

Spezialisierung einer Unfallerkennung am Zweirad mittels Smartphone auf Unabhängigkeit der Trageposition

Master-Thesis

des Studiengangs Medizintechnik an der
Universität Stuttgart

Cand. M.Sc. Oays Darwish

3480821

Prüfer: Dr. Thomas Günther
Betreuer: Dipl.- Ing. Nino Häberlen
Abgabedatum: 15.12.2022



Masterarbeit

Spezialisierung einer Unfallerkennung am Zweirad mittels Smartphone auf Unabhängigkeit der Trageposition

Unternehmensbeschreibung

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

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

18. März 2022

Aufgabenbeschreibung

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

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

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

Kontakt

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

Kurzzusammenfassung

DE

Abstract

Englisch

Danksagung

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

Stuttgart, im November 2022

Oays Darwish

Inhaltsverzeichnis

Kurzzusammenfassung	v
Abstract	v
Danksagung	vii
Abbildungsverzeichnis	xi
Tabellenverzeichnis	xiii
Abkürzungsverzeichnis	xv
Symbolverzeichnis	xvii
1 Einleitung	1
2 Grundlagen	3
2.1 Unfall	3
2.1.1 Deffinition	3
2.1.2 Zeitliche Phasen eines Unfalls	4
2.1.3 Mechanik und Biomechanik des Unfalls	5
2.1.4 Statistische Zahlen über Unfälle	8
2.2 Technische Grundlagen	13
2.2.1 Senesoren und Signale der Smartphones	13
2.2.2 Matlab/Simulink	15
2.3 Mathematische Grundlagen	17
2.3.1 FFT	17
2.4 Methoden der Softwareentwicklung	19
2.5 Unfallerkennungsalgorithmus	19
2.5.1 Kalibrierung	20
2.5.2 Übersicht der bereits erkennbaren Unfälle	20
2.5.3 TipOver	22
2.5.4 GroundHit	22
2.5.5 CollisionHit	27
3 Unfallerkennung im Pocket-Mode	29
3.1 Kritische Szenarien/Fälle	30

3.2	Lauferkennung	31
3.2.1	Lauferkennung - Spitzendzähler	32
3.2.2	Frequenzbasierte Lauferkennung	36
3.3	Auf- und Absteigen	49
3.4	An Ampel stehen und Fuß runter	49
3.5	Verifikation des Algorithmus'	50
3.5.1	Groundtruth-Daten sammeln, Tatsächliche Aktivitätsdaten . . .	50
4	Ergebnisse	55
5	Ausblick	57
Literaturverzeichnis		xix
A	Anhang	xxiii
A.1	Entscheidungsfunktion	xxiii
Eidesstattliche Erklärung		xxv

Abbildungsverzeichnis

2.1	Einfache Darstellung des Regelkreises “Fahrer-Fahrzeug-Umfeld” [H. Appel, 2002]	4
2.2	Beispiel der zeitlichen Phasen einer kritischen Fahrsituation [H. Appel, 2002]	5
2.3	Vereinfachte Darstellung der Kräfte bei stationärer Kurvenfahrt (Abb. 3-2)	7
2.4	Verschiedene Kurventechniken bei gleicher Geschwindigkeit (Abb. 3-6) .	7
2.5	Geschwindigkeitsüberschreitungen nach PKW- und Motorradfahrer an 7 Messstationen aus der VSS-Datenbank [d. Maire, 2020]	9
2.6	Unfallschwere über Geschwindigkeit aus der GAIDA-Datenbank[d. Maire, 2020]	10
2.7	Anzahl der Unfälle gegenüber Unfallgegner	11
2.8	Anteil der aktiven sowie passiven Unfälle	12
2.9	Anzahl der Unfälle über der Art der Ursache	13
2.10	Beispiel von einem FFT (Zeitbereich und Frequenzbereich)[NTI-Audio]	17
2.11	Achsenrichtung in Sensorframe sowie in Bikeframe [Schnee u. a., 2020] .	20
2.12	Entscheidungsbaum [Schnee u. a., 2021]	21
2.13	Beispiel eines Fahrrads beim Umkippen mit einer aufrechten Postion und mit einer Neigung	23
2.14	Kalibrierungsmodul - Nachkaklibrierung deaktiviert	24
2.15	Testfahrt ohne GroundHit - Full View	24
2.16	Testfahrt ohne GroundHit - auf die Energie gezoomt	25
2.17	Crash mit GroundHit - ID 2806 - Full View	26
2.18	Crash mit GroundHit - ID 2806 - auf die Energie gezoomt	26
2.19	Crash mit CollisionHit - ID 2546 - Fullview	27
2.20	Crash mit CollisionHit - ID 2546 - gezoomt	28
3.1	Spiegelinstitute - Umfrage - Pocket-Mode	29
3.2	Liste der Use- und Edgecases mit der erwarteten Reaktion des Algorithmus’	31
3.3	Beispiel: Beschleunigungssignal beim Laufen (**x-Achse 0-25 Sekunden, y-Achse (-2500 - 0))	32
3.4	Testbeispiel - Lauferkennung - Peaks Zähler	33
3.5	Testbeispiel - Lauferkennung - Sinussignalgenerator - Spezifikationen .	34
3.6	Testbeispiel - Lauferkennung - Sinussignal	34
3.7	Skizze eines einfaches ideales Signal sowie eines komplexeres Signal .	36
3.8	Testbeispiel - Frequenzbasierte Lauferkennung - Sinussignal	38

3.9	Testbeispiel - Frequenzbasierte Lauferkennung - Ausgabe des Spektrum- Analyzer und seine Spezifikationen	38
3.10	Testbeispiel - frequenzbasierte Lauferkennung - Ausgabe des Spektrum- Analyzer im Bereich zwischen 0-0.5 kHz	39
3.11	Testbeispiel - Frequenzbasierte Lauferkennung - Ausgabe des Spektrum- Analyzer mit anderen Spezifikationen	39
3.12	Testbeispiel - Frequenzbasierte Lauferkennung - FFT	41
3.13	Ablaufschema des Testmodells	42
3.14	Das Ergebnis der FFT - Spiegelung entfernt und Beträge	43
3.15	verschiedene Signale ($f = 23, 8; f = 47, 8; f = 1, 1$) mit deren Summe f_g	43
3.16	Lauferkennung - Frequenzbasierte - FFT - Echtmodell	44
3.17	Ein- und Ausgänge des Entscheidungsskripts	45
3.18	Entscheidungsbaum des Laufens	46
3.19	Abgeschnittene Teile eines Beispieldesignals	48
3.20	Beinpositionen während einer Fahrt und gestreckt	50
3.21	Beispiel der exportierten Tabelle mit der neuen Spalte	52
3.22	Die definierte Labels	53
A.1	Entscheidungsskript durch mehrere Kriterien (Frequenz und Geschwin- digkeit)	xxiii

