## 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]

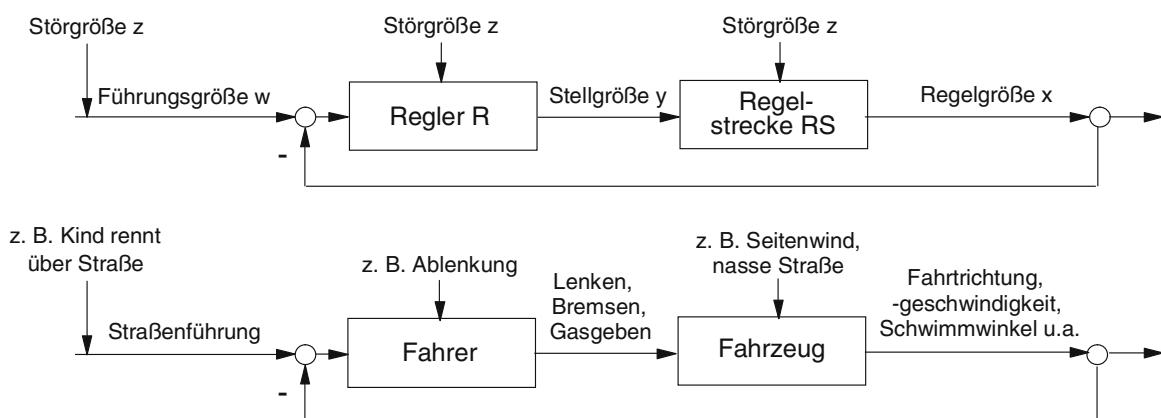


Abbildung 2.1: Einfache Darstellung des Regelkreises „Fahrer-Fahrzeug-Umfeld“ [H. Appel, 2002]

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 Trennung. Bei Mehrfachkollisionen werden mehrere Kollisionsphasen auftreten.
- Post-Crash-Phase (Folgephase): Die Folgephase ist der Zeitraum vom Trennen 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 des Fahrzeugs 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 verschiedensten 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 Unfallsituation 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 einer „Fahrer-Fahrzeug-Interaktion“ führt, die zu Unfällen führen kann.[H. Appel, 2002]

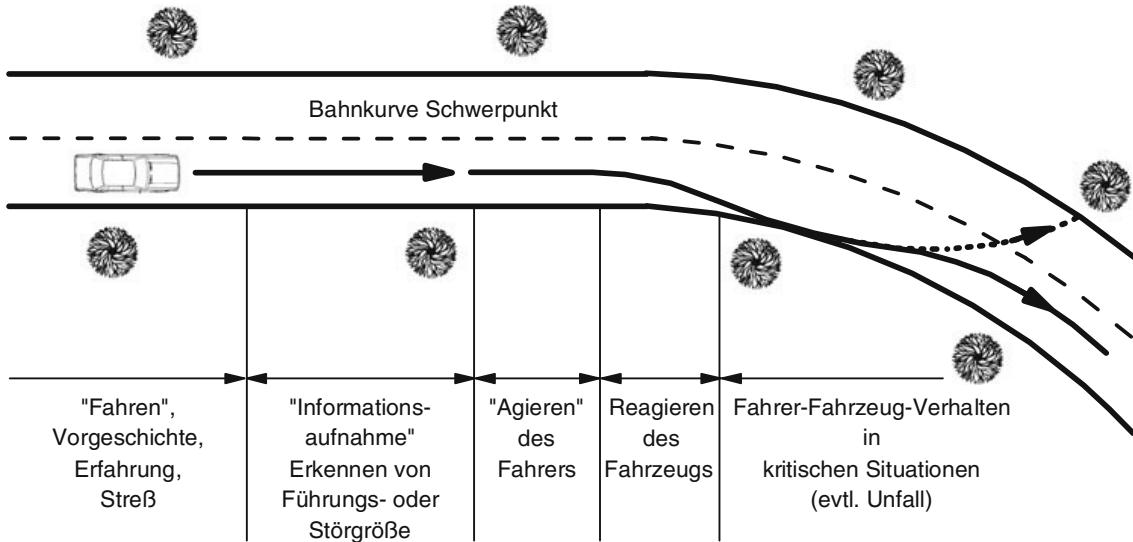


Abbildung 2.2: Beispiel der zeitlichen Phasen einer kritischen Fahrsituation [H. Appel, 2002]

### 2.1.3 Statistische Zahlen über Motorradunfälle

Das baden-württembergische Verkehrsministerium stellt ein Portal zur Verfügung, über das einzelne Verkehrsmessstellen abgefragt werden können. Das Verkehrsverhalten wurde an mehreren ausgewählten Stationen betrachtet. 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 ausgewählten Stationen das Geschwindigkeitslimit regelmäßig überschritten wird. Die Motorradfahrer missachten die Geschwindigkeitsbegrenzung häufiger als die PKW-Fahrer. [d. Maire, 2020]

Die Daten aus der German In-Depth Accident Study (GIDAS) enthalten 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 Geschwindigkeit 0 km/h gibt es keine tödlichen Unfälle, wobei die Unfälle bei Einer von über 100 km/h fast immer mit schwer verletzten oder toten Verkehrsteilnehmern enden. Die hohe Geschwindigkeit kommt immer mit hohen Kräften zusammen, welche bei einem Unfall auf den Fahrer einwirken. Im Fall eines Motorradfahrers ist das

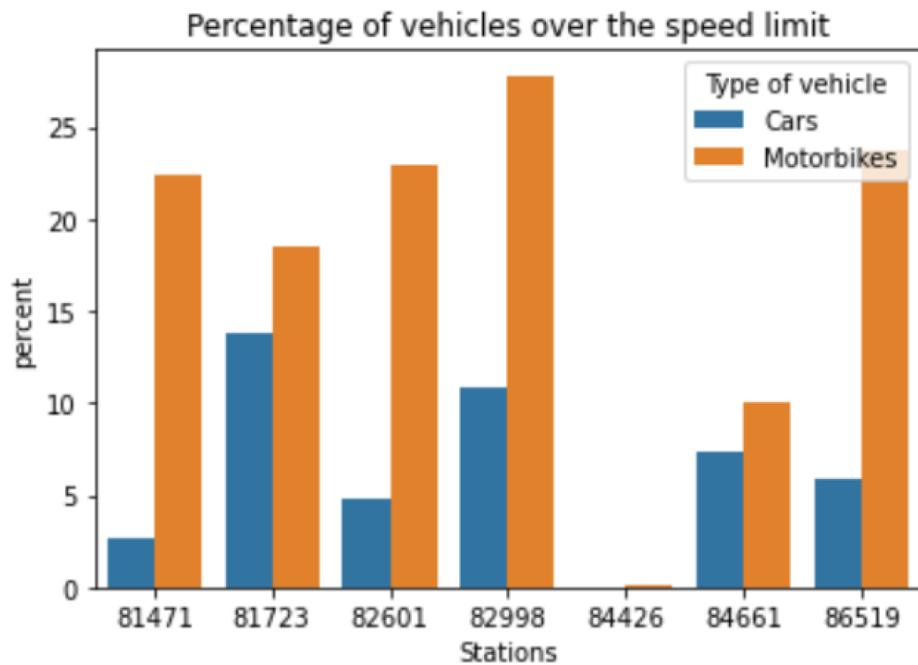


Abbildung 2.3: Geschwindigkeitsüberschreitungen nach PKW- und Motorradfahrer an sieben Messstationen [d. Maire, 2020]

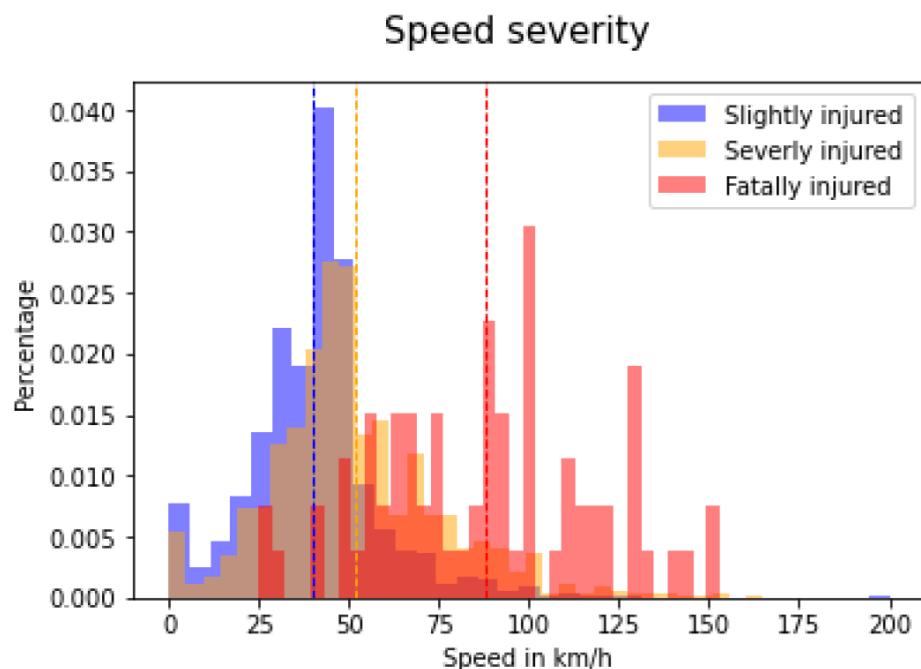


Abbildung 2.4: Unfallschwere über Geschwindigkeit aus der GAIDAS-Datenbank [d. Maire, 2020]

als besonders wichtig zu betrachten, da diese Kräfte auf den Fahrer direkt übertragen werden. [d. Maire, 2020]

### **Interne Statistik aus mehreren Videoaufnahmen**

Im Sinne der Verifizierung des angepassten Unfallerkennungsalgorithmus muss erstmals bekannt sein, welche Unfallarten am häufigsten vorkommen, damit diese vertieft betrachtet werden. Dazu wurden mehrere Videos von Motorradunfällen auf dem Plattform „YouTube“ stichpunktartig angesehen und die vorgestellten Unfälle statistisch analysiert. Es wurden insgesamt 32 Unfallsituationen ausgewertet. In der Auswertung wurden die Unfallgegner und der Ablauf des Unfalls betrachtet. [Moto-Passion, 2018]

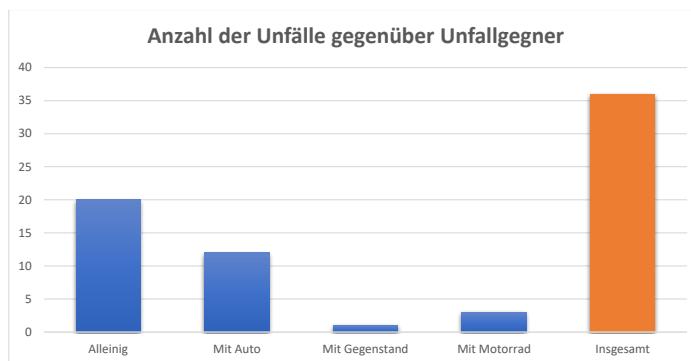
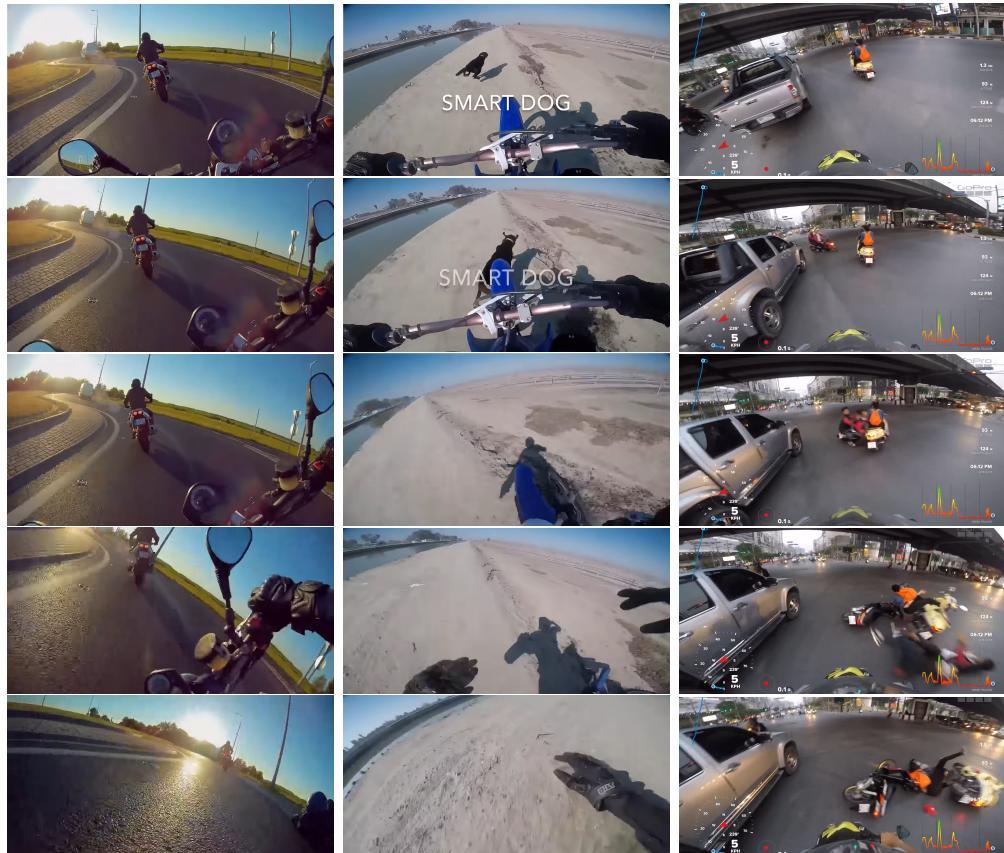


Abbildung 2.5: Anzahl der Unfälle gegenüber Unfallgegner

In der Abbildung 2.5 stellt die Grafik die Anzahl der Motorradunfälle gegenüber dem Unfallgegner dar. Es wurde hier zwischen Alleinunfall, Unfall mit einem Gegner (Auto, Motorrad) oder Unfall wegen eines Gegenstands aufgrund des unterschiedlichen Verhaltens während eines Unfalls unterschieden. Von insgesamt 36 Unfälle waren die Alleinunfälle am häufigsten gefolgt von den Unfällen mit einem Auto. Die Unfälle mit einem anderen Motorrad oder wegen eines Hindernisses sind am seltensten.

Die Abbildung 2.6 zeigt ein paar Unfallbeispiele aus den YouTube-Videos mit verschiedener Unfallarten (Alleinunfall, Unfall mit einem anderen Verkehrsgegner sowie Unfall mit einem Gegenstand). In der Abbildung 2.6a ist das Beispiel eines Alleinunfalls in einer Kurve, indem der Fahrer während einer Kurvenfahrt rutschte. Die Bilderserie zeigt mit paar Screenshots den Sturzablauf des Unfalls. Die Abbildung 2.6b stellt einen Unfall mit einem Gegenstand (in diesem Fall mit einem Hund) dar. Der Fahrer war alleine auf einer Landstraße unterwegs und ein Straßenkötter kam ihm plötzlich in die Quere. Der Fahrer hatte keine Zeit den Kontakt mit dem Hund zu vermeiden und ist gestürzt. Die Abbildung 2.6c zeigt ein Beispiel eines Unfall mit einem Gegner. In diesem Fall geschah der Verkehrsunfall mit einem anderen Motorrad an einer Ampel. Die vorgestellte Beispiele zeigen, wie die Motorradfahrer zu der gefährdeten Gruppe gehören. Unter Alleinunfällen treten zwei Szenarien (in einer Kurve rutschen und Kontrolle verlieren) am häufigsten auf, was sehr gut in der Abbildung 2.7 sichtbar ist. Das Szenario, in dem das Motorrad auf ein Auto auffährt, hatte die dritte Stelle



(a) Beispiel eines Alleinunfalls in einer Kurve (b) Beispiel eines Unfalls mit einem Gegenstand (c) Beispiel eines Unfalls mit einem Motorrad

Abbildung 2.6: Beispiele verschiedener Unfallarten (Alleinunfall, Unfall mit einem Gegenstand sowie mit einem Motorrad) im Straßenverkehr

besetzt. Die Hauptgründe der Unfälle mit einem Gegner waren vor allem die hohe Geschwindigkeit und das schlechte Wetter (z.B. Nässe, Schnee), was zu Schwierigkeiten geführt hat, die kritische Situation rechtzeitig zu erkennen und das Motorrad gut zu kontrollieren.

Die Abbildung 2.8 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 zum anderen Verkehrsteilnehmer hält.