Tabellenverzeichnis

3.1 Ausgangsmöglichkeiten der Entscheidungsfunktion	45
---	----

Abkürzungsverzeichnis

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

Symbolverzeichnis

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

1 Einleitung

Jedes Jahr wird das Leben von etwa 1,3 Millionen Menschen durch einen Verkehrsunfall beendet. Zwischen 20 und 50 Millionen weitere Menschen erleiden nicht-tödliche Verletzungen, wobei viele infolge ihrer Verletzung eine Behinderung erleiden. 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.[health organization, 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 % von allen zugelassenen Kraftfahrzeugen in Deutschland [Hädrich, 2012].

Motorradfahrer stellen im Straßenverkehr eine besonders gefährdete Gruppe an Verkehrsteilnehmern dar. Studien aus dem Jahr 2014 zeigten, dass die Wahrscheinlichkeit in der USA ein Motorradfahrer bei einem Unfall zu sterben, 27-mal höher ist als die eines Autofahrers, und dass Verletzungen sechsmal so wahrscheinlich sind [NHTSA]. Verzögerungen bei der Erkennung und Versorgung der an einem Verkehrsunfall beteiligten Personen erhöhen die Schwere der Verletzungen. Die Versorgung von Verletzungen nach einem Unfall ist äußerst zeitkritisch. Verzögerungen von Minuten können über Leben und Tod entscheiden. Die Verbesserung der Versorgung nach einem Unfall erfordert die Sicherstellung des Zugangs zu rechtzeitiger präklinischer Versorgung und die Verbesserung der Qualität sowohl der präklinischen als auch der stationären Versorgung.[health organization, 2022]

Die Reaktionszeit der Rettung spielt eine besonders große Rolle dabei, Leben zu retten. -; Unfallerkennungsalgorithmus;

-; Automatische Unfallerkennung

-; kürzere Reaktionszeit; Genauere Informationen mitgeteilt; usw.

-; kurze Versorgungszeit -; Leben retten!!

- Geschwindigkeitsüberschreitung bei Motorrädern viel häufiger als bei Autos.

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

- Pocket-Mode
- Edgecases
- Mögliche Maßnahmen
-

2 Grundlagen

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

ODER:

In diesem Kapitel wird die Relevanz der vorliegenden Arbeit erörtert.

2.1 Unfall

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

2.1.1 Deffinition

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. Dieses Modell macht es leichter den Ablauf eines Unfalls zu verstehen und anschließend weitere Unfälle zu vermeiden. Durch eine Störung an den Fahrer beziehungsweise ans Fahrzeug ändert sich der Ablauf einer Fahrt (oder eines Unfalls), da Störungen die Reaktionszeit sowie Reaktionsart der Fahrer stark beeinflussen. Diese sind im Fall eines Unfalls von großer Bedeutung.

2.1.2 Zeitliche Phasen eines Unfalls

Nach dem zeitlichen Unfallverlauf werden folgende Unfallphasen unterschieden:

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

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

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

Beispiel

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

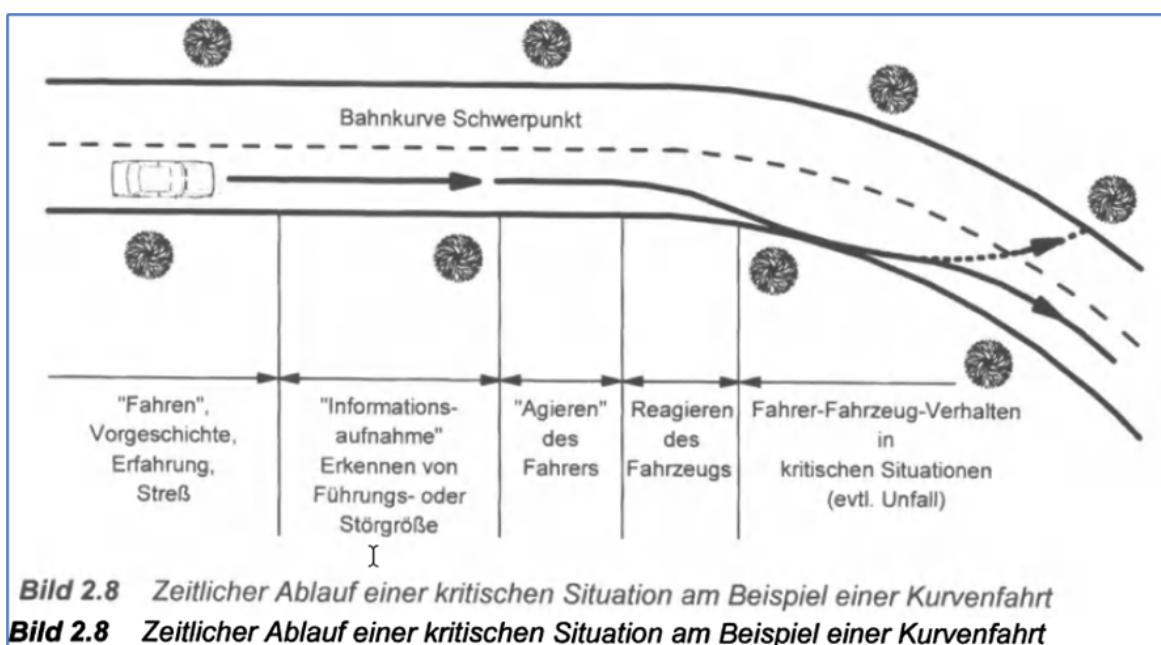


Abbildung 2.2: Beispiel der zeitlichen Phasen einer kritischen Fahrsituation [H. Appel, 2002]

2.1.3 Mechanik und Biomechanik des Unfalls

- Kinematik und Verletzungsbilder
- Zweiradfahrer (Bilder)

Kurvenfahrt

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

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

Kräfte am Fahrzeugschwerpunkt bei stationärer Kurvenfahrt:

Beim Einleiten einer Kurvenfahrt mit konstantem Bahnradius wirkt eine konstante Querbeschleunigung auf die Einheit "Fahrer-Maschine". Diese Querbeschleunigung bewirkt eine Seitenkraft im Gesamtschwerpunkt der Einheit "Fahrer-Maschine" und wird wie folgt berechnet:

$$a_q = \frac{v^2}{R} = \frac{1}{D} \quad (2.1)$$

Der Schrägwinkel lässt sich wie folgt berechnet:

$$\begin{aligned} \tan(\lambda) &= \frac{F_q}{F_G} \\ \tan(\lambda) &= \frac{a_q}{g} \\ \lambda &= \arctan\left(\frac{a_q}{g}\right) \\ \lambda &= \arctan\left(\frac{F_a}{F_G}\right) \end{aligned} \quad (2.2)$$

Die Abbildung 2.3 zeigt die wirkenden Kräften 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 zum Kippen zu bringen, wird das Fahrzeug die Kurve unter Schräglage durchfahren. Dadurch verschiebt sich der Schwerpunkt der Einheit zur Kurvenseite und wird ein Moment der Gewichtskraft um den Reifenaufstandspunkt resultiert, welches dem Moment der Fliehkraft entgegen wirkt und wieder ein Gleichgewicht sichert.

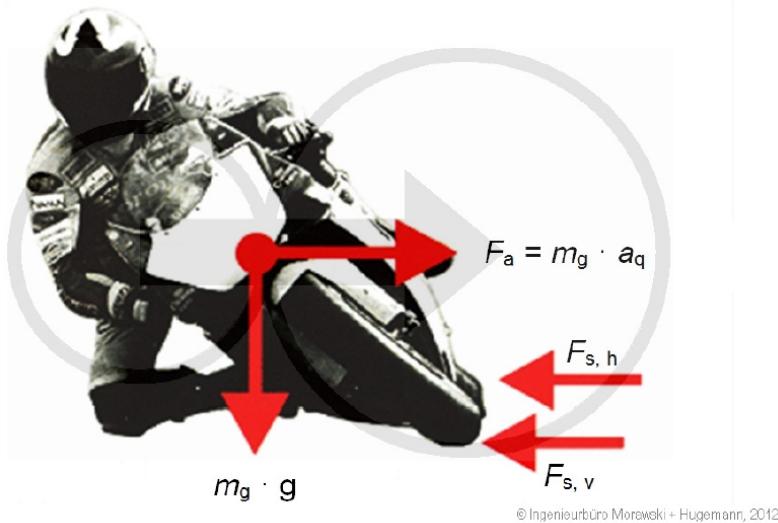


Abb. 3-2: Vereinfachte Darstellung der Kräfte bei stationärer Kurvenfahrt [ECK10a]

Abbildung 2.3: Vereinfachte Darstellung der Kräfte bei stationärer Kurvenfahrt (Abb. 3-2)

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 zu durchfahren, vgl. (Abbildung 2.4).

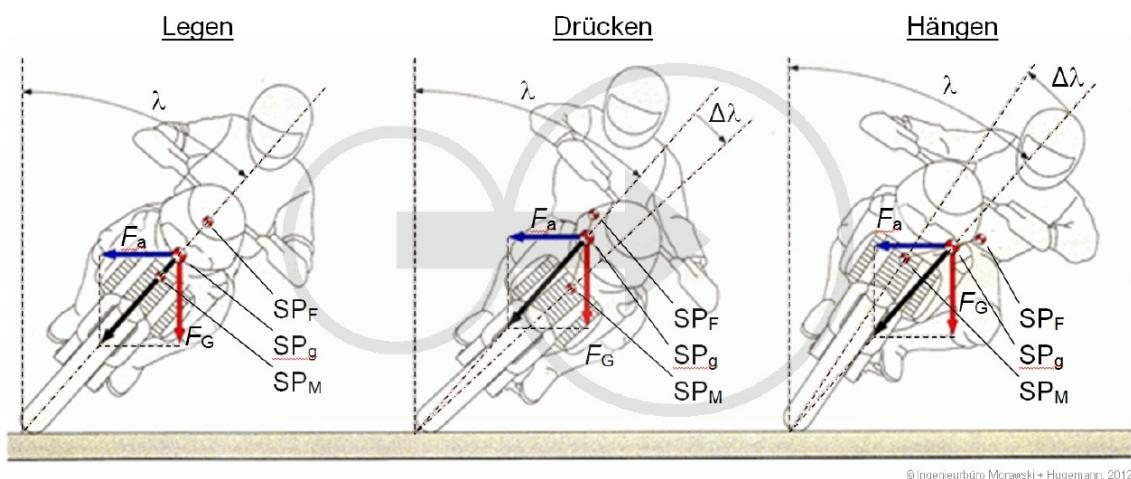


Abb. 3-6: Verschiedene Kurventechniken bei gleicher Geschwindigkeit [COC05]

Abbildung 2.4: Verschiedene Kurventechniken bei gleicher Geschwindigkeit (Abb. 3-6)

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

Zusammenhang des Grips auf der Straße. Es ist durchaus vorstellbar, dass mit zunehmender Schräglage die Reifenaufstandsfläche (beziehungsweise Reifenlatsch) abnimmt.

Fahrer von Renn- und Supersport-Maschinen bedienen sich der Tatsache des sich verändernden Reifenlatsches und „Hängen“ sich während der Kurvenfahrt von der Maschine. In diesem Fall liegt der Schwerpunkt des Fahrers SPF unterhalb des Schwerpunktes der Maschine SPM, so dass das Motorrad die Kurve mit deutlich geringerer Schräglage durchfahren kann, als beim „Legen“, (Abbildung 2.4). Gegenüber der anderen zwei Schräglagen kann durch das „Hängen“ mit gleichen Schrägwinkel höhere Seitenkräfte übertragen werden, da hier die Radaufstandsfläche größer ist.

2.1.4 Statistische Zahlen über Unfälle

Das baden-württembergische Verkehrsministerium stellt ein Portal zur Verfügung, über das einzelne Verkehrsmessstellen abgefragt werden können. Die Messstationen wurden nach zwei Kriterien ausgewählt. Einerseits müssen Unfallschwerpunkte in unmittelbarer Nähe zu Messstationen sein, um zuverlässige Aussagen über Verkehr und Störstellen zu treffen. Andererseits muss darauf geachtet werden, dass es keine Abzweigungen zwischen der Messstation und dem Unfallschwerpunkt gibt, da sonst das richtige Verkehrsaufkommen nicht erfasst werden kann.

Die Abbildung 2.5 stellt dar, wie oft die Geschwindigkeit von Motorradfahrer sowie von PKW-Fahrer an verschiedenen Stationen überschritten wurde. Aus der Abbildung 2.5 ist deutlich zu erkennen, dass an sechs der sieben Stationen das Geschwindigkeitslimit regelmäßig überschritten wird. Die Motorradfahrer missachten die Geschwindigkeitsbegrenzung häufiger als die PKW-Fahrer. [d. Maire, 2020]

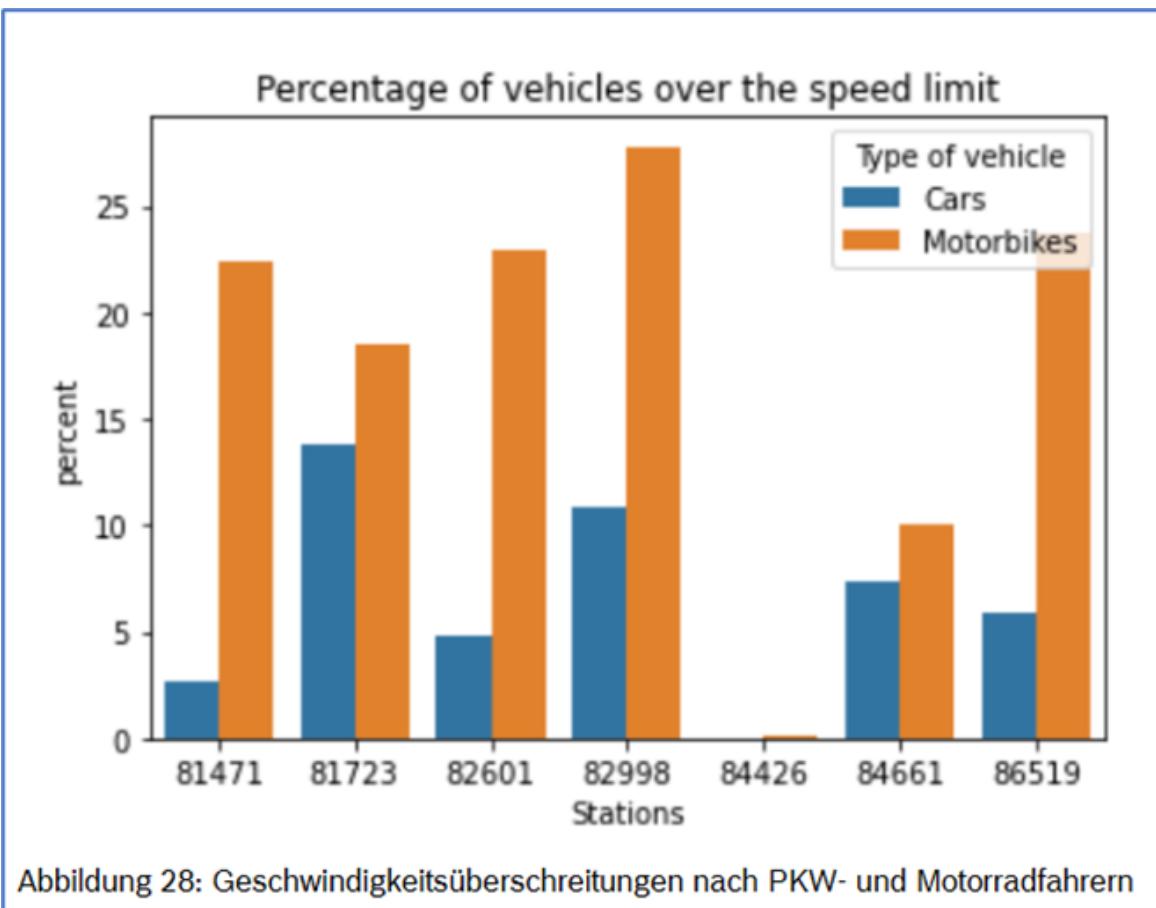


Abbildung 2.5: Geschwindigkeitsüberschreitungen nach PKW- und Motorradfahrer an 7 Messstationen aus der VSS-Datenbank [d. Maire, 2020]

Die GIDAS-Daten verfügen über die Ausgangsgeschwindigkeit des Motorradfahrers, was die Ermittlung des Einflusses dieser Geschwindigkeit auf die Unfallfolgen ermöglicht. In der Abbildung 2.6 sind die Unfallschwere nach Motorradgeschwindigkeit als Histogramm dargestellt. Die Unfälle werden dabei nach Unfallschwere (leicht verletzt, schwer verletzt, tödlich verwundet) unterteilt. Die gestrichelten Linien repräsentieren die Mittelwerte der Histogramme. Die Grafik zeigt, dass die Verletzungsschwere stark von der Geschwindigkeit abhängig ist. Bei einer 0km/h gibt es keine tödlichen Unfälle, wobei die Unfälle bei einer Geschwindigkeit von 100 fast immer mit einer schweren Verletzung oder tödlichen Verkehrsteilnehmer enden. Die hohe Geschwindigkeit kommt immer mit hohen Kräfte zusammen, welche bei einem Unfall dem Fahrer bewirken. Im Fall eines Motorradfahrers ist das besonders wichtig zu betrachten, da diese Kräfte dem Fahrer direkt übertragen werden. [d. Maire, 2020]