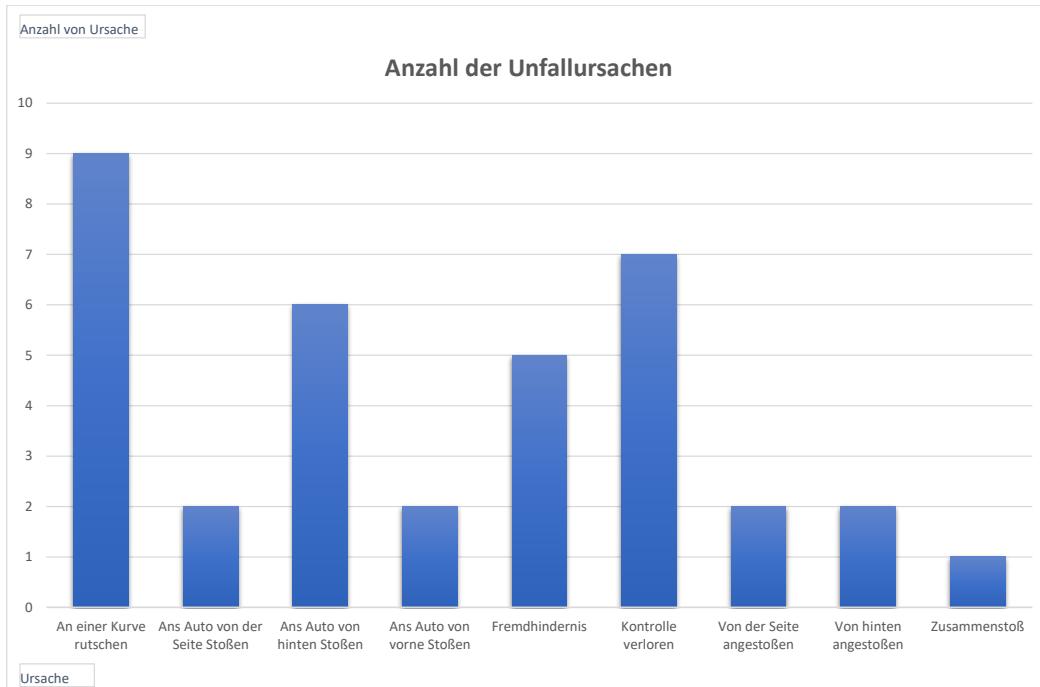


Abbildung 2.7: Anzahl der Unfälle über der Art der Ursache

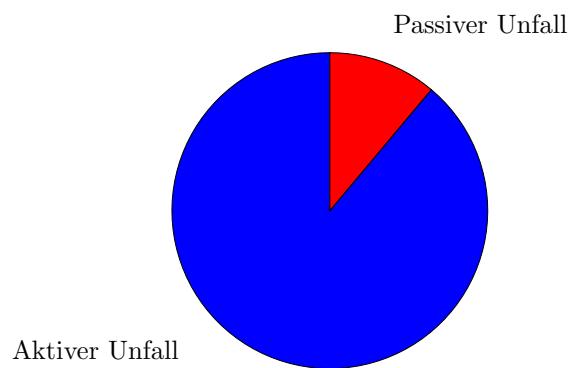


Abbildung 2.8: Anteil der aktiven sowie passiven Unfälle

### 2.1.4 Mechanik der Motorradfahrt

Aus kinematischer Sicht besteht jedes mechanische System aus einer Anzahl starrer Körper, die durch eine Anzahl Gelenke miteinander verbunden sind. Jeder Körper hat sechs Freiheitsgrade (DOF), da seine Position und Orientierung im Raum vollständig durch sechs Parameter definiert sind, wie z.B. die drei Koordinaten eines Punktes (x, y, z) und drei Winkel (yaw, roll, pitch). In der Praxis hat der Fahrer beim Fahren seine Hände am Lenker. Dies kann die Stabilität des Zweirads aufgrund der Schleife verändern, die zwischen dem Körper des Fahrers, dem hinteren Rahmen (wo der Fahrer sitzt) und dem vorderen Rahmen (wo sich die Hände des Fahrers befinden) entsteht.[Cossalter u. a., 2014]

Das Verhalten des Fahrers während der Fahrt hat einen starken Einfluss auf die Mechanik des Motorrads. Zunächst wird die Kurvenfahrt näher betrachtet.

#### Kurvenfahrt

Der Fahrer versucht während der Fahrt immer das Gleichgewicht zu halten, damit er und das Motorrad nicht umkippt. Befährt ein Motorradfahrer eine Kurve, tritt im Vergleich zu einer Geradeausfahrt eine Instabilität im Gleichgewicht auf und der Fahrer versucht die Maschine wieder ins 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 2.9 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 in Schräglage durchfahren. Dadurch verschiebt sich der Schwerpunkt der Einheit zur Kurveninnenseite und erzeugt ein Moment der Gewichtskraft um den Reifenaufstandspunkt, welches dem Moment der Fliehkraft entgegen wirkt und wieder ein Gleichgewicht sichert.

Der Schrägwinkel ( $\lambda$ ) der Einheit lässt sich wie folgt berechnen:

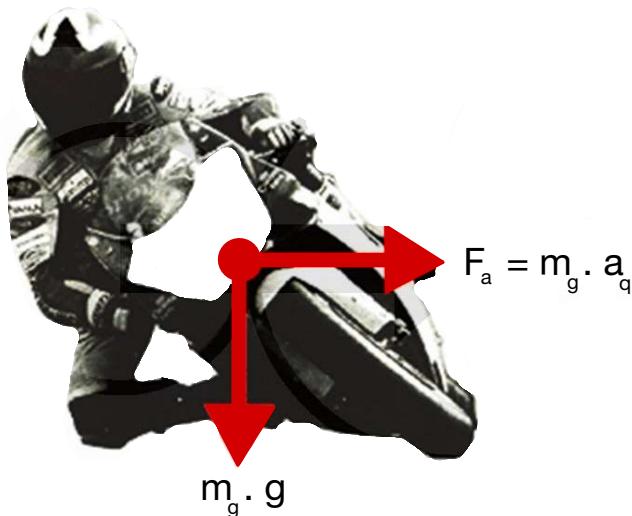


Abbildung 2.9: Vereinfachte Darstellung der Kräfte bei stationärer Kurvenfahrt [Hädrich, 2012]

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

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

## 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 (Abbildung 2.10).

Der Fahrstil „Drücken“ hat im Vergleich zum Fahrstil „Legen“ keinen Vorteil im Zusammenhang mit dem Grip 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 2.10). Gegenüber der anderen zwei Schräglagen können durch das „Hängen“ mit gleichen Schrägwinkel höhere Seitenkräfte übertragen werden, da hier die Radaufstandsfläche größer ist.

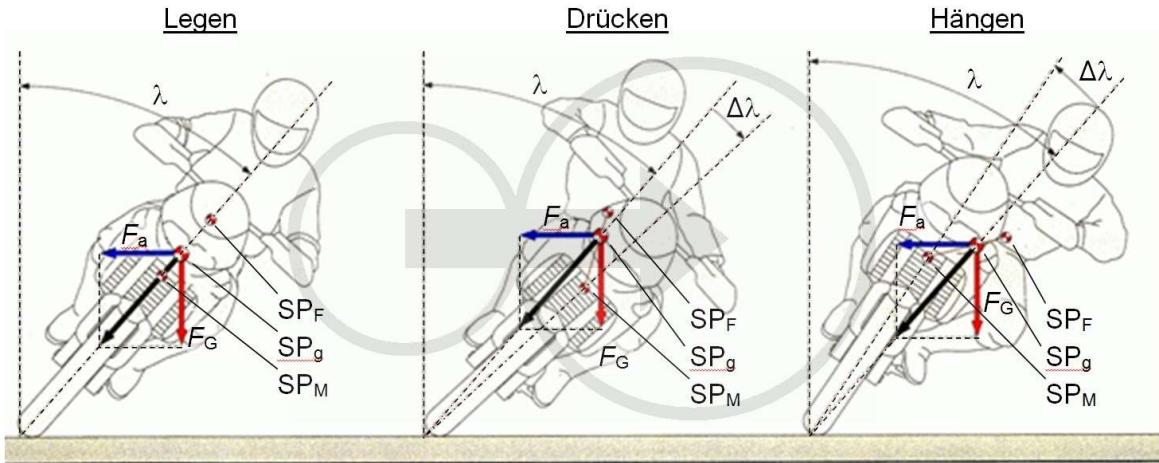


Abbildung 2.10: Verschiedene Techniken bei Kurvenfahrt mit gleicher Geschwindigkeit [Hädrich, 2012]

## 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 Bewegung des Geräts spiegelt normalerweise die direkte Benutzereingabe wider, kann aber auch die physische Umgebung widerspiegeln, in der sich das Gerät befindet. Das Smartphone bewegt sich mit der Person, die es am Körper hat und sich selbst bewegt.[Developers, 2022]

#### Beschleunigungssensoren

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-Beschleunigungswerte, die zum Messen der Position und der Beschleunigung des Geräts verwendet werden. Die XYZ-Kombination stellt die Richtung und Position des Geräts dar, an dem eine

Beschleunigung aufgetreten ist. Die vom Gerät bereitgestellten Beschleunigungsmesswerte 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  $\text{Am/s}^2$  in den Himmel geschoben wurde, ist der Wert der z-Beschleunigung gleich  $A+9,81 \text{ m/s}^2$ , 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

Ein 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 Gyrosensor in Mikro Elektronisch Mechanischen Systeme (MEMS) ist winzig (zwischen  $1 \mu\text{m}$  und  $100 \mu\text{m}$ ). 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]. Viele Smartphones haben einen Gyroskop-Sensor verbaut, der es ermöglicht, die lineare Ausrichtung des Handys sehr genau zu bestimmen. Apps können das Gyroskop nutzen, um durch Kippen oder Neigen des Handys beispielsweise ein Spiel zu steuern.[Schanze, 2017]

## Global Positioning System (GPS)

Das 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 tatsächlich dort sind, wo sie sich befinden sollen. Ein Empfänger in dem Smartphone wartet ständig auf ein Signal von diesen Satelliten und findet heraus, wie weit er von mehreren Satelliten entfernt ist. Sobald die Entfernung zwischen einem Empfänger und vier oder mehr Satelliten berechnet wird, ist genau bekannt, wo der Empfänger sich befindet. Der Basis-GPS-Dienst bietet Benutzern eine Genauigkeit von etwa 7 Metern, in 95% der Zeit. GPS-Empfänger zeigen die Position an und berechnen die Geschwindigkeit mithilfe von Algorithmen im Kalman-Filter.[Nasa, 2019][FAA, 2021][Deiss, 1999]

### 2.2.2 Android-integrierte Aktivitätserkennung

Die menschliche Aktivitätserkennung mit kinematischen Sensoren ist einer der weit verbreiteten Forschungsbereiche auf der Basis von Smartphones. Die Entwicklung in der Sensornetzwerktechnologie war die Geburtsstunde der Anwendungen, die intelligente und freundschaftliche Dienste basierend auf der Aktivitätserkennung von Menschen anbieten können. Smartphones sind zu einer wichtigen Technologie geworden und mit leistungsstarken und vielfältigen Sensoren ausgestattet, die zur Aktivitätsüberwachung verwendet werden. [Rasheed u. a., 2015]

In Android-Smartphones hat Google eine Reihe von Application Programming Interfaces (APIs) eingeführt, die es den Entwickler ermöglichen, die Ergebnisse der Erkennung menschlicher Aktivitäten zu erhalten. Dieses API verwendet die eingebauten Sensoren, um zwischen 8 Benutzeraktivitäten zu unterscheiden. Darunter werden „in vehicle“, „on bicycle“, „on foot“, „running“, „still“ und „walking“ erkannt. [Tran u. Phan, 2016][Elbayoumy, 2018]

Die Tabelle 2.1 zeigt einen Ausgabenübersicht der Android-Aktivitätserkennung.

Tabelle 2.1: *Ausgaben der Google-Aktivitätserkennung[Elbayoumy, 2018]*

Activity Name	Activity Description
IN VEHICLE	The device is in a vehicle, such as a car.
ON BICYCLE	The device is on a bicycle.
ON FOOT	The device is on a user who is walking or running.
RUNNING	The device is on a user who is running.
STILL	The device is still (not moving).
TLTING	The device angle relative to gravity changed significantly.
UNKNOWN	Unable to detect the current activity.
WALKING	The device is on a user who is walking.

### 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 am 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 Anzeigeblocks können die Simulationsergebnisse während der Simulation analysiert werden. Die Simulationsergebnisse können zur Nachbearbeitung und Visualisierung in den Matlab-Arbeitsbereich gestellt werden. [Iov u. a., 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 Entwerfen 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 Simulationsysteme 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 nicht der Fall ist.

### **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 hohe Geschwindigkeit und eine weitaus bessere Möglichkeit zur Speicherverwaltung.[AWS, 2022][Joy, 2021]

Für die Generierung eines C-Codes aus Simulink-Modele wird der integrierte C-Coder verwendet.

### C/C++-Coder

Der C/C++-Code-Generator wird für Rapid Prototyping, Simulationsbeschleunigung oder einfach als 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.[Mathworks, 2021]

Der C-Coder ist ein weiterer Vorteil von Simulink gegenüber LabVIEW und daher ist Simulink die richtige Wahl für die Implementierung der Softwares beziehungsweise Algorithmen, die später in Smartphones verwendet werden.

## 2.3 Mathematische Grundlagen

In dieser Arbeit wird eine Fast-Fourier-Transformation (FFT) eingesetzt, für welche das folgende Hintergrundwissen zum Verständnis benötigt wird.

### 2.3.1 Fast Fourier Transformation (FFT)

Die „Fast Fourier Transformation“ ist ein wichtiges Messverfahren und wurde erstmals von Cooley und Tukey 1965 diskutiert. 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 2.11 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.

### Schritt für Schritt

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

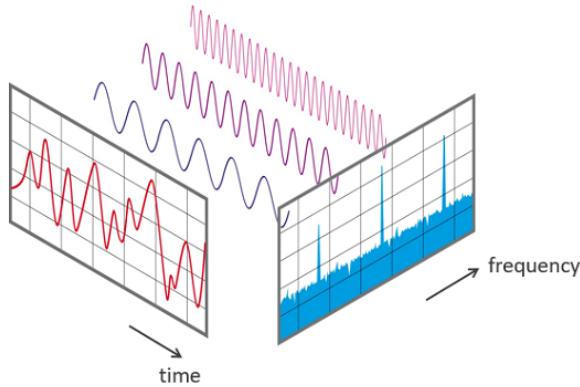


Abbildung 2.11: Beispiel von einer FFT (Zeitbereich und Frequenzbereich)[NTI-Audio, 2019]

1. Die Abtastrate  $f_s$  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 eine FFT bestimmt werden kann und wird wie folgt berechnet:

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

Beispielsweise können bei einer Abtastrate von 100 Hz Frequenzanteile bis maximal 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 und lässt sich mit

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

angeben.

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 und 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. Diesen 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. Bei einer agilen Softwareentwicklung werden die Entwicklungsphasen in mehrere Sprints eingeteilt. Die Länge eines Sprints wird am Anfang festgestellt. Nach jedem Sprint wird das Ergebnis ausgewertet. Dieses wird ggf. für die Anpassung des Entwicklungsvorgehens im nachfolgenden Sprints benutzt. [Brunskill, 2019]

## 2.5 Unfallerkennungsalgorithmus

In diesem Teil wird der bisherige Unfallerkennungsalgorithmus erläutert und näher betrachtet. Der Algorithmus wurde sowohl für Fahrräder als auch für Motorräder entwickelt und verarbeitet die Signale von Beschleunigungssensoren sowie Gyroskopen im Smartphone und die über GPS gemessene Geschwindigkeit. In dem Unfallerkennungsalgorithmus sind drei Hauptfeatures (Collision, GroundHit und TipOver) eingebaut. 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 usw.) über verschiedene Fenstergrößen der analysierten IMU-Beschleunigungsdaten zu wissen. Der Anova-Ansatz erfordert, dass die vorverarbeiteten Daten normalverteilt sind. 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:

$$\Delta_{xy,u}(i) = \left( \int_{i-u}^i a(k) dk \right)^2 + \left( \int_{i-u}^i a(k) dk \right)^2 \quad (2.5)$$

$\Delta e_{xy,u}$  stellt die Änderung der massenspezifischen kinetischen Energie dar. Es beschreibt ein Ereignis in der XY-Ebene im Zweiradkoordinatensystem (also dem Zweiradrahmen (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 Zweiradsystems zurückgeführt. Zusätzlich bietet die Varianz der x- und y-Beschleunigungssignale über ein Zeitfenster weitere Trennmöglichkeiten für GroundHit-Ereignisse. [Schnee u. a., 2021]

Die Entwicklung des Unfallerkennungsalgorithmus sowie des Pocket-Mode wird mittels der agilen Methode durchgeführt. Der implementierte Algorithmus enthält einen verbreiteten und komplexen Entscheidungsbaum, welcher sich mit Simulink sowohl übersichtlicher als auch einfacher darstellen lässt als mit LabVIEW. 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.

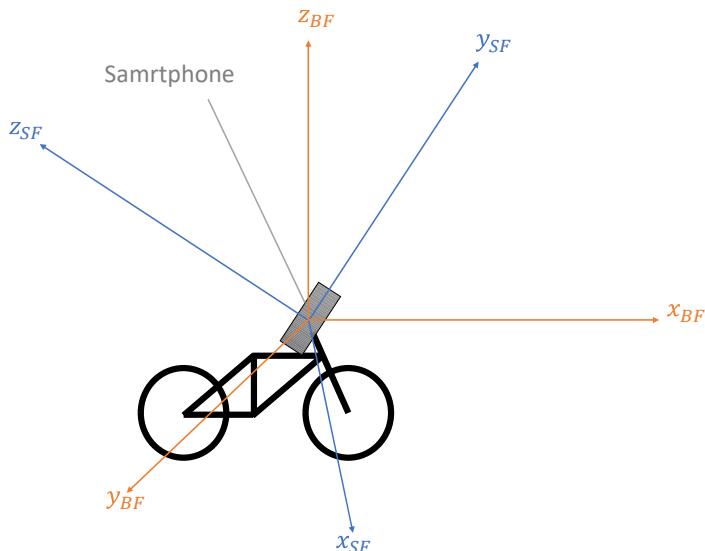


Abbildung 2.12: Achsenrichtung in Sensorframe sowie in Bikeframe

### 2.5.1 Kalibrierung

Die Kalibrierung dient dazu, die Ausrichtung des Motorrads zu erkennen, damit die Richtung der Fahrt sowie diesbezügliche Bewegungen (Beschleunigung, Bremsen, Neigung, ...usw.) richtig erkannt und gut ausgewertet werden.

In der Abbildung 2.12 ist der Unterschied zwischen der originalen Achsenrichtung (vom Smartphone) und der des Objekts anhand eines Fahrradbeispiels abgebildet. In der Abbildung sind die Achsen des Smartphones mit den Buchstaben „SF“ (Sensor Frame) gekennzeichnet, sowie diese des Fahrrads mit den Buchstaben „BF“ (Bike Frame).

Während der Kalibrierung wird die Position des Fahrrads beziehungsweise Motorrads sowie die Richtung der Fahrt erkannt und die gesammelten Daten aus dem Smartphone so umgerechnet, dass sie fürs Koordinatensystem des Motorrads (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 nachträglich ändern, kalibriert sich der Algorithmus langsam nach. Die Nachkalibrierung erfolgt langsamer als die Erstkalibrierung, damit die Unfälle durch eine schnelle Nachkalibrierung nicht übersehen werden.

### 2.5.2 Übersicht der bereits erkennbaren Unfälle

In der Abbildung 2.13 ist der Entscheidungsbaum abgebildet, wonach eine Unfallklassifizierung erfolgt wird. Die Einzelfeatures (Collision, GroundHit, TipOver) werden später separat erläutert.

Durch die Kombination der verschiedenen Parameterwerten lässt sich eine Unfallklasse angeben. Die Klasse enthält die Aussage, ob ein Unfall tatsächlich passierte, und entsprechend ebenfalls eine geschätzte Unfallschwere. Ein Unfall mit einer starken Kollision hat eine andere Klassifizierung als dieser mit einer leichten Kollision, wenn die anderen Parameter gleich bleiben. Die verschiedene TipOver-Möglichkeiten (Umkippen) unterscheiden zwischen einem seitlichen und frontalen Umkippen, was ebenfalls zu verschiedener Klassifizierung führen kann. Eine GroundHit-Erkennung könnte (vor allem bei einem leichten Aufprall) zu einer anderen Entscheidung führen.

Tabelle 2.2: *Erklärung der verschiedenen Features-Zustände im Entscheidungsbaum in der Abbildung 2.13 [Schnee u. a., 2021]*

	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 handelbar	0: upright bicycle 1: TipOver (Rollover) 2: NoseOver

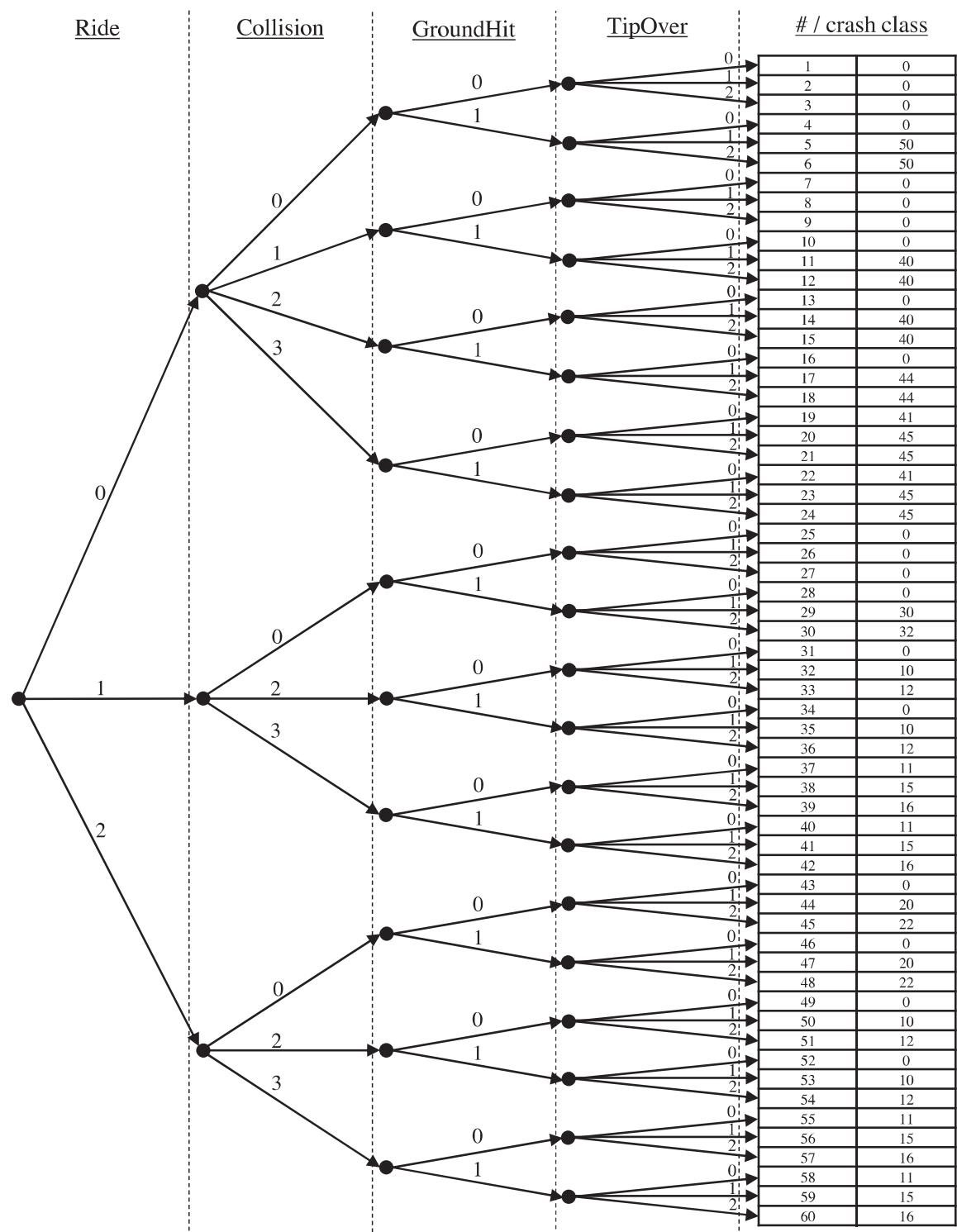


Abbildung 2.13: Entscheidungsbaum des Unfallerkennungsalgorithmus anhand der drei Features sowie des Fahrzustands [Schnee u. a., 2021]

Die Tabelle 2.2 beinhaltet die Klassifizierung für jede der vier Komponenten im Entscheidungsbaum sowie deren Bedeutung.

### 2.5.3 1. Feature: TipOver

Diese Komponente prüft den Neigungswinkel des Motorrads und stellt fest, wann 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 Kurvenfahrt wirken zwei Kräfte ( $F_q$ ) und ( $F_G$ ) am Motorrad (Abbildung 2.9). Wenn die 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  $45^\circ$ , 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. Feature: GroundHit

Dieses Modell dient dazu, den Schlag zu erkennen, wenn das Motorrad am Boden ankommt. Dieses Modell geht von der kinetischen 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 Zweirad nach einer ursprünglich aufrechten Position umkippt, und die Abbildung 2.14b das Umkippen eines Zweirads 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 ebenfalls angepasst. Je größer die Motorradneigung ist, desto kleiner ist der GroundHit-Schwellwert.

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

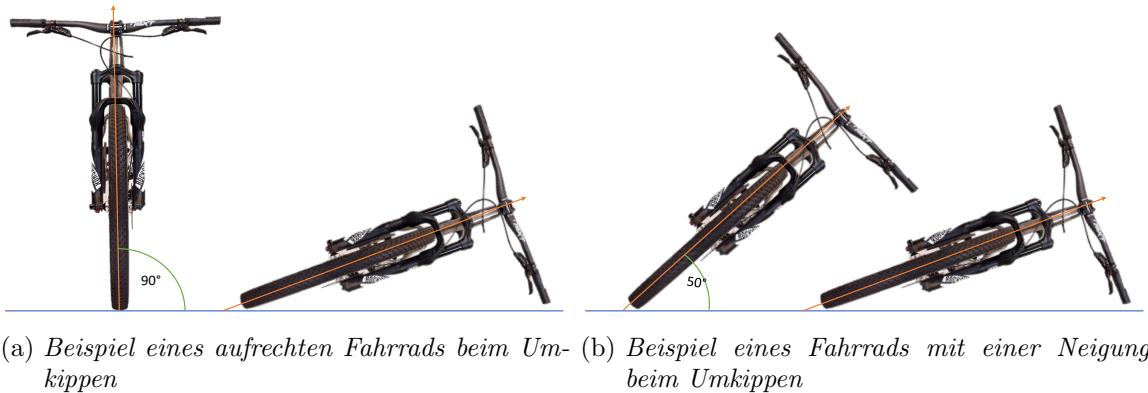


Abbildung 2.14: Beispiel eines Fahrrads beim Umskippen mit einer ursprünglichen aufrechten Position sowie mit einer ursprünglichen Neigung. Die Beispiele gelten ebenfalls für ein Motorrad

### Beispiel 1: Kein GroundHit

In diesem Beispiel wird eine Testfahrt ohne GroundHit durchgeführt und anschließend analysiert. Nach der Fahrt und bei späterer Simulation wurde entdeckt, dass die Kalibrierung nicht richtig war und manuell in der Simulation angepasst werden musste. 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 dafür 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. Während dieser Fahrt fand ebenfalls kein Unfall statt. Die oberen Grafiken zeigen die Geschwindigkeit (blau) sowie die GroundHit-Auslösungen aus dem Feld (grün) oder aus der Simulation (orange) 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 rechten Grafiken veranschaulichen einen kleinen kritischen Bereich der jeweiligen Signale, damit der Signalverlauf an der Stelle klar wird.

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.

### Beispiel 2: Unfall mit GroundHit

In der Abbildung 2.16 sind die Daten eines Unfalls mit einem erkannten GroundHit 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 rechten Grafiken veranschaulichen

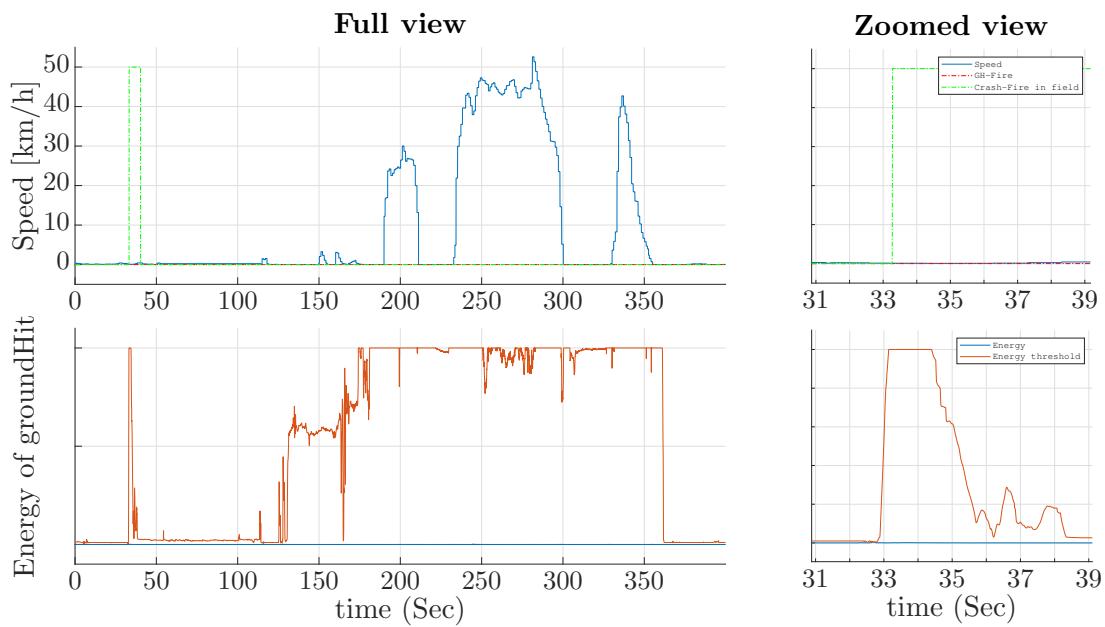


Abbildung 2.15: Verlauf der Energie sowie des Energieschwellwerts bei einer Testfahrt ohne GroundHit

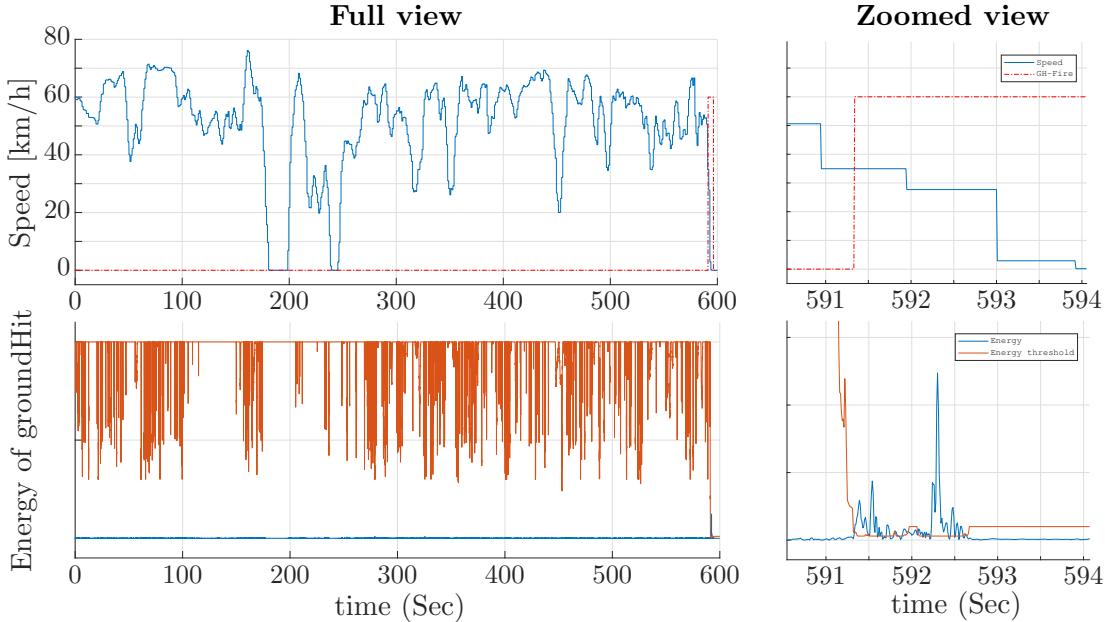


Abbildung 2.16: Verlauf der Energie sowie des Energieschwellwerts bei einer Echtfahrt mit GroundHit

einen kleinen Bereich der jeweiligen Signale, in dem einen Unfall aufgetreten ist. In der Abbildung ist zu erkennen, dass der Energieschwellwert bei der Sekunde 591 stark sinkt, da an dieser Stelle der Neigungswinkel ebenfalls größer wird. Der Wert wird an dieser Stelle von der tatsächlichen kinetischen Energie überschritten, was einen Alarm auslösen muss. Die obere rechte Grafik zeigt die Auslösung an der gleichen Stelle, an der den Schwellwert überschritten wird.

### 2.5.5 3. Feature: Collision

Dieses Feature dient dazu, eine Kollision zu erkennen. Es geht ebenfalls von der kinetische Energie (Gleichung 2.5) aus und hat vier mögliche Stufen, die in Tabelle 2.2 bereits erklärt wurden. Es ist allerdings wichtig zu wissen, dass der Kollisionsschwellwert von der Geschwindigkeit, die zu den Schwellwert umgekehrt proportional ist, und dem Neigungswinkel des Motorrads beeinflusst wird.

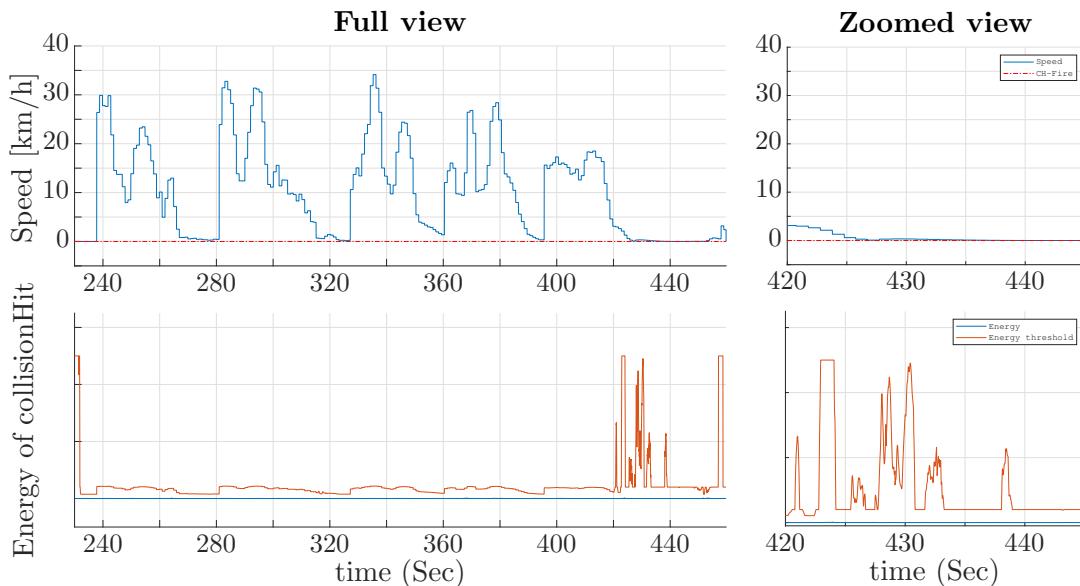


Abbildung 2.17: Verlauf der Energie sowie des Energieschwellwerts bei einer echten Fahrt ohne Kollision

In der Abbildung 2.17 ist der Fall dargestellt, in dem keine Kollision erkannt wurde. Während dieser Fahrt fand ebenfalls kein Unfall statt. Die oberen Grafiken zeigen die Geschwindigkeit (blau) sowie die Kollision-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 Signale, wo eine Überschneidung der Signale nicht klar ist, um diese besser erkennbar zu machen.

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

Analog dazu ist in der Abbildung 2.18 der Fall dargestellt, in dem eine Kollision erkannt wurde. Während dieser Fahrt fand ebenfalls ein Unfall statt. In der Grafik ist ei-

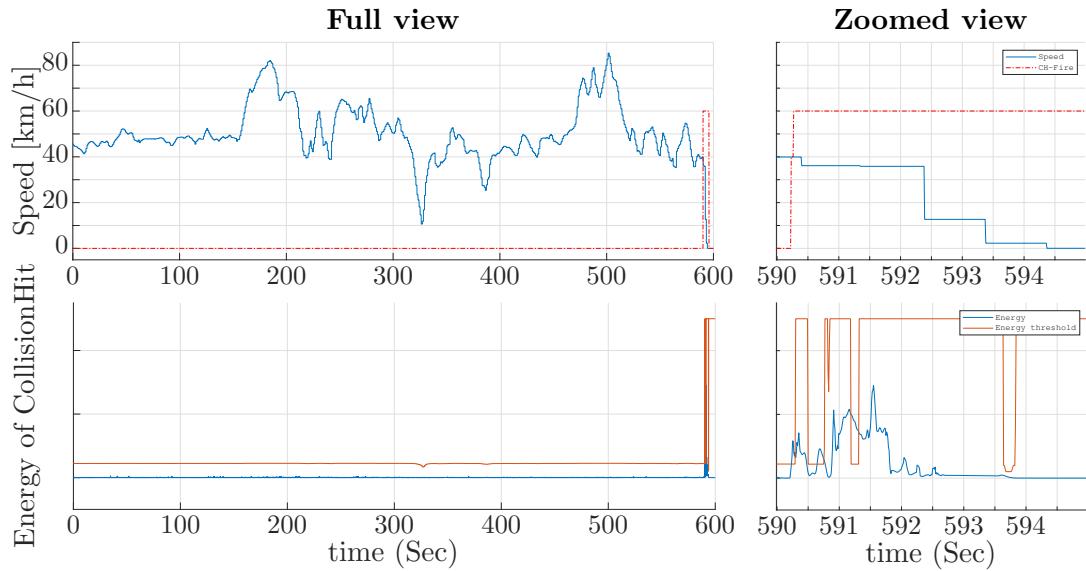


Abbildung 2.18: Verlauf der Energie sowie des Energieschwellwerts bei einer echten Fahrt mit einer Kollision

ne Kollision-Alarmauslösung etwa bei der Sekunde 590. Die rechten Grafiken zeigen die Kollisionsstelle im Signal genauer an und lassen die Überschreitung des Schwellwerts durch die tatsächliche Energie gut erkennen. An der Stelle, wo die Überschreitung geschehen ist, findet eine Alarmauslösung statt, was ebenfalls die Erwartungen entspricht.