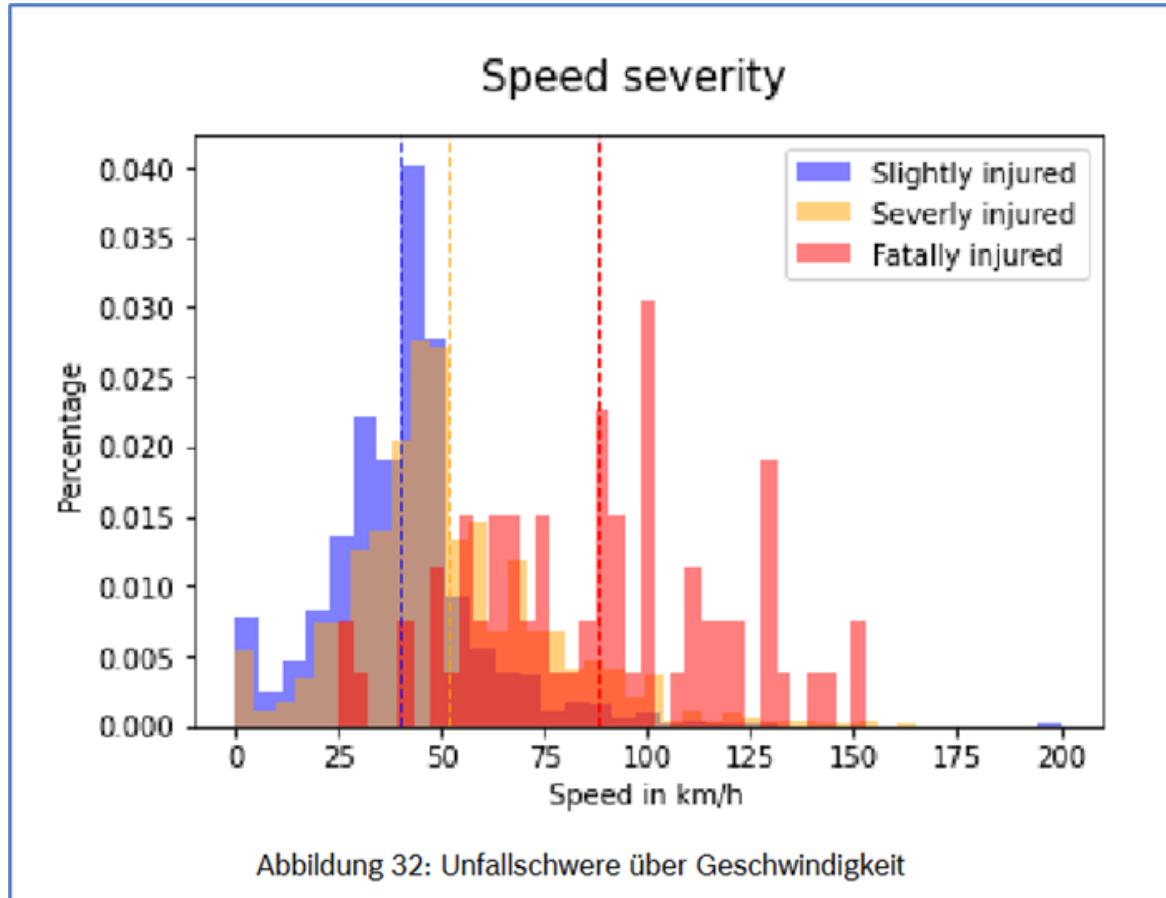


Abbildung 32: Unfallschwere über Geschwindigkeit

Abbildung 2.6: *Unfallschwere über Geschwindigkeit aus der GAIDA-Datenbank/d. Maire, 2020]*

Youtube-Statistik

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

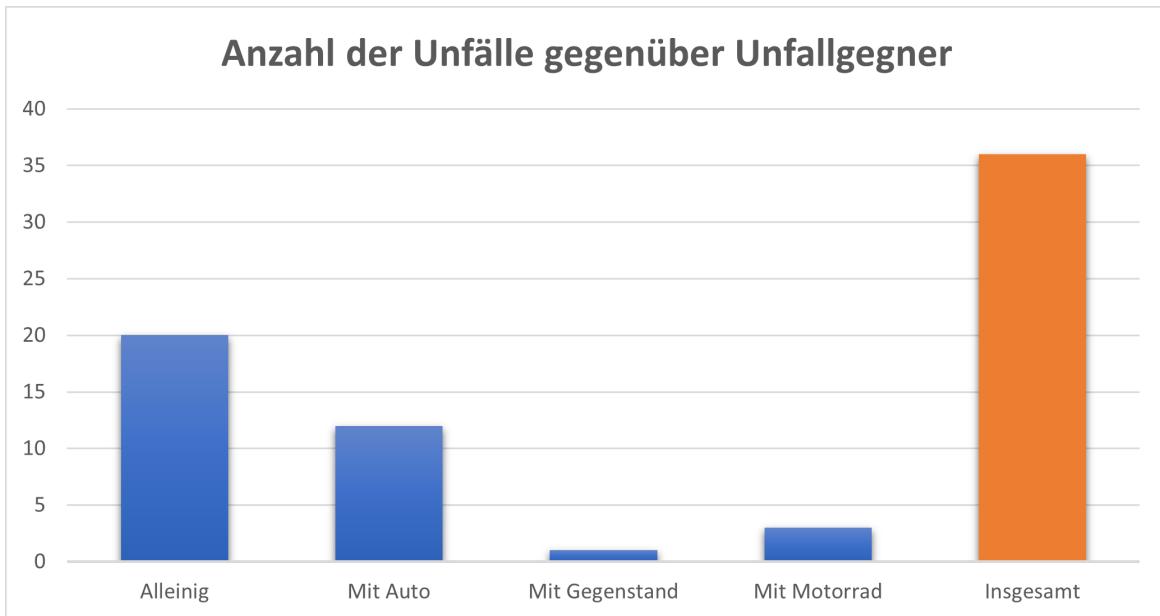


Abbildung 2.7: *Anzahl der Unfälle gegenüber Unfallgegner*

In der Abbildung 2.7 stellt die Grafik die Anzahl der Motorradunfälle gegenüber der Unfallgegner dar. Es wurde hier zwischen alleiniger Unfall, Unfall mit einem Gegner (Auto, Motorrad) oder Unfall wegen eines Gegenstands unterschieden. Diese Unterscheidung ist wegen der Verhaltensunterschied während eines Unfalls. Von insgesamt 36 Unfälle waren die alleinige Unfälle am meisten gefolgt von den Unfällen mit einem Auto. Die Unfälle mit einem anderen Motorrad oder wegen eines Hindernis sind am wenigsten. Unter alleinige Unfälle werden die zwei Szenarien (An einer Kurve rutsche und Kontrolle verloren) am meisten aufgetreten. Das kann sehr gut in der Abbildung 2.9 sichtbar sein. Das Szenario, in dem das Motorrad an ein Auto von hinten angestoßen hat, hatte die dritte Stelle besetzt. Die Hauptgründe der Unfällen mit einem Gegner waren v.A. die hohe Geschwindigkeit und das schlechte Wetter (z.B. Nässe, Schnee), was zu Schwierigkeiten geführt hat, das Motorrad in kritischen Situationen zu kontrollieren.

Die Abbildung 2.8 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, z.B. durch langsamer fahren oder mehr Abstand mit den anderen Verkehrsteilnehmer.