# 3 Unfallerkennung im Pocket-Mode

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

## 3.1 Begründung für den Pocket-Mode

Die Anzahl der App-Nutzer (Unfallerkennungsalgorithmus) lag im November 2021 bei ca. 1190. Die Anzahl der zugelassenen 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-Institut im Auftrag von Bosch Help Connect im Zeitraum zwischen November und Dezember 2021 in vier Ländern (Deutschland, Frankreich, Italien, Spanien) mit jeweils 333 Befragten durchgeführt.

Die zwei wichtigsten relevanten 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 beieinander, 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 (Hosen-)Tasche beziehungsweise im Rucksack.

Aus diesem Grund ist die Entwicklung einer Unfallerkennung im Pocket-Mode wichtig, um das Smartphone nicht mehr unbedingt am Lenker befestigen zu müssen. Die Weiterentwicklung der Unfallerkennung erfolgt mit agilen Methoden.

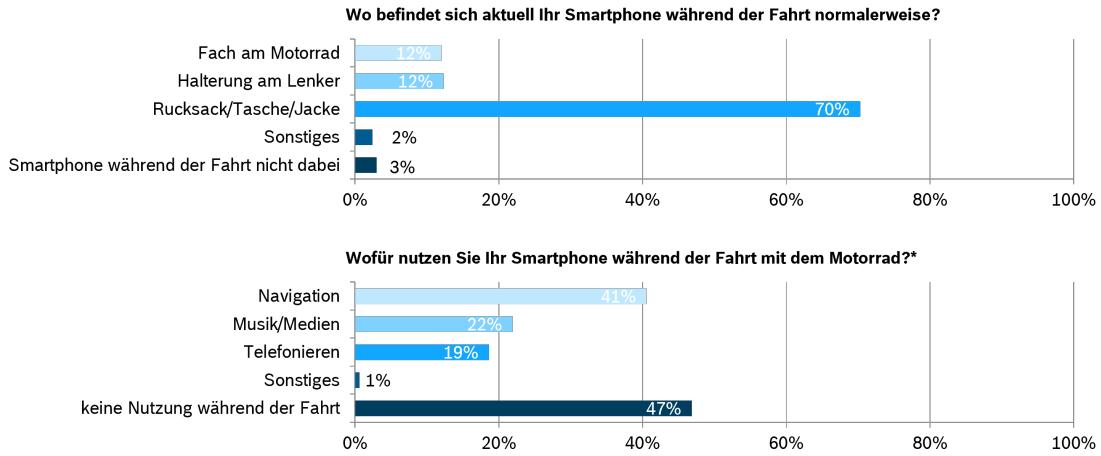


Abbildung 3.1: Ergebnisse der Umfrage vom Spiegelinstiute über das Taschenmodus

## 3.2 Kritische Szenarien

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

Um solche Konflikte zu vermeiden, wird eine Liste der Use- sowie Edgecases vorbereitet, in der die erwarteten Reaktionen des aktuellen Algorithmus im Verhältnis zum jeweiligen Szenario aufgelistet wird. Damit erfolgt eine Übersicht der möglichen Konflikte sowie der Fälle, in denen ein falscher Alarm ausgelöst werden könnte, und gleichzeitig eine mögliche Gegenmaßnahme.

Die Tabelle 3.1 zeigt die Liste. Die Spalte „Szenario“ enthält eine allgemeine Beschreibung des Szenarios, das auch unterteilt wird. In der Spalte „Beschreibung“ ist eine nähere Erklärung des Szenarios erläutert. Da das Verhalten des Smartphones in der Hosentaschen (am Oberschenkel) und am Oberkörper unterschiedlich sein könnte, werden diese separat betrachtet. 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. Unter der Bewegung des Oberkörpers fallen verschiedene Möglichkeiten, z.B. seitliches Lehnen, was der Algorithmus problemlos erkennen würde und keine falsche Alarne auslöst.

Bei den kritischen Szenarien, in denen der Algorithmus den Fall nicht richtig erkennen würde, ist eine mögliche Gegenmaßnahme zum Korrigieren der Algorithmus-Entscheidung beschrieben. Einige Szenarien (z.B. Auf der Fußraste während einer Fahrt stehen) sollen ausführlich getestet werden, um die richtige Reaktion des Algorithmus zu analysieren und gegebenenfalls eine Gegenmaßnahme einzuplanen.

Tabelle 3.1: Die Use- und Edgecases mit der erwarteten Reaktion des Algorithmus

ID	Szenario	Nähere Beschreibung	Algo-Reaktion (Erwartungen)		Bemerkung	Potenzielle Gegenmaßnahmen
			Handy am Körper	Handy am Motorrad		
1	Bewegung des Körpers	Umdrehen, Nach hinten schauen, Lenker mit einer Hand halten	IO	-	Keine kritische Winkeländerung um die X-Y-Achsen	-
2		Nach vorne/hinten lehnen	IO	-	Keine kritische Winkeländerung um die X-Y-Achsen	-
3		Seitliches Lehnen	IO	-	Keine Winkeländerung um die Z-Achse	-
4	Normale Fahrt	Fahrt, Bremse, Beschleunigung... usw.	IO	-	Bereits abgedeckt	-
5	Motorrad abstellen	-	IO	-	Bereits abgedeckt	-
6	Hanging off	In der Kruve hängen	IO	-	Keine ausreichende Winkeländerung für ein TipOver	-
7	Anhalten	Starke Bremse nach einer Fahrt und dann Fuß runtersetzen	IO	-	Keine kritische Winkeländerung um die X-Y-Achsen	-
8	Ab- und Aufsteigen	-	NIO	-	Bewegungsabhängig, enthält Winkeländerung (TO)	Ausführliches Testen

9	Laufen	-	NIO	-	Im Taschenmodus sehr wahrscheinlich	Lauferkennung einbauen und die Unfallerkennung während des Laufen deaktivieren
10	Nutzung des Smartphones		NIO	-	-	Erkennung durch das phone-lifting-Funktion
11	Auf der Fußrasste stehen	Stehend fahren	Testen	-	-	Ausführliches Testen
12	Rutschung des Handys in der Tasche	Starke Winkeländerung	NIO		Fahrt und Winkeländerung verursacht Unfallerkennung	Schnelles Nachkalibrierung oder in AGB bekannt machen (Handy befestigen)
13		Keine Winkeländerung (keine Umdrehung)	Testen	-	-	Ausführliches Testen
14	Wheelie fahren	Auf das Hinterrad fahren	NIO		Extremer Fall (Not intended use)	In AGB ausschließen
15	Auf der Motorrad-sitzbank stehen	-	NIO		Extremer Fall (Not intended use)	In AGB ausschließen

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

### 3.3 Lauferkennung

In der bereits bestehenden Version des Algorithmus wird davon ausgegangen, dass das Smartphone am Lenker befestigt wird. Wenn die Person das Handy nach einer Fahrt in die Hosen- beziehungsweise Jackentasche einsteckt und anfängt zu laufen, wird öfters ein falscher 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. Tankpause) vergisst (oder ignoriert), die Unfallerkennung zu deaktivieren, und mit dem Smartphone an sich läuft. Das führt dazu, dass die Anzahl der falschen Alarme im Pocket-Mode wesentlich steigt.

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

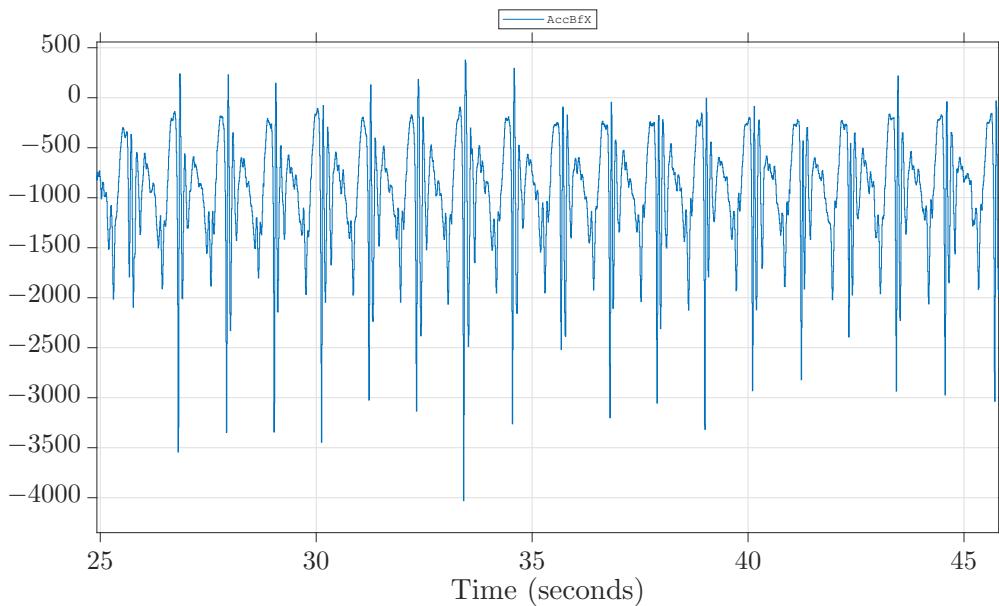


Abbildung 3.2: Beispiel eines Beschleunigungssignal beim Laufen

In der Abbildung 3.2 ist ein Beispielsignal aus dem Beschleunigungssensor im Smartphone während des Laufens abgebildet. Eine Person kann bis zu zwei Schritte pro Sekunde im Schnitt zurücklegen. In der Grafik können die Peaks innerhalb einer Sekunde aufgezählt werden und die durchschnittliche Anzahl der Schritte kann ermittelt werden. Wenn diese unter zwei pro Sekunde liegt, ist vom Laufen auszugehen, da ein

Motor mit mehr als zwei Umdrehungen pro Sekunde läuft und damit durch die Vibration erzeugten Peaks häufiger auftreten. Im nächsten Abschnitt werden diese Peaks aufgezählt, um die Anzahl der Schritte beziehungsweise Umdrehungen zu ermitteln.

### 3.3.1 Spitzenzähler

Wie bereits erwähnt wurde, kann die Person bis zu zwei Schritte pro Sekunde laufen. D.h. aus einem typischen Laufsignal (z.B. Abbildung 3.2) sollen maximal zwei Schritte pro Sekunde aufgezählt werden. Es soll ein System implementiert werden, das die Anzahl der Schritte beziehungsweise Spitzen aufzählt und den Mittelwert pro Sekunde zurückgibt. Zur Vereinfachung der Implementierung wird eine Simulink-Testumgebung aufgebaut, in der ein bekanntes Sinussignal generiert, dargestellt und verarbeitet wird. Die Abbildung 3.3 zeigt eine Testumgebung, in dem zwei Methoden zum Spitzenzähler

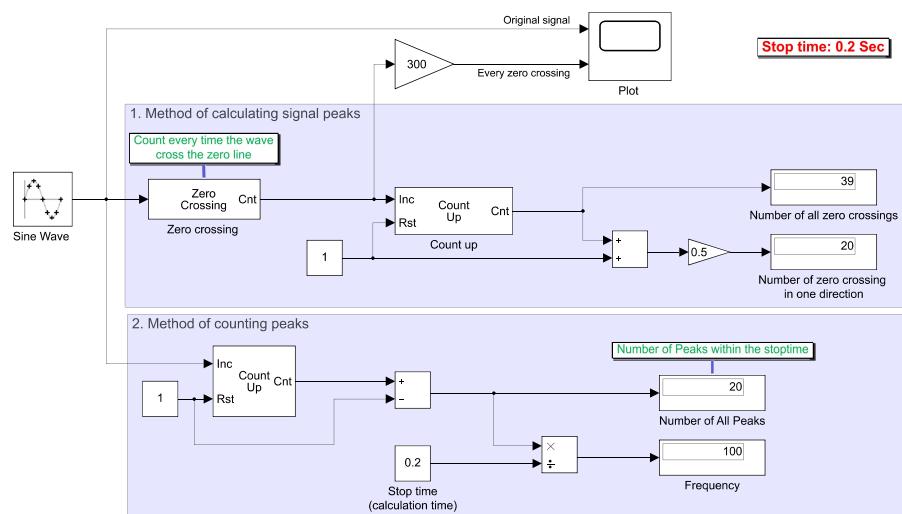


Abbildung 3.3: Testmodell der Lauferkennung - Spitzenzähler

implementiert wurden. 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 3.4 (blau) darstellen sowie wie oft das Signal die x-Achse überschneidet (rot). Aus der Grafik ist die Anzahl der Peaks einfach zu ermitteln und diese beträgt in diesem Fall umgerechnet 100 Hz.

In der ersten (oberen) Methode wird die Funktion „Zero Crossing“ verwendet. Diese zählt wie oft das Signal die x-Achse überquert. Dieses Modell liefert das richtige erwartete Ergebnis, wenn das Signal um die x-Achse dargestellt ist, und hilft allerdings nicht, wenn das Signal ein Offset hat (Verschiebung auf der y-Achse), da in diesem

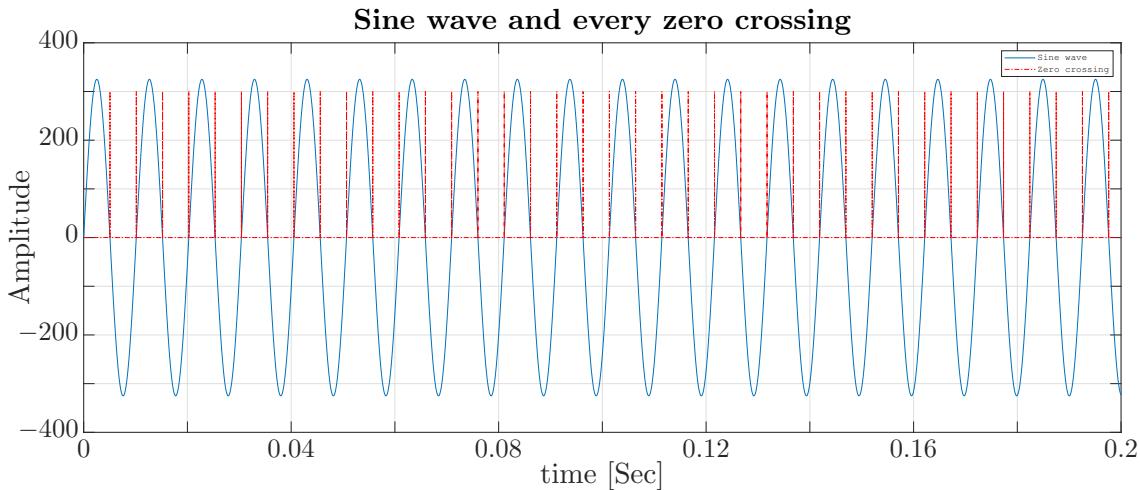


Abbildung 3.4: Darstellung des im Testmodell der Lauferkennung generierten Sinussignal sowie jede Überschneidung der x-Achse

Fall das Signal die x-Achse je nach der Amplitude nicht mehr überschneidet. Das führt dazu, dass das Ergebnis nicht mehr zuverlässig ist.

Eine zweite Methode hat sich dadurch ergeben, dass die Funktion „Counter up“ in dem Modell verwendet wird. Dieses Block zählt wie oft das Signal in die positive Richtung geht. Die neue Implementierung hat ein zuverlässigeres Ergebnis im Vergleich zum vorherigen Modell geliefert.

Beim Einsetzen des gleichen Vorgehens beziehungsweise Modells auf das echte Laufsignal (Abbildung 3.2) wird eine Frequenz von ca. 11 Hz beim Laufen ausgegeben, was eigentlich nicht wahr ist, 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 gedachte Modell nicht ausfiltern oder ignorieren, was zu einem falschen Ergebnis führt. Die Abbildung 3.5 stellt ein gutes Beispiel dieses Unterschieds dar, in dem das Rauschen die Anzahl der Spitzen erhöht.

Die Abbildung 3.5 zeigt zwei sinusförmige Signale. Die obere Grafik stellt ein einfaches Signal mit einer Frequenz von ungefähr  $f = 5,5$  Hz und die untere ein komplexes Signal dar. Das Modell hat für das untere Signal 19 Spitzen pro Sekunde geliefert, was der Wahrheit nicht entspricht.

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

### 3.3.2 Frequenzbasierte Lauferkennung

Das Laufsignal stellt ein wiederholtes Muster (Pattern) durch die Fußbewegungen dar, was auch durch eine Frequenzermittlung erkannt werden kann. Es soll ein Modell erstellt werden, das diese Frequenz ermitteln und auswerten kann. Analog zum

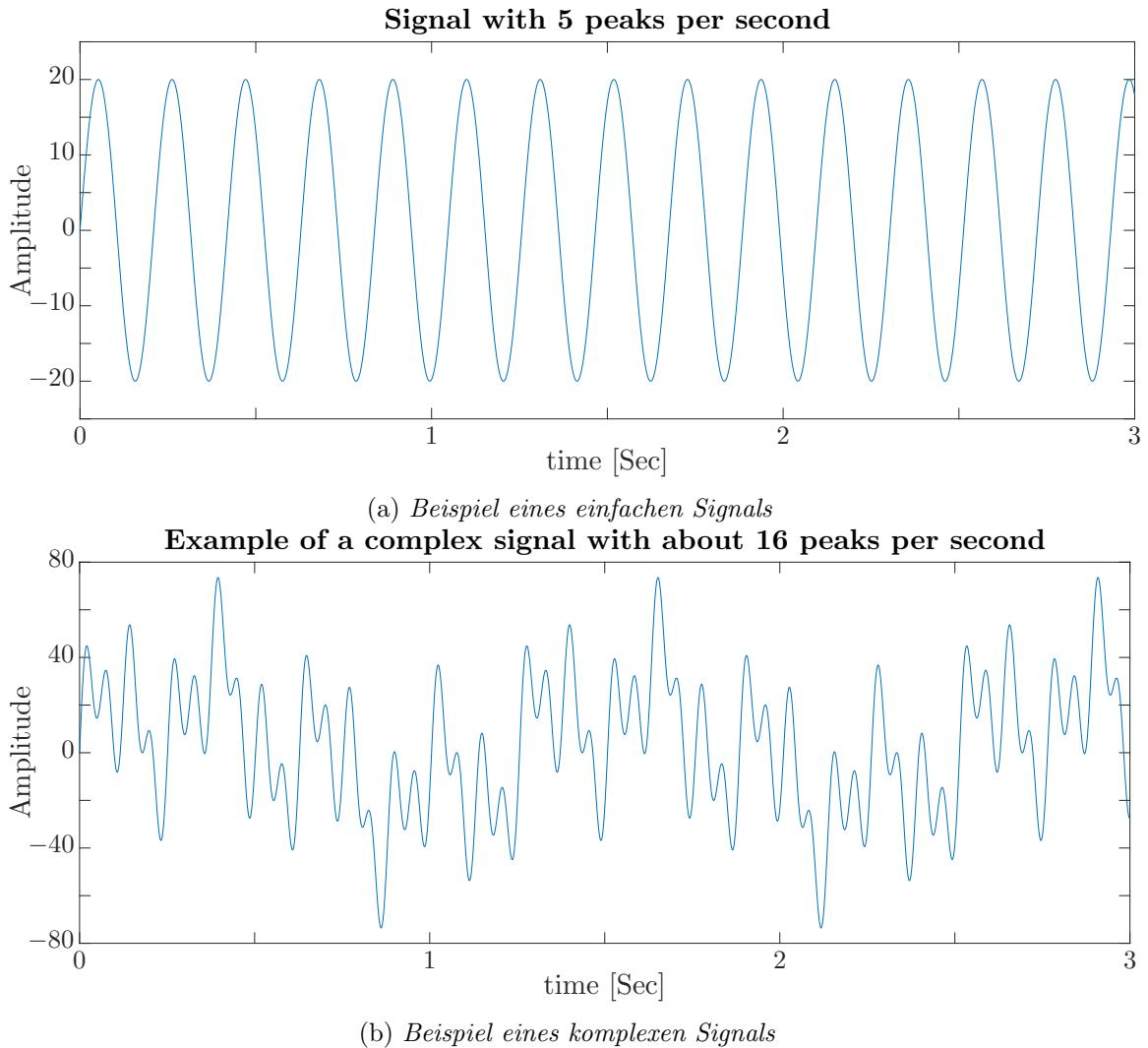


Abbildung 3.5: Skizze eines einfachen Signals sowie eines komplexeren Signals

Unterabschnitt 3.3.1 wird hier nochmal eine neue Hypothese festgelegt, die durch ein Testsystem überprüft werden soll.

Die Hypothese: Die Frequenz während des Laufens sollte kleiner als 2 Hz sein und beim Fahren über 7 Hz liegt. Wenn eine Frequenz von über 7 Hz ermittelt wird, ist eine Laufaktivität ausgeschlossen, da ein Mensch auf keinen Fall 7 Schritte innerhalb einer Sekunde zurücklegen kann. Die Fahr frequenz ist durch die Motorumdrehungen entstanden.

Die Transformation vom Zeitbereich zum Frequenzbereich wird durch eine FFT erreicht. Die App der Unfallerkennung hat eine Abtastrate von  $f_s = 100$  Hz und nimmt 100 Messwerte pro Sekunde auf. Bezogen auf die Nyquist-Frequenz (Gleichung 2.2) ergibt sich die maximale erkennbare Frequenz (Bandbreite)  $f_n = 50$  Hz.

## Spectrum Analyzer

In der Abbildung 3.6 sind die generierten Sinussignale mit den Frequenzen  $f = 23,8$  Hz,  $f = 47,8$  Hz und  $f = 1,1$  Hz zu sehen, die summiert werden, damit eine komplexes Signal entsteht. Die drei Sinussignale sowie deren Summe sind in der Abbildung 3.7 dargestellt.

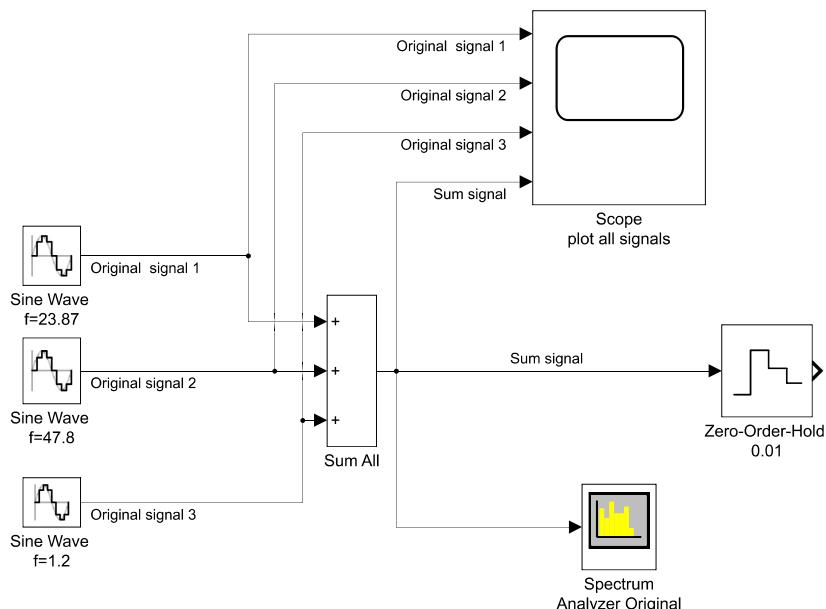


Abbildung 3.6: Testbeispiel - Frequenzbasierte Lauferkennung - Sinussignal

Danach wird in diesem Modell ein Block „Spectrum Analyzer“ als Referenz verwendet, was die Grundfrequenzen des komplexen Signals ausgibt. Der Benutzer kann die Spezifikationen vom „Spectrum Analyzer“ einstellen. Die Ausgabe vom „Spectrum Analyzer“ 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 3D-Darstellung „Frequenz-Zeit-Intensität“ (Spektrogramm), wobei die

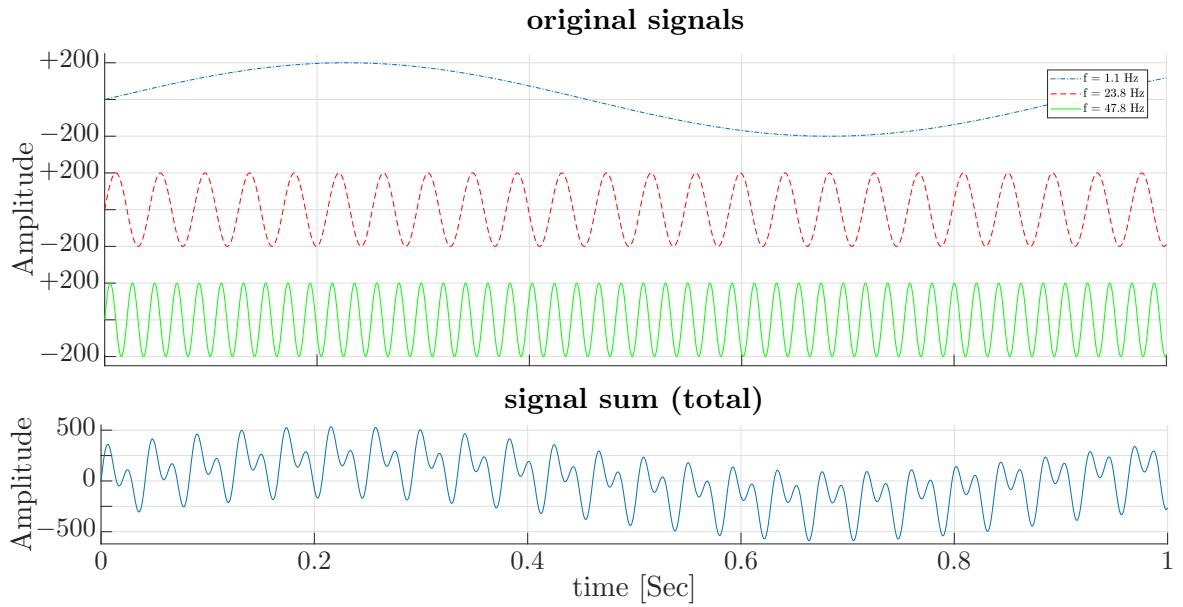


Abbildung 3.7: verschiedene Einzelsignale ( $f = 23,8$ ;  $f = 47,8$ ;  $f = 1,1$ ) mit deren Summe  $f_g$

Farbe die Intensität repräsentiert. Wenn die Abbildung näher betrachtet wird, sind die Grundfrequenzen von  $f = 23,8$  Hz,  $f = 47,8$  Hz und  $f = 1,1$  Hz gut sichtbar.

### Testmodell

Nachdem das theoretische Prinzip getestet wurde, ist nun eine Vervollständigung des Modells wichtig. Das vollständige Testmodell ist in der Abbildung 3.10 veranschaulicht. Das Ziel ist die Funktionalität des Prinzips zu überprüfen, bevor dieses mit einem Echtesignal getestet wird.

In dem System werden drei Sinussignale mit verschiedenen Frequenzen generiert, die aufsummiert werden, um ein komplexes Signal zu erstellen (Abbildung 3.7). Eine vereinfachtes Ablaufschema des Modells wird in der Abbildung 3.9 gezeigt.

Das Testmodell erstellt zuerst ein komplexes Signal mit bekannten Grundfrequenzen und konvertiert dieses mit dem „Zero-Order-Hold“-Block zu einem diskreten Signal, da die FFT nicht auf ein kontinuierliches Signal anwendbar ist. Danach wird das FFT-Fenster durch das „Buffer“-Block eingestellt und dann die FFT auf das entsprechende Fenster angewandt. Das Ergebnis der FFT wird weiterbearbeitet, in dem der Betrag gebildet und die Spiegelung entfernt wird. Das Resultat ist eine 2-D-Matrix mit den Intensitäten sowie ihren Indexen auf der Skala 1 bis 512, wobei die Indexe den Frequenzen entsprechen. Das Resultat ist in der Abbildung 3.11 dargestellt. Die Indexe werden extrahiert und in eine Skala von 1-100 umgerechnet, in dem diese mit 100/512 multipliziert werden. Das Endergebnis des Modells ist eine sortierte Liste der tatsächlichen Grundfrequenzen. Die ersten drei Werte haben eine wesentlich höhere Intensität als alle anderen und sind somit die gesuchten Frequenzen mit minimaler

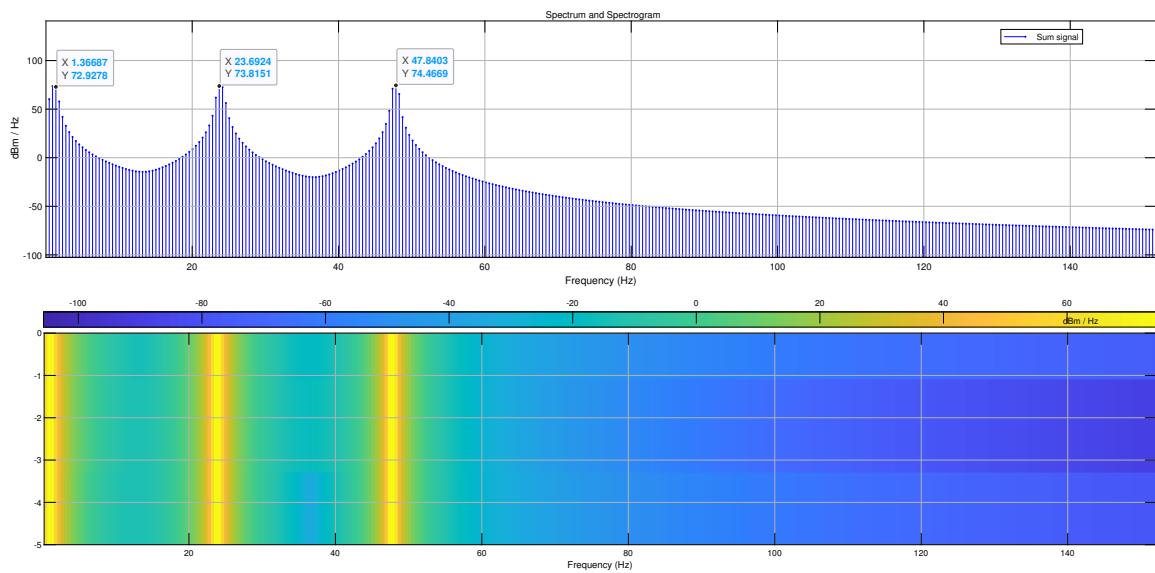


Abbildung 3.8: *Testbeispiel - frequenzbasierte Lauferkennung - Ausgabe des Spektrum-Analyzers im Fall eines komplexen Signals*

Abweichung.