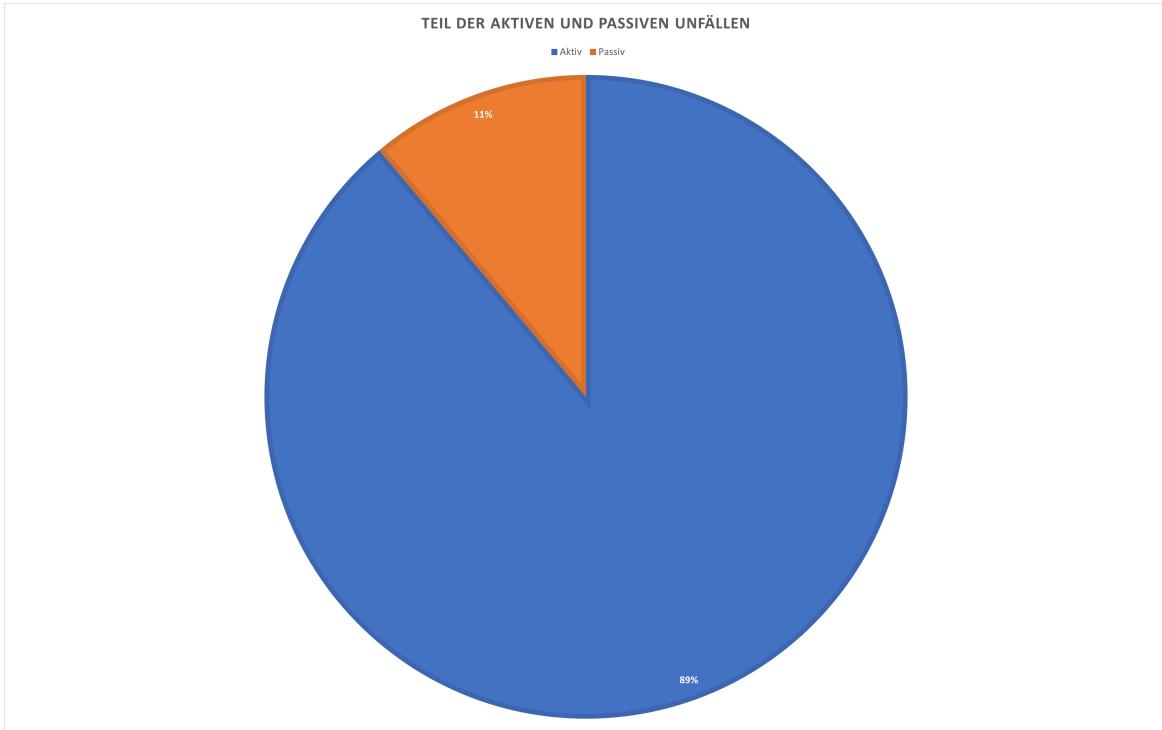


Abbildung 2.8: Anteil der aktiven sowie passiven Unfälle

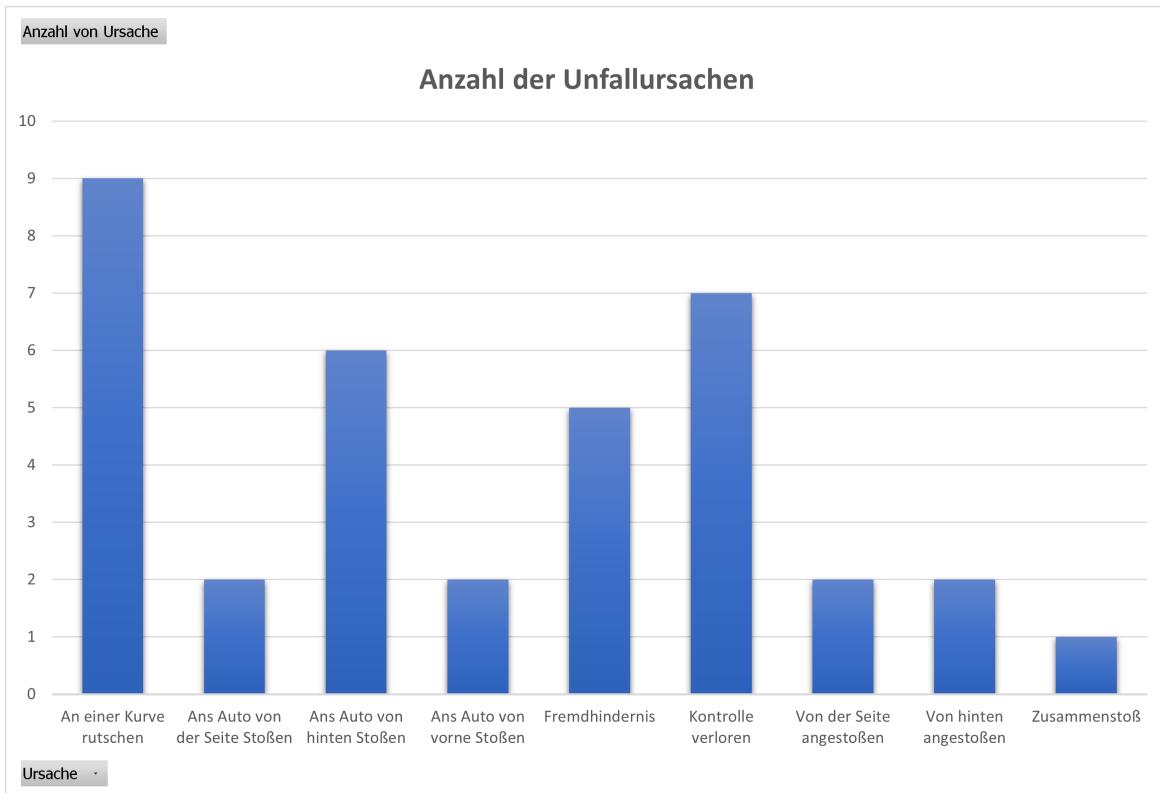


Abbildung 2.9: Anzahl der Unfälle über der Art der Ursache

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. Bewegungssensoren sind nützlich zum Überwachen von Gerätebewegungen wie Neigung, Schütteln, Drehung oder Schwingen. Die Bewegung spiegelt normalerweise die direkte Benutzereingabe wider, kann aber auch die physische Umgebung widerspiegeln, in der sich das Gerät befindet (z.B. Das Smartphone bewegt sich mit der Person, die es am Körper hat und sich selbst bewegt).[Developers, 2022]

Beschleunigungssensoren

- Beispieldaten

Der Beschleunigungssensor ist ein elektromechanisches Gerät, das die Beschleunigungskraft misst, die durch Bewegung, Schwerkraft oder Vibration verursacht wird. Diese Kräfte können statisch wie Schwerkraft, dynamische Bewegungsempfindungen

oder Vibrationen sein. Mathematisch gesehen ist die Beschleunigung ein Maß für die zeitliche Geschwindigkeitsänderung (beziehungsweise Geschwindigkeit geteilt durch die Zeit). 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 ein besseres Anzeigerlebnis erzielen. [Sharma, 2020]

Der Beschleunigungssensor im mobilen Gerät liefert die XYZ-Koordinatenwerte, die zum Messen der Position und der Beschleunigung des Geräts verwendet werden. Die XYZ-Koordinate stellt die Richtung und Position des Geräts dar, an dem eine Beschleunigung aufgetreten ist. Die Drehrichtung und -position werden mit Gyroskopsensoren gemessen. Die vom Gerät bereitgestellten Beschleunigungsmesserwerte enthalten normalerweise auch die Schwerkraft. Das Signal des Beschleunigungssensor wird in die Tief-/Hochpassfilter geleitet, um das Ergebnis basierend auf der verwendeten Anwendung zu verfeinern. [Sathish, 2021]

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

Im Allgemeinen ist der Beschleunigungssensor ein guter Sensor, wenn Sie die Bewegung des Geräts überwachen. [Developers, 2022]

Gyroskop

- Beispieldaten

Gyroskop ist ein Gerät, das ein sich schnell drehendes Rad oder einen umlaufenden Lichtstrahl enthält. Gyroskop wird verwendet, um die Abweichung eines Objekts von seiner gewünschten Ausrichtung zu erkennen. Gyroskope werden zur automatischen Lenkung und zur Korrektur der Dreh- und Nickbewegung in Marschflugkörpern und ballistischen Flugkörpern verwendet. [Rogers] Der Gyroskopsensor im MEMS ist winzig (zwischen 1 und 100 Mikrometer, die Größe eines menschlichen Haars). 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]

Global Positioning System (GPS)

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

2.2.2 Matlab/Simulink

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

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

Simulink und LabVIEW

LabVIEW ist eine von "National Instruments" entwickelte Software. Sie wird häufig von Ingenieuren, Wissenschaftlern und Studenten für die Datenerfassung, Instrumentsteuerung 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] [Kasanakoglu, 2015]

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

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

App-Entwicklung

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

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

C/C++-Coder

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

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

2.3 Mathematische Grundlagen

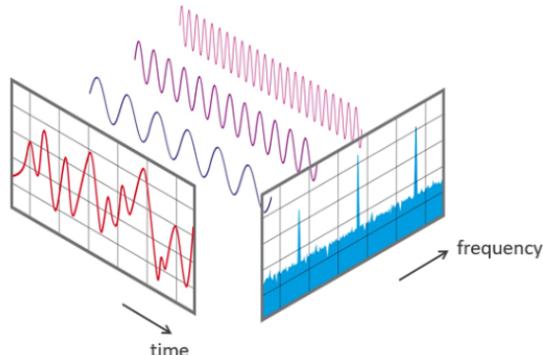
In dieser Arbeit wird eine FFT eingesetzt, für welche das folgende Hintergrundwissen zum Verständnis benötigt wird.

2.3.1 FFT

[NTI-Audio][Weisstein]

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

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



View of a signal in the time and frequency domain

Abbildung 2.10: Beispiel von einem FFT (Zeitbereich und Frequenzbereich)[NTI-Audio]

Schritt für Schritt

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

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

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

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

$$f_n = \frac{f_s}{2} \quad (2.3)$$

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

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

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

Frequenzauflösung: d_f gibt den Frequenzabstand zwischen zwei Messergebnissen an.

$$d_f = \frac{f_s}{B_L} = \frac{1}{D} \quad (2.5)$$

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

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

Spiegelfrequenzen

Wird die Nyquist-Frequenz (Gleichung 2.3) 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.

2.4 Methoden der Softwareentwicklung

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

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

2.5 Unfallerkennungsalgorithmus

In diesem Teil wird der Unfallerkennungsalgorithmus erläutert und näher betrachtet. Der Algorithmus ist sowohl für Fahrräder als auch für Motorräder entwickelt und bearbeitet die Signale der Beschleunigungssensor sowie Gyroskope im Smartphone und die über Gps gemessene Geschwindigkeit. In dem Unfallerkennungsalgorithmus werden drei Hauptkriterien (CollisionHit, GroundHit und TipOver) berücksichtigt.

Die Modellierung der Merkmale GroundHit und Collision erfordert weitere unabhängige vorverarbeitete Signale. Zu diesem Zweck wird der ANOVA-Ansatz (Analysis of Variance) (Dalgaard, 2008) verwendet, um verschiedene statistische Eigenschaften (z. B. Mittelwert, Standardabweichung, Varianz, Extrema und Integral Figo et al., 2010) über verschiedene Fenstergrößen des zu analysieren IMU-Beschleunigungsdaten. Der ANOVA-Ansatz erfordert, dass die vorverarbeiteten Daten normalverteilt sind. Der Anderson-Darling-Test wird auch angewandt, um diese notwendige Bedingung zu überprüfen. Das Ergebnis der ANOVA-Analyse liefert die spezifische Energie als optimalen Indikator unter denen aus den untersuchten statistischen Ansätzen zur Klassifizierung der Kollisions- und Bodentreffer-Ereignisse wie folgt:

$$\Delta_{xy,u}(i) = \left(\int_{i-u}^i a_{Bf,x}(k) dk \right)^2 + \left(\int_{i-u}^i a_{Bf,y}(k) dk \right)^2 \quad (2.6)$$