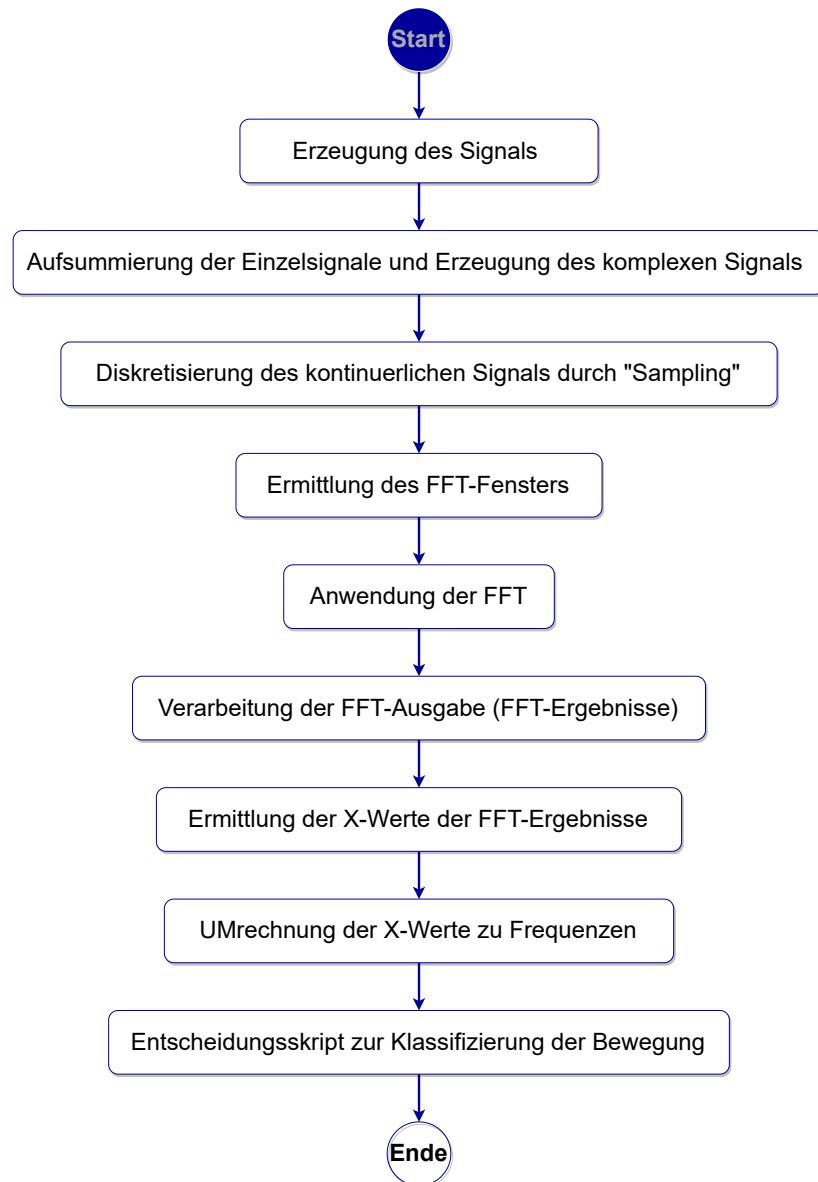


Abbildung 3.9: Ablaufschema des Testmodells der frequenzbasierten Lauferkennung

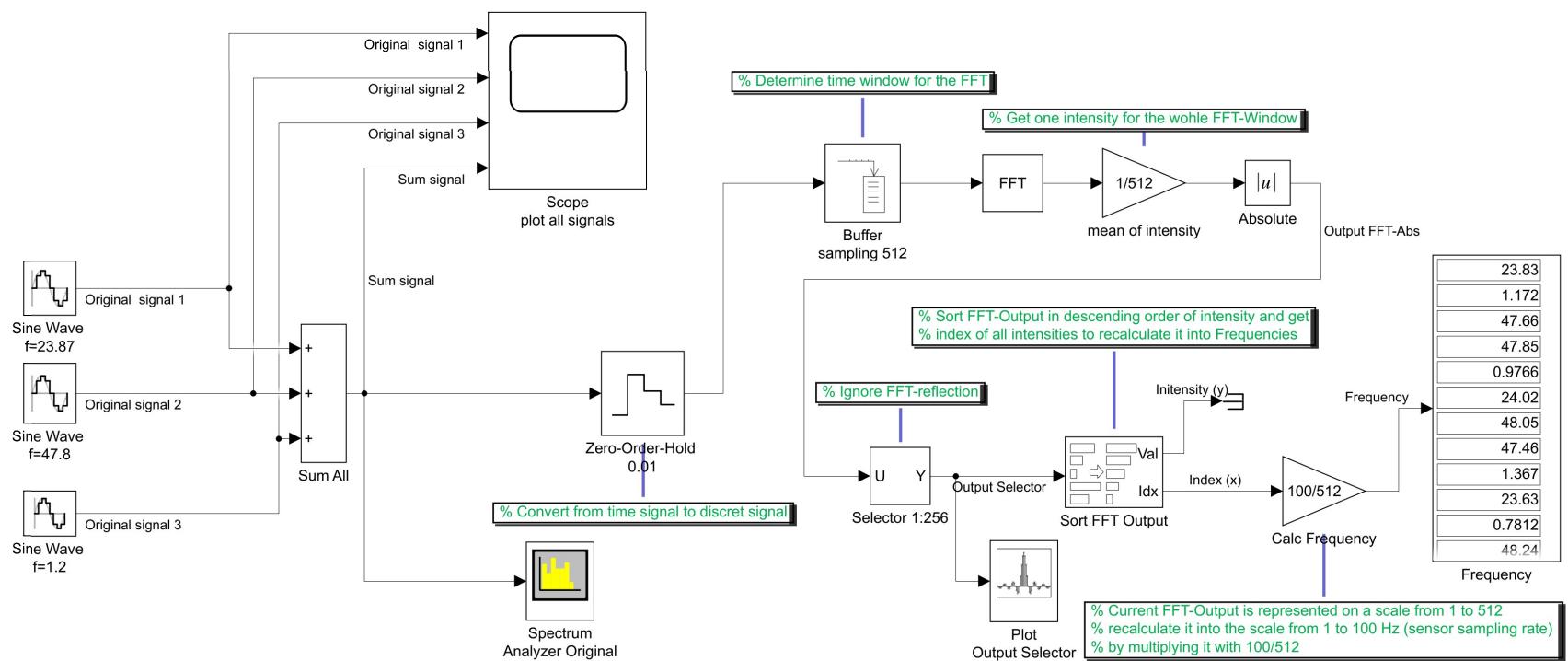


Abbildung 3.10: Testbeispiel - Frequenzbasierte Lauferkennung - FFT

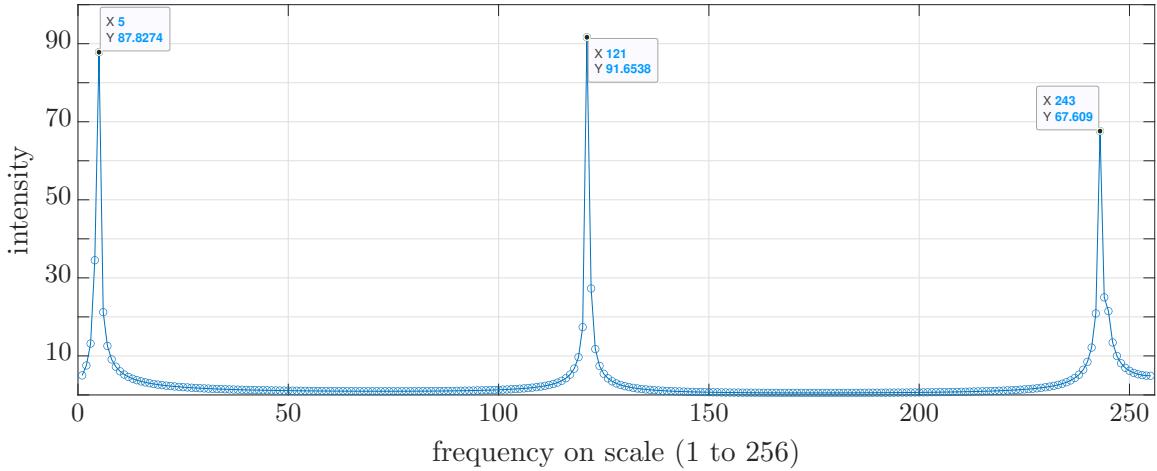


Abbildung 3.11: Das Ergebnis der FFT - Nach Entfernung der Spiegelung und Ermittlung der Beträge

Nachdem die Ergebnisse des Testmodells auf Richtigkeit geprüft wurden, wird das System mit einem Echtesignal getestet.

### Anwendung auf ein Echtesignal

Zum Anwenden auf ein Echtesignal wird ein Untersystem „MotionDetection“ erstellt, das die Bewegung (Laufen oder Fahren) erkennt und ausgibt. In der Abbildung 3.12 ist einen Teil des genannten Modells sichtbar.

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

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

Im 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 wird das Fenster auf  $B_L = 2,56\text{ s}$  (d.h. 256 Proben) mit einer Überlappung von 50% eingestellt. Die minimale erkennbare Frequenz lässt sich durch

$$f_{min} = \frac{1}{B_L} = 0,3906 \text{ Hz}$$

berechnen (siehe Unterabschnitt 2.3.1).

Eine größeres FFT-Fenster  $B_L$  hätte eine bessere Frequenzermittlung erlaubt, 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“ unterdrückt.

Der Ausgang dieses Teils ist eine 2-D-Liste auf einer Skala von 1 bis 256, die sortiert werden muss.

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

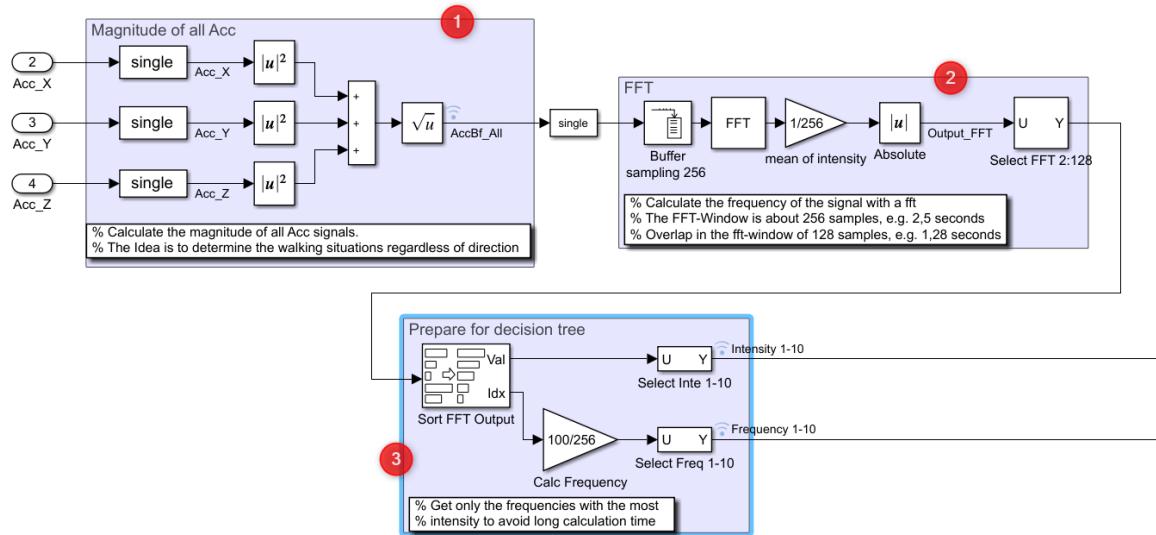


Abbildung 3.12: Das vollständige Echtmodell der Frequenzermittlung im Rahmen der Lauferkennung

Die von dem dritten Teil erzeugten Daten werden zu einer Matlabdatei (der Übersicht in der Abbildung 3.13) geleitet, wo eine Entscheidung getroffen wird.

### Entscheidungskriterien - Matlabskript

Die Abbildung 3.13 zeigt die Matlab-Funktion, die die Entscheidung übers Laufen treffen soll. Die Eingänge der Funktion sind die zehn Frequenzen mit den höchsten zehn Intensitäten sowie die aktuelle Geschwindigkeit. Die letzte durch die Funktion erkannte Aktivität sowie deren Zeitpunkt werden auch in die Funktion weitergeleitet. Nach dem Durchlauf liefert die Matlab-Funktion eine ID-Zahl, die einer Aktivität entspricht. Der Aktivität-ID-Zusammenhang ist in der Tabelle 3.2 abgebildet. Das Skript unterscheidet zwischen vier Klassen. Es wird hauptsächlich zwischen Laufen und Fahren unterschieden, sollte diese Entscheidung nicht möglich, liefert das Skript eine Aussage (Unbekannt oder Konflikt) zurück. Die entscheidende Funktion ist Teil des Simulink-Systems.

Die Abbildung 3.14 zeigt den vereinfachten Entscheidungsbaum, wonach die Aktivitätserkennung erfolgt.

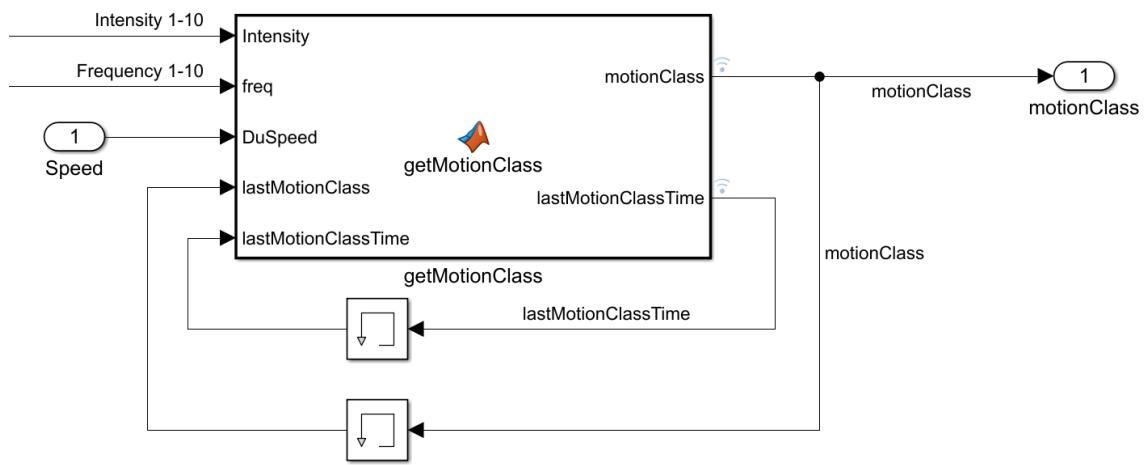


Abbildung 3.13: Ein- und Ausgänge des Entscheidungsskripts

Tabelle 3.2: Ausgangsmöglichkeiten der Entscheidungsfunktion

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

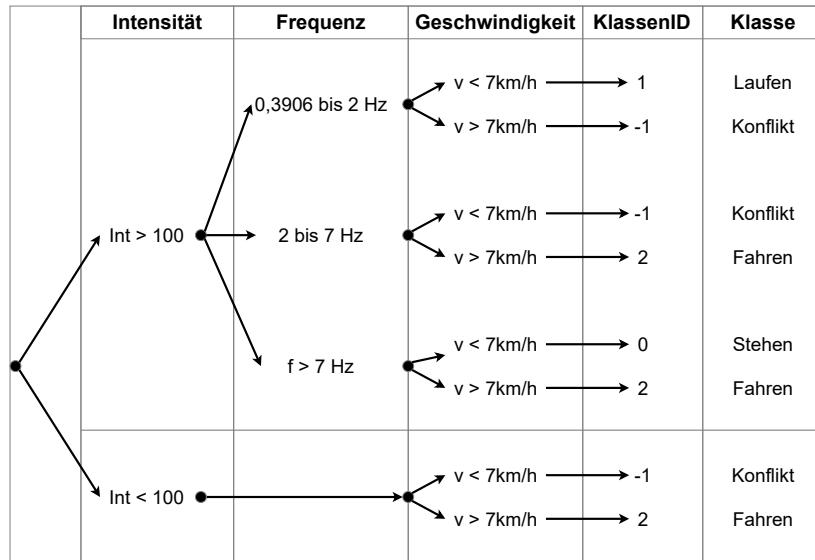


Abbildung 3.14: Entscheidungsbaum der Aktivitätserkennung

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.

- 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 vertrauenswürdig sein und es werden bis zu vier nachfolgende Intensitäten untersucht. Sollte sich immer noch keine zulässige Intensität ergeben, wird sofort nach der Geschwindigkeit geschaut und diese für die Entscheidung verwendet. Z.B. bei einer Geschwindigkeit von 30 km/h ist von einer Fahrt 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 Konfliktfall auszugehen. In diesem Fall werden bis zu vier der größten nachfolgenden Frequenzen überprüft.

### **Testbeispiel**

Die Signale in der Abbildung 3.15 sind aus einem Echtesignal ausgeschnitten, in dem der Fahrer nach einer Fahrt gelaufen ist. Das Fahren ist in der Abbildung 3.15a sowie das Laufen in der Abbildung 3.15b abgebildet.

Nachdem die Aktivitätserkennung mit diesem Signal getestet wurde, hat das Modell die richtigen Frequenzen erkannt. Das Modell hat für das erste Teilsignal aus der Abbildung 3.15a eine Frequenz von durchschnittlich 20 Hz geliefert, das Teilsignal aus der Abbildung 3.15b eine Frequenz von etwa 1,8 Hz. Die Frequenzermittlung der Lauferkennung war sehr nahe an der Realität. Nachdem die Frequenzen ermittelt wurden, sind diese ins Entscheidungsskript weiterzuleiten. Das Skript entscheidet dann mit einer Geschwindigkeitsprüfung, welche Aktivität (Fahren oder Laufen) vorgekommen ist.

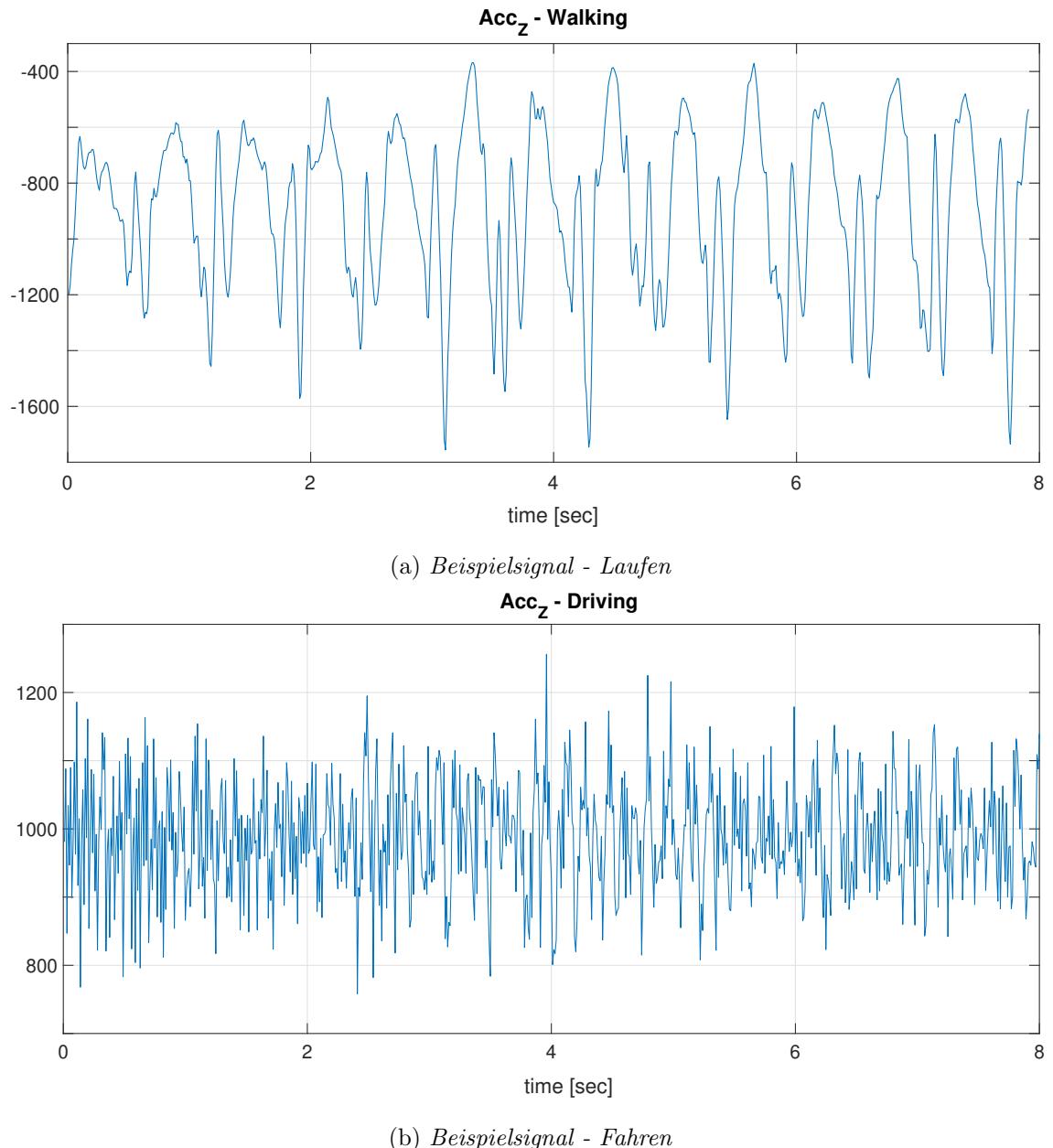


Abbildung 3.15: Abgeschnittene Teile eines Beispieldesigns

### Sinnvolle Werte auswählen

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

- **Intensität:** Der Entscheidungsbaum sucht nach einer Intensität von über 100, damit die dazugehörige Frequenz zuverlässig zur Entscheidung verwendet werden kann. Nach mehreren Testungen wurde eine Intensität von 100 ausgewählt. Ein niedrigerer Intensitätswert führt zu einer möglichen falschen Entscheidung, da das Rauschen beziehungsweise die kleinen Frequenzen aus dem Frequenzbereich fälschlicherweise in das Entscheidungsskript eingeleitet werden.
- **Frequenzbereich:** dieser entspricht der Frequenzbereiche vom Laufen und vom Fahren.  
Der Lauffrequenzbereich beträgt in der Regel weniger als 2 Hz. Das Motorrad erzeugt wesentlich mehr als 7 Hz (mehr als 420 Umdrehungen pro Minuten).
- **Geschwindigkeit:** In dem Skript wird 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 allerdings mit einer Geschwindigkeit zwischen 5 km/h und 10 km/h.

## 3.4 Auf- und Absteigen

In der Übergangsphase 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, erkennt es die Winkeländerung ganz klar mit. Eine theoretische Betrachtung dieser Bewegung lässt eine Fehlalarmauslösung erwarten, da die Winkeländerung zu der Sitzposition über 45° liegt, was in der Regel durch das Modell „TipOver“ als Umkippen erkannt wird. Aus diesem Grund wird dieses Szenario auf seine Richtigkeit getestet.

## 3.5 Anhalten

In diesem Abschnitt wird ein Szenario aus der Tabelle (Tabelle 3.1) analysiert, in dem der Fahrer während einer Fahrt an eine Ampel für kurze Zeit anhält und seinen Fuß auf den Boden herunter setzt. In diesem Fall ist das Smartphone in der Hosentasche des bewegenden Beins platziert. Es wird angenommen, 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 Beinpositionen, was das Modell „TipOver“ aktiviert und zu einer Alarmauslösung führt. Die erste Position hat das Bein während einer

Fahrt mit einer horizontalen Beinstellung. Wenn der Fuß am Boden ist, steht das Bein in einer vertikalen Position.

In der Abbildung 3.16 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.  $90^\circ$ . Diese Winkeländerung aktiviert das „TipOver“-Modell und schlägt eine Unfallerkennung vor. Dieses Szenario wird ebenfalls auf seine Richtigkeit getestet und analysiert.

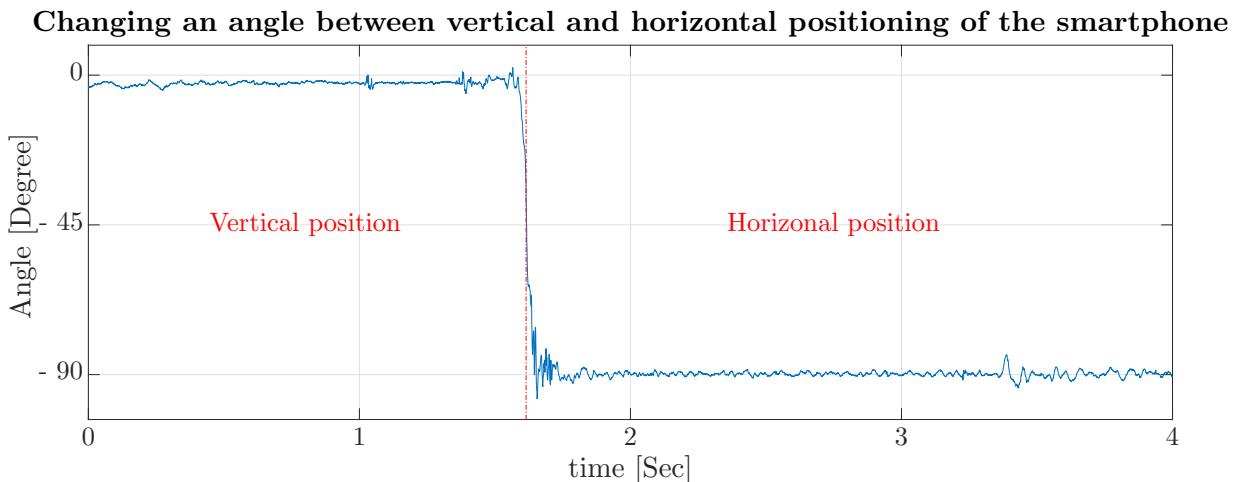


Abbildung 3.16: Winkeländerung im Signal zwischen vertikaler sowie horizontaler Smartphone-Positionierung

## 3.6 Verifikation des Algorithmus

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

Im Rahmen dieser Verifizierung wird eine Versuchsplanung durchgeführt. Danach wird die Versuchsplanung sowie das Versuchsvorgehen erläutert.

### 3.6.1 Versuchsplanung

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

Tabelle 3.3: Die Zusammenfassung der getesteten Szenarien

Sz-Nr	Anz. Durchläufe	Beschreibung	Smartphones				
			iPhon SE	Samsung A5	Samsung S10	Pixel 3	Huawei P30
1	2	Fahrt, Auf der Fußraste Stehen Auf- und Absteigen Laufen	Lenker	Hosentasche	Tankrucksack	Brusstasche	Rucksack
2	1	Anhalten	Lenker	x	x	Hosentasche	Hosentasche
3	1	Handy im Tankrucksack rutschen lassen (keine Winkeländerung)	Lenker	x	x	Tankrucksack	Tankrucksack
4	1	Handy im Tankrucksack rutschen lassen (Winkel- änderung)	Lenker	x	x	Tankrucksack	Tankrucksack
5	2	Crash simulieren; Rucksack während einer Fahrt fallen lassen	Lenker	x	x	Rucksack	Rucksack

Die Tabelle 3.1 zeigt eine Liste der Use- und Edgecases, in der einige Szenarien getestet werden sollen. Es wurden diesbezüglich fünf verschiedene Szenarien geplant. Die Tabelle 3.3 fasst diese zusammen und zeigt die verwendeten Smartphones sowie ihre Ablageorte (Hosentasche, Brusttasche... usw.) während des Testens. Das erste Szenario soll die implementierte Lauferkennung testen sowie das Verhalten beim Auf- und Absteigen aufzeichnen. Dadurch soll das Signalunterschied zwischen der Fahrt und dem Laufen erkennbar sein. Zusätzlich wird die Winkeländerung zwischen den zwei Fahrerpositionen (stehend/sitzend) während einer Fahrt analysiert und getestet. Das zweite Szenario testet die tatsächliche Winkeländerung zur ursprünglichen Position des Smartphones nach der Kalibrierung sowie die Reaktion des Algorithmus beim Anhalten, da dieser Fall während einer Fahrt sehr oft vorkommen könnte.

Das dritte sowie vierte Szenario sollen das Verhalten des Smartphones testen, wenn es sich im Tankrucksack ohne Befestigung befindet. Das letzte Szenario testet die Reaktion des Algorithmus, wenn der Fahrer während einer Fahrt stürzt und ob der Algorithmus diesen Fall zuverlässig erkennt. Um die zufälligen Fehler bei der Signalaufnahme zu vermeiden werden bei der Testung immer mehr als ein Smartphone verwendet. Es werden ebenfalls bei manchen Szenarien mehrere Smartphones an mehreren Ablageorten angebracht, um möglichst viele Daten gleichzeitig zu sammeln und das unterschiedliche Verhalten des Algorithmus an unterschiedlichen Orten zu erforschen. Die Auswertung dieser Daten wird je nach der verfügbaren Zeit durchgeführt.

Der Versuchsablauf wird vor dem Testen vorbereitet und möglichst detailliert geplant, damit während des Testens kein Zeitverlust entsteht. Nachfolgend werden die allgemeinen Schritte des Versuchsablaufs aufgelistet.

- 1 Befestigung und Einstellung der Kameras am Lenker und an der Brust, um die Wahrheit (Ground Truth) für die spätere Auswertung aufzunehmen
- 2 Befestigung der Smartphones (inklusive des Referenzhandys am Lenker) an den geplanten Stellen, Einschaltung und Prüfung der Signalaufnahme
- 3 Eine Kalibrierfahrt
- 4 Prüfung der Smartphonekalibrierung
- 5 Start der Videoaufnahme und Dokumentation von Datum sowie Uhrzeit
- 6 Durchführung des Testversuchs (Variiert je nach dem Versuchsszenario)
- 7 Fahrtende und absteigen
- 8 Datenexport und -Übertragung
- 9 Daten im Zielordner prüfen
- 10 Löschung der Daten im Smartphone
- 11 Vorbereitung des nächsten Versuchs beziehungsweise Durchlaufs



Der Versuch fängt mit einer Kalibrierfahrt an, um die Smartphones und den Algorithmus zu kalibrieren

Nachdem die Kalibrierung gesichert wurde, fängt der tatsächliche Versuch an und fährt die Person stehend weiter

Nach der Fahrt stehend setzt sich der Fahrer wieder hin und fährt eine Runde normal

Die Versuchsperson steigt ab

In der Abbildung ist die Beinbewegung mit einer sehr starken Winkeländerung gut zu sehen. Die Bein wird hochgehoben und nach hinten gestreckt

Loslaufen

Laufen

Die Versuchsperson steigt wieder auf und fährt eine Runde weiter vor dem Versuchsende

Abbildung 3.17: Die Versuchsschritte des ersten Versuchsszenarios, indem die Versuchsperson sitzend und stehend fährt, absteigt, läuft und aufsteigen

Das erste Szenario ist das Wichtigste für diese Arbeit, weil dadurch die implementierte Lauferkennung sowie der 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: Fahrt mit verschiedener Fahrerpositionierung und Laufen sowie der Übergang dazwischen (Auf- und Absteigen). Die Bilder in der Abbildung 3.17 zeigen ein Paar Screenshots von dem Versuch.
  - 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 eine normale Fahrt für ca. eine Minute
  - 8 Fahrt in stehender Fahrerposition für ca. 10 Sekunden
  - 9 Normale Fahrt für ca. eine Minute
  - 10 Fahrtende und Fahrerabstieg

Gleichzeitig werden die durch das integrierte Google-Tool erkannten Aktivitäten aufgenommen, damit diese mit der Ausgabe des implementierten Modell „Aktivitätserkennung“ verglichen werden. Das Ziel ist zu prüfen, ob es zuverlässiger wäre, das integrierte Tool zu verwenden.

#### 3.6.2 Referenz-Aktivitätsdaten (Ground truth)

Die Lauferkennung beziehungsweise Aktivitätserkennung ergibt die Aussage (Fahren oder Laufen) und soll getestet werden. Für die Testung muss der tatsächliche Verlauf einer Fahrt bekannt werden. Zu diesem Zweck werden die Tests mit Videoaufnahmen nach der Durchführung optisch manuell mithilfe eines intern entwickelten LabVIEW-Tool ausgewertet. Diese Versuche werden mit Videos aufgenommen, die allerdings nur einen kleinen Teil der aufgezeichneten Signale abdecken. 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 durchführen. 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 Labels der Aktivitäten beinhaltet. Diese Tabelle kann mit den Ergebnissen der Lauferkennung verglichen werden.

Die erwähnten Labels muss der Benutzer vorher definieren und in das Tool importieren. In der Tabelle 3.4 ist die Tabelle der definierten Labels zu sehen. Die Tabelle hat vier 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. „Driving“ repräsentiert die Fahrt und „Walking“ das Laufen.

In der Abbildung 3.18 ist ein verkürztes Beispiel der exportierten Tabelle, in der die neue Spalte „GroundTruthID“ zu sehen ist. In der Tabelle sind mehrere ausgeschnittene Phasen enthalten, die auch durch die GroundTruth-Daten unterschieden werden.

timeStamp	acc	gyr	speed	angle	groundTruthID	groundTruth
167298	-987	953	29.67358	1.982636	2	Driving
167299	-855	-3893	29.67358	1.8454454	2	Driving
167300	-831	-6851	29.67358	1.6600937	2	Driving
167302	-710	1347	29.67358	1.3874936	0	No Movement
167895	-780	-5993	41.201656	-4.920791	0	No Movement
167896	-783	-7708	42.258587	-4.896966	0	No Movement
167897	-832	-5459	42.258587	-4.8466144	1	Walking
167898	-833	-1916	42.258587	-4.7872252	1	Walking
167899	-874	516	42.258587	-4.7530065	1	Walking
170005	143	-1907	30.31143	2.5846107	0	No Movement
170006	152	-1837	30.31143	2.5952225	0	No Movement
170007	130	-1697	30.31143	2.611972	0	No Movement
170008	138	-988	30.31143	2.6297727	2	Driving
170009	148	78	30.31143	2.6537619	2	Driving
170010	141	78	30.31143	2.7010856	2	Driving
170012	151	1006	3.5	2.760584	0	No Movement
170013	150	796	2.1	2.7934167	0	No Movement
170014	148	796	0	2.8240378	0	No Movement
170020	158	2694	30.31143	3.1481538	-1	Undefined
170021	153	1986	30.31143	3.2035604	-1	Undefined
170022	149	1487	30.31143	3.2570984	-1	Undefined

Abbildung 3.18: Beispiel der exportierten Tabelle mit der neuen Spalte

Tabelle 3.4: Die definierte Labels von Groundtruth

ID	Class	F-Key
-1	Undefined	F4
0	No Movement	F5
1	Walking	F1
2	Driving	F2



# 4 Ergebnisse

In diesem Kapitel wird die implementierte Lauferkennung ausgewertet und es werden ebenfalls die Ergebnisse der Testversuche vorgestellt und diskutiert.

Die Auswertung der Versuche beziehungsweise der Testergebnisse wird aus zeitlichen Gründen qualitativ aber nicht quantitativ sein.

## 4.1 Verifizierungsversuche

Die bereits im Unterabschnitt 3.6.1 aufgelisteten Versuchsszenarien wurden alle mindestens einmal getestet. Einige Testszenarien wurden in zwei Testdurchläufen durchgeführt. Nach der Durchführung der Verifizierungsversuche werden die Ergebnisse analysiert und diskutiert.

Die Lauferkennung ist ein großer Teil dieser Arbeit und wird demnächst mit den tatsächlichen Daten (Ground Truth) sowie der von Google integrierte Aktivitätserkennung verglichen.

## 4.2 Lauferkennung

In der Abbildung 4.1 werden die verschiedenen Methoden der Aktivitätserkennung sowie die Geschwindigkeit in der obersten Grafik über die Zeit dargestellt.

Die zweite Grafik zeigt die tatsächliche Aktivitätsdaten (Ground Truth), die in vier Klassen (Tabelle 4.1) unterteilt sind. Die Aktivitätsbereiche sind auch mit vertikalen roten Linien optisch unterteilt, die auch in anderer Grafiken sichtbar sind, damit diese einfach und schnell zugeordnet werden können. Bei der Sekunde 460 s sowie 500 s ist die Versuchsperson vom Motorrad ab- oder aufgestiegen, deswegen werden diese zwei Stellen mit „No Motion“ klassifiziert. Ab der Sekunde 556 s gibt es keine Videoaufnahme mehr und es wurde „Undefined“ eingetragen.

In der dritten Grafik ist die Ausgabe der im Rahmen dieser Arbeit implementierten Lauferkennung beziehungsweise Aktivitätserkennung abgebildet. Es ist zu bemerken, dass die Lauferkennung eine sehr große Übereinstimmung mit der Ground Truth hat. Laufen sowie Fahren werden dadurch gut erkannt. Das Auf- und Absteigen bei den Sekunden 465 s und 508 s wird der „No Motion“-Klasse vom Lauferkennung-Modell zugeordnet.

Die letzte Grafik stellt die Ausgabe der im Smartphone integrierten Aktivitätserkennung dar, die für diese Auswertung in die gleiche Klassifizierung aus der Tabelle 4.1 umgerechnet wird.

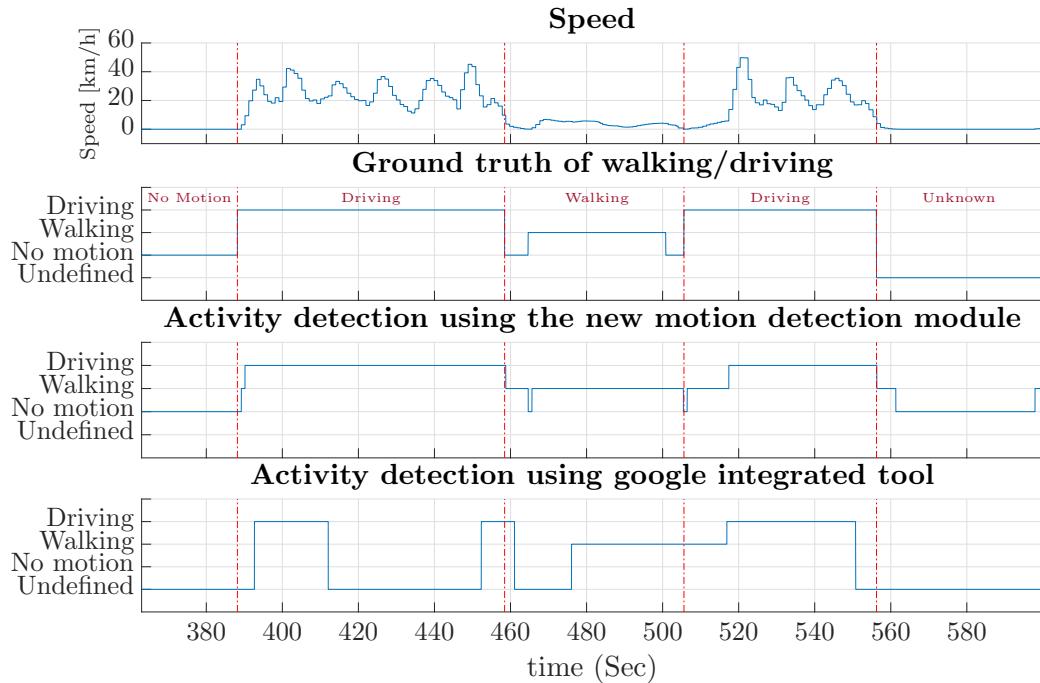


Abbildung 4.1: *Ergebnis des Lauferkennungsmodells*

Aus der Abbildung 4.1 ist letztendlich zu erkennen, dass die Aktivitätserkennung durch das implementierte Model gute Ergebnisse liefert, die mit der Wahrheit sehr gut übereinstimmen. Im Vergleich zu der Google-Aktivitätserkennung (unterste Grafik) hat das Model eine wesentlich bessere Aussage getroffen. Die implementierte Aktivitätserkennung verbraucht keine große Rechenzeit (ca. 2 bis 3 Sekunden) und erfolgt in Echtzeit mit einer kleinen Verzögerung (ca. 2,5 s), da die FFT erst durchführbar ist, wenn 256 Messwerte vorhanden sind.

Tabelle 4.1: *Beschreibung der Klassen in der Abbildung 4.1*

Klasse	Beschreibung
Undefined	Unbekannt oder keine Videoaufnahme
No Motion	Keine klassifizierbare Bewegung
Walking	Laufen
Driving	Fahren

### 4.3 Verschiedene Fahrerpositionierung

Die Abbildung 4.2 dient zur Auswertung des ersten Testszenarios aus der Tabelle 3.3, in dem die Person während einer Fahrt auf der Fußraste des Motorrads für paar Sekunden steht und sich wieder setzt. Das Ziel dieses Tests ist zu prüfen, ob die Änderung der Fahrerposition während einer Fahrt zur falschen Alarmauslösung führt.

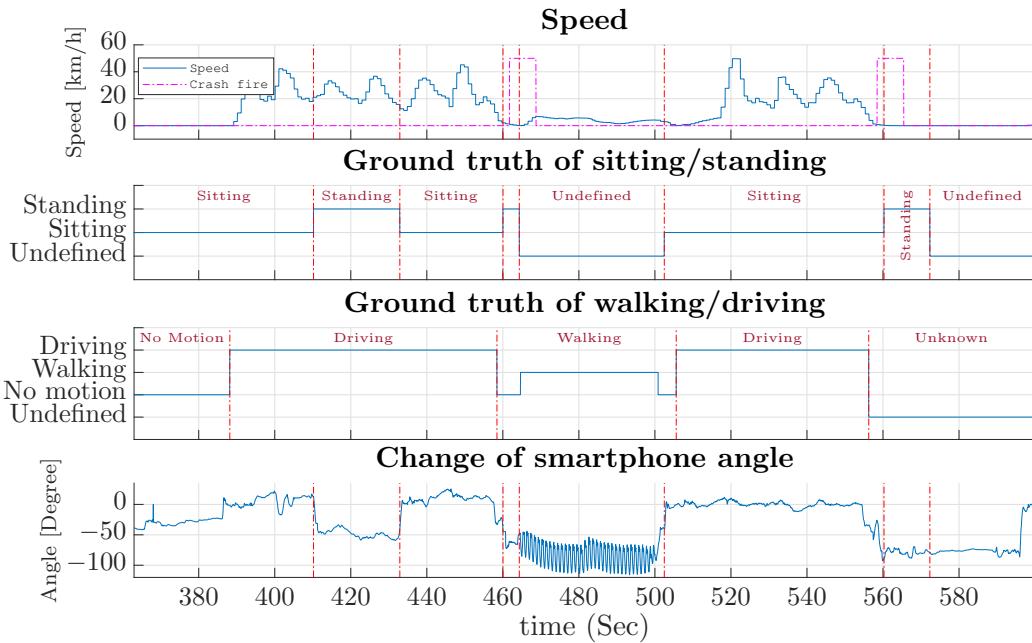


Abbildung 4.2: Zeitlicher Verlauf der Geschwindigkeit, Ground Truth der Fahreraktivität und -Position sowie der Winkeländerung

Diese Darstellung zeigt die Geschwindigkeit, die Fahreraktivität (Fahren/Laufen), die Fahrerposition sowie die Winkeländerung über die Zeit.

Die Geschwindigkeit (blau) sowie die Alarmauslösungen während der Fahrt (lila) sind in der ersten Grafik abgebildet. Die zweite Grafik stellt die tatsächliche Fahrerposition (Ground Truth) über die Zeit dar, die in drei Klassen (Sitzen, Stehen oder unbekannt) unterteilt ist. Die Fahrerpositionsbereiche sind ebenfalls durch vertikale rote Linien optisch geteilt, die auch in den anderen Grafiken sichtbar sind, damit diese einfach und schnell zugeordnet werden können. In dem Zeitbereich zwischen 465 s und 505 s ist die Versuchsperson gelaufen, deswegen lautet die Klassifizierung „Unbekannt“, da die geplanten Klassen einen solchen Fall nicht abdecken.

In der dritten Grafik sind die tatsächlichen Aktivitätsdaten (Ground Truth) darstellt, die in vier Klassen (Tabelle 4.1) unterteilt sind. Die Aktivitätsbereiche sind auch mit vertikalen roten Linien optisch unterteilt. Beim Betrachten der zweiten und dritten Grafik lässt sich ein guter Überblick über die Fahrt verschaffen. Zwischen den Sekunden 388 s und 459 s fand eine Fahrt statt, während dieser hat der Fahrer zwischen den Sekunden 410 s und 433 s gestanden.

Die Kurve aus der letzten Grafik zeigt die Winkeländerung des Smartphones im Vergleich zur ursprünglichen Position nach der Kalibrierung. Aus der Darstellung ist die Winkeländerung während der ersten Fahrt, als die Fahrerposition sich verändert hat, sehr gut sichtbar. Das Laufmuster ist in diesem Signal zwischen 465 s und 500 s ebenfalls gut erkennbar.

Es ist wichtig zu bemerken, dass die Änderung der Fahrerposition während einer Fahrt keine Fehlalarme ausgelöst hat, da die Winkeländerung in so einem Fall nicht groß

genug ist, um das „TipOver“-Model zu aktivieren. Beim Absteigen (ca. 460 s und 560 s) gibt es allerdings eine Alarmauslösung, da es an der Stelle eine sehr große Winkeländerung gibt, da der Fahrer seine Beine nach außen und hinten gestreckt hat.

## 4.4 Anhalten

Dieser Abschnitt diskutiert das Ergebnis des zweiten Testszenarios aus der Tabelle 3.3, in dem die Versuchsperson während einer Fahrt kurz anhält und seinen Fuß auf den Boden herunter setzt.

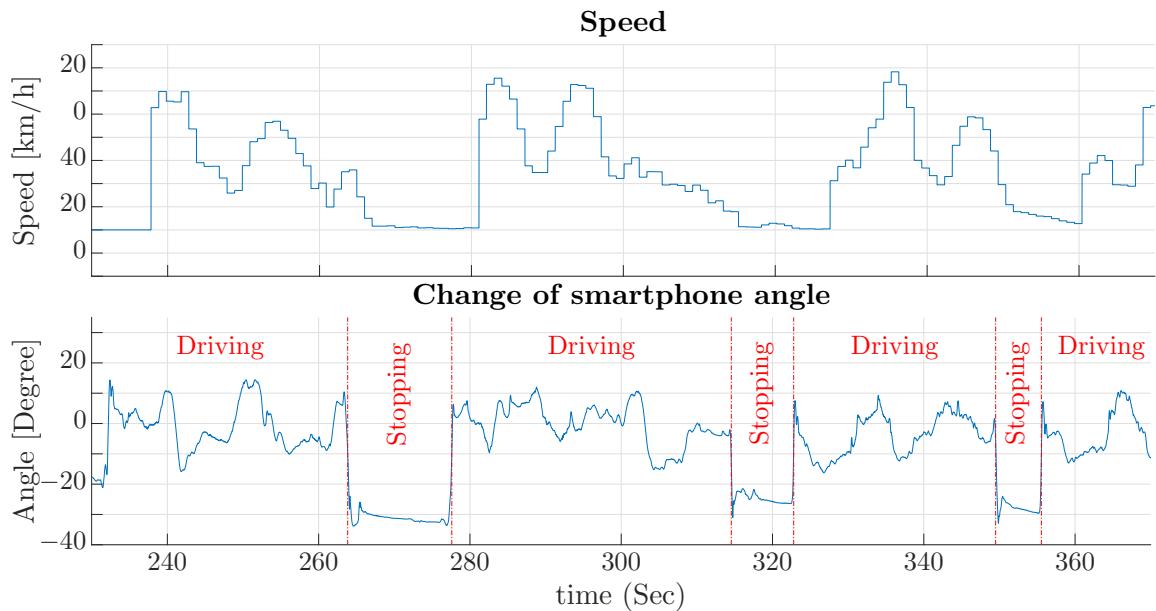


Abbildung 4.3: Winkeländerung des Smartphones durch das Anhalten

Die Grafik Abbildung 4.3 zeigt die Geschwindigkeit sowie die Winkeländerung zur ursprünglichen Position nach der Kalibrierung über die Zeit. Während des Tests ist zu sehen, dass der Fahrer dreimal anhält und seinen Fuß herunter setzt, was in der Geschwindigkeitskurve sichtbar ist (262 s, 315 s und 350 s). Es wurde während des Tests entgegen den Erwartungen keine Fehlalarmauslösung gegeben. Nach einer genaueren Analyse ist der Grund bekannt worden.

Die Winkeländerung beträgt nicht  $90^\circ$  sondern nur ca.  $20^\circ$ - $30^\circ$  (Abbildung 4.4). Die Person hat ihr Bein beim Sitzen nicht genau horizontal sondern leicht nach unten geneigt. Wenn der Fahrer sein Fuß herunter setzt, ist diese ebenfalls nicht genau vertikal sondern leicht geneigt mit einem Winkel von ca.  $10^\circ$ - $20^\circ$  zur Vertikalen . Daher ist die Winkeländerung nicht über  $45^\circ$  und sollte auch zu keinen Alarmauslösungen führen.



(a) Beinposition während einer Fahrt



(b) Beinposition Beim Anhalten

Abbildung 4.4: Beinpositionen während einer Fahrt und beim Anhalten



## 5 Ausblick

Da die Motorradfahrer eine besonders gefährdete Gruppe unter den Verkehrsteilnehmern darstellen, soll eine automatisierte Unfallerkennung mithilfe des Smartphones von großer Bedeutung sein, welche Unfälle erkennt, klassifiziert und mithilfe von Bosch geschulten Agenten die lokale Rettungsdienste alarmiert. Diese Arbeit war ein Teil der Entwicklung des Unfallerkennungsalgorithmus im Taschenmodus, damit die Unfallerkennung auch mit dem Smartphone in der Jacken- oder Hosentasche funktionsfähig bleibt.

Aus zeitlichen Gründen und da das Projekt „Bosch Help Connect“ Ende 2022 eingestellt wurde, war eine umfangreiche Entwicklung des Taschenmodus leider nicht mehr möglich. Im Folgenden wird ein Ausblick auf mögliche Ideen zur Weiterentwicklung der bisherigen Version des Algorithmus gegeben, welche an den Inhalt dieser Arbeit anknüpfen sowie weiteren Nutzen für die Unfallerkennung bringen würden.

Im aktuellen Unfallerkennungsalgorithmus ist klar, dass der Fahrer die Unfallerkennung manuell aktivieren und deaktivieren muss. Eine kontinuierliche Aktivitätserkennung im Hintergrund könnte implementiert werden und bei einer Motorradfahrterkennung wird die Unfallerkennung automatisch aktiviert. Das spart dem Benutzer Zeit und Aufwand, vor Allem wenn das Smartphone sich im Rucksack befindet.

Die Aktivität- beziehungsweise Lauferkennung soll weiterentwickelt und zuverlässiger implementiert werden, indem andere Smartphone-Daten wie z.B. Bildschirmaktivität oder App-Nutzung mit analysiert werden. Eine Funktion zur Erkennung der Smartphone-Anhebung (mobile lifting detection) sollte die Smartphonenuutzung ebenfalls erkennen. Die Unfallerkennung sollte in diesem Fall temporär deaktiviert werden, damit falsche Alarmauslösungen während dieser Benutzung verhindert werden.

Die Verifizierung der im Rahmen dieser Arbeit implementierten Lauferkennung fand nur qualitativ statt, diese soll demnächst quantitativ durchgeführt werden. Nach einer ausreichende Testung sollen die Schwellwerte wie Laufgeschwindigkeit oder Fahrfrequenz gegebenenfalls angepasst werden.

Der Algorithmus zur Unfallerkennung soll auch für anderen Transportmittel wie z.B. Mofa, E-Scooter, usw. angepasst werden, damit dieser einheitlich bei mehreren Transportmittel verwendbar bleibt. Eine Sturzerkennung der Person während des Laufens oder der anderen alltäglichen Aktivitäten könnte sehr hilfreich sein.

Die Reaktionstestung des Algorithmus soll ebenfalls für alle mögliche Szenarien ausreichend getestet werden, damit eine signifikante Aussage getroffen werden kann. Bei einer Abweichung von der Erwartung ist eine Gegenmaßnahme zu testen und gegebenenfalls diese Maßnahme anzupassen.

Die Ermittlung der Unfallschwere soll zukünftig mit höherer Zuverlässigkeit entwickelt werden. Bei einem schweren erkannten Unfall kann die Zwischenphase mit dem Bosch

## *5 Ausblick*

---

Agent übersprungen werden, in dem die Rettungsdienste automatisch kontaktiert werden und dadurch Zeit gespart wird.

# Literaturverzeichnis

[AWS 2022] AWS: *What is Mobile Application Development?* <https://aws.amazon.com/mobile/mobile-application-development/>. Version: Oktober 2022. – Aufgerufen am 22.11.2022

[Bosch 2021] BOSCH: *Help Connect vereint automatische Unfallerkennung, Notruffunktion und persönlichen Notfalldienst.* <https://www.bosch-presse.de/pressportal/de/de/help-connect-vereint-automatische-unfallerkennung-notruffunktion-und-persoenlichen-notfalldienst-225737.html>. Version: März 2021. – Aufgerufen am 21.11.2022

[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

[Cossalter u. a. 2014] Kapitel 1. In: COSSALTER, Vittore ; LOT, Roberto ; MASSARO, Matteo: *Motorcycle Dynamics*. John Wiley and Sons, Ltd, 2014. – ISBN 9781118536391, 1-42

[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](https://developer.android.com/guide/topics/sensors/sensors_motion#sensors-motion-accel). Version: August 2022. – Aufgerufen am 01.08.2022

[Elbayoumy 2018] ELBAYOUMY, Mahmoud Hany M.: *MOBILITAPP PROJECT: USER TRANSPORTATION ACTIVITYRECOGNITION VIA MOBILE DEVICESSENSORS*, TALLINN UNIVERSITY OF TECHNOLOGY, Diplomarbeit, 2018. <file:///E:/Eagleget/ce3b057f59404cb298543bdc3a2d165e.pdf>

[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](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

- [Gibbons 2021] GIBBONS, Jordan: *New smartphone tech could save your life after a crash.* <https://www.motorcyclenews.com/news/new-tech/bosch-accident-detection-system/>. Version: April 2021. – Aufgerufen am 21.11.2022
- [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
- [Iov u. a. 2004] IOV, F. ; HANSEN, A. D. ; SORENSEN, P. ; BLAABJERG, 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
- [Joy 2021] JOY, Ashwin: *26 Real-world Applications of C Language.* <https://pythonistaplanet.com/applications-of-c-language/>. Version: September 2021. – Aufgerufen am 22.11.2022
- [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
- [Mathworks 2021] MATHWORKS: *MATLAB Coder.* <https://de.mathworks.com/products/matlab-coder.html>. Version: März 2021. – Aufgerufen am 21.11.2022
- [Moon 2020] MOON, M.: *Bosch's motorcycle crash detection automatically alerts emergency services.* <https://www.engadget.com/bosch-motorcycle-crash-detection-060119306.html>. Version: Juni 2020. – Aufgerufen am 21.11.2022
- [Moto-Passion 2018] MOTO-PASSION: *Motorcycle Crashes on the Road.* <https://www.youtube.com/@motopassion3883>. 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

[Rasheed u. a. 2015] RASHEED, Muhammad B. ; JAVAID, Nadeem ; ALGHAMDI, Turki A. ; MUKHTAR, Sana ; QASIM, Umar ; KHAN, Zahoor A. ; RAJA, M. Haris B.: Evaluation of Human Activity Recognition and Fall Detection Using Android Phone. In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, 2015, 163-170

[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

[Schanze 2017] SCHANZE, Robert: *Gyroskop-Sensor im Handy: Was macht er? Wie funktioniert's?* <https://www.giga.de/apps/android/specials/gyroskop-sensor-im-handy-was-macht-er-wie-funktionierts-einfach-erklaert/>. Version: März 2017. – Aufgerufen am 19.12.2022

[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

[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

[Tran u. Phan 2016] TRAN, Duc N. ; PHAN, Duy D.: Human Activities Recognition in Android Smartphone Using Support Vector Machine. In: *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, 2016

[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 Matlab-Entscheidungsfunktion

```
1 function [motionClass, lastMotionClassTime, frequency, Inten] =  
2     getMotionClass(Intensity, freq, DuSpeed, lastMotionClass,  
3     lastMotionClassTime)  
4 %%%%%% function to detect the activity and differentiate  
5 %%%%%% between walking and driving. When a walking detected,  
6 %%%%%% the Crash detection should be paused.  
7 %%%%%%  
8 %%%%%% The function output should be a MotionClass-ID of 4 Classes  
9 %%%%%% ID = -1      => Error  
10 %%%%%% ID = 0      => No Movement  
11 %%%%%% ID = 1      => Walking  
12 %%%%%% ID = 2      => Driving  
13 %%%%%%  
14 % Check the biggest five frequencies. The idea is to determine the right  
15 % detected frequency  
16 for iFq = 1:5  
17     %% first check the intensity of the frequency to get only  
18     %% the frequency with an enough energy (intensity)  
19     if Intensity(iFq) > 100  
20         %% check the actual frequency. Three intervals are defined  
21         %% (walking, driving, between zones)  
22         if freq(iFq) >= 7  
23             %% high frequency detected (most likely driving)  
24  
25             %% doppel check with the speed  
26             if DuSpeed < 5  
27                 %% high frequency but very low speed => No Movement  
28                 motionClass = 0;  
29             else  
30                 %% there is movement => Driving  
31                 motionClass = 2;  
32             end  
33             break  
34  
35         elseif freq(iFq) >= 0.390625000 && freq(iFq) <= 2  
36             %% Low frequency, (most likely walking)  
37  
38             %% doppel check with the speed  
39             if DuSpeed <=7 && DuSpeed > 0
```

```

40         % walking speed => Walking
41         motionClass = 1;
42         break
43     elseif DuSpeed == 0
44         % no speed => no Movement
45         motionClass = 0;
46         break
47     else
48         % speed > 7 kmh => not walking => conflict
49         % => check next frequency
50         motionClass = -1;
51         continue;
52     end
53 elseif freq(iFq) > 2 && freq(iFq) <7
54     % Between zone. Maybe walking and maybe driving.
55
56     %% Double check with the speed.
57     %% If the speed is bigger than 7 km/h than he must
58     %% be driving. If the speed is smaller than 7 km/h
59     %% it should stay undeclared.the person might be
60     %% driving with no GPS-Signal (e.g. no speed)
61 if DuSpeed >= 7
62     % Driving speed => Driving
63     motionClass = 2;
64     break;
65 else
66     % Not driving speed => Conflict => check next frequency
67     motionClass = -1;
68     continue;
69 end
70 else
71     % not intended to get here; all usecases were
72     % defined previously
73     motionClass = lastMotionClass; % keep last detected class
74     break
75 end
76 else
77     % The intensity is smaller than 100 => the frequency could 't
78     % be used => determine motionClass only using the speed
79 if DuSpeed >= 7
80     % Driving speed => Driving
81     motionClass = 2;
82     break;
83 elseif DuSpeed < 7 && DuSpeed > 0.1
84     % Walking speed => Walking
85     motionClass = 1;
86     break;
87 else
88     % Speed = 0 => No movement
89     motionClass = 0;
90 end
91 end

```

```
92 end  
93 % save the frequency used to determine motionClass  
94 frequency = freq(iFq);  
95 % save the intensity used to determine motionClass  
96 Inten = Intensity(iFq);  
97 end
```