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

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

2.5.1 Kalibrierung

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

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

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

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

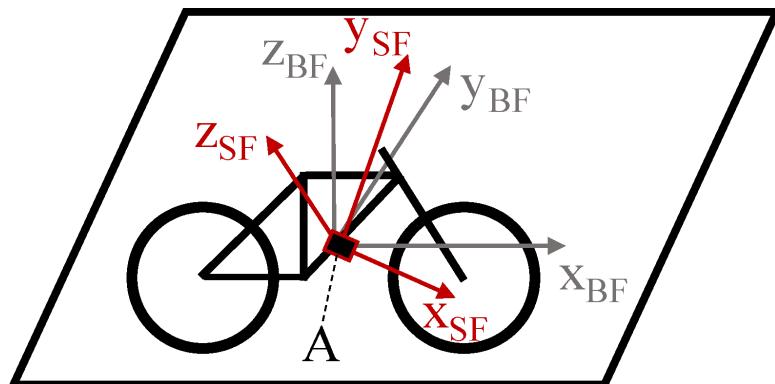


Abbildung 2.11: Achsenrichtung in Sensorframe sowie in Bikeframe [Schnee u. a., 2020]

2.5.2 Übersicht der bereits erkennbaren Unfälle

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

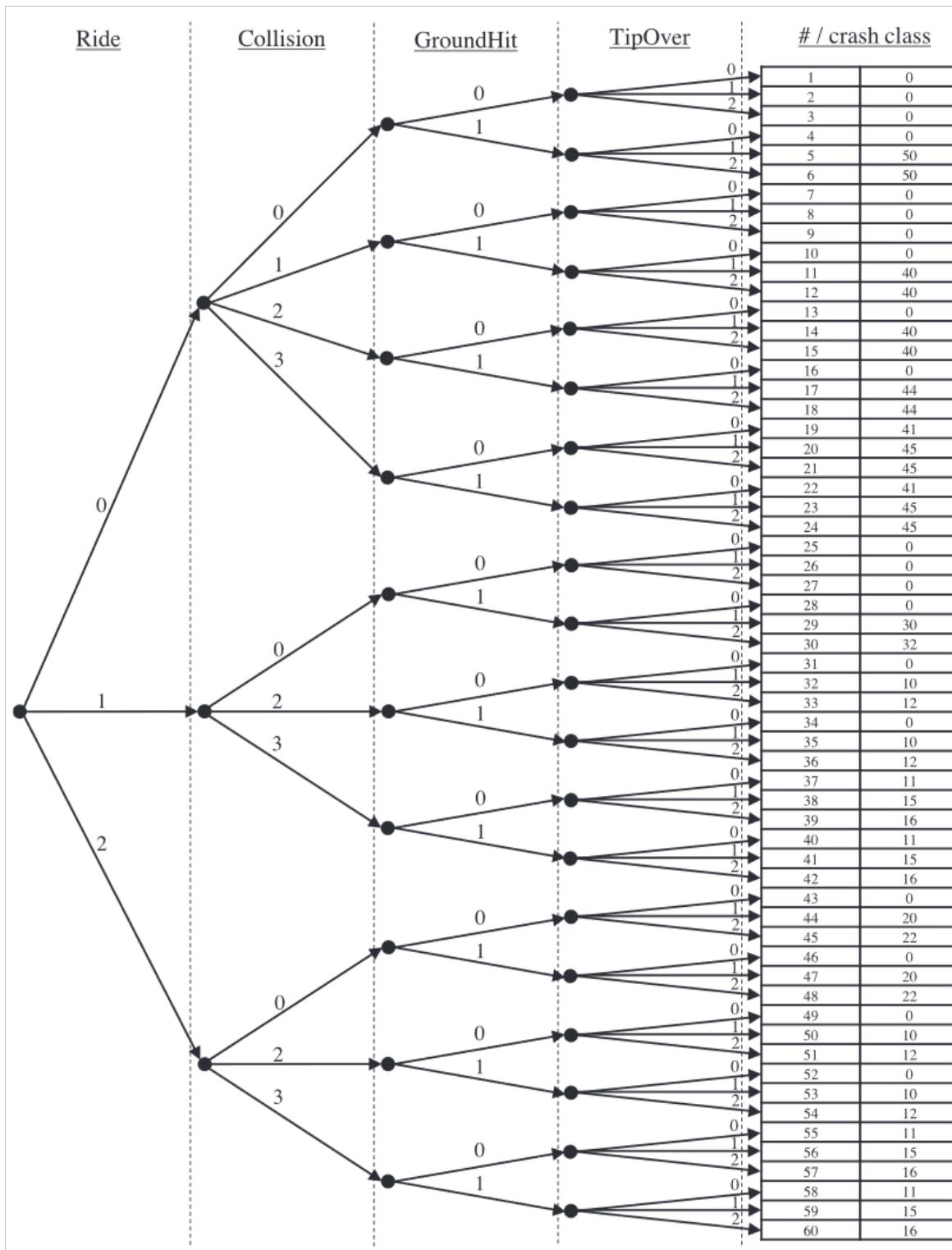


Abbildung 2.12: Entscheidungsbaum [Schnee u. a., 2021]

2.5.3 TipOver

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

Bei Motorradunfällen auf Straßen, die sich im Anschluss an eine Kurve ereignen, stellt sich immer wieder die Frage nach der maximalen Geschwindigkeit, mit der die Kurve auf einem Motorrad durchfahren werden konnte. Um das Motorrad durch die Kurve zu bewegen, muss sich der Fahrer mit seiner Maschine in die Kurve legen. Während einer Fahrt in der Kurve wirken zwei Kräfte (F_q) und (F_G) am Motorrad (Abbildung 2.3). Wenn der Kraft (F_q) größer als (F_G) tritt, kippt das Motorrad um. Im normalen Fall soll

$$F_G \geq F_q$$

immer gültig sein.

D.h.

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

Beim Einsetzen in der Gleichung 2.2 ergibt sich der maximale zulässige Winkelwert:

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

D.h. Bei einer Neigung von über 45° , sollte das Motorrad umkippen.

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

2.5.4 GroundHit

Dieses Modell dient dazu, einen Schlag zu erkennen, wenn das Motorrad am Boden ankommt. Dieses Modell geht von der Energie aus der Gleichung 2.6 aus. Dieses Modell hat einige Spezifikationen, die die verschiedene Szenarien abdeckt. Zwei typische Szenarien sind in der Abbildung 2.13 dargestellt. Die Abbildung 2.13a zeigt der Fall, wenn ein Fahrrad nach einer ursprünglichen aufrechten Position umkippt, und die Seite 23 das Umkippen eines Fahrrads mit einer ursprünglichen Neigung (z.B. in einer Kurve). Die Energie vom GroundHit im ersten Fall (Abbildung 2.13a) ist deutlich größer, deswegen wird der Schwellwert nach dem Neigungswinkel angepasst. Je größer die Motorradneigung ist, desto kleiner ist der GroundHit-Schwellwert.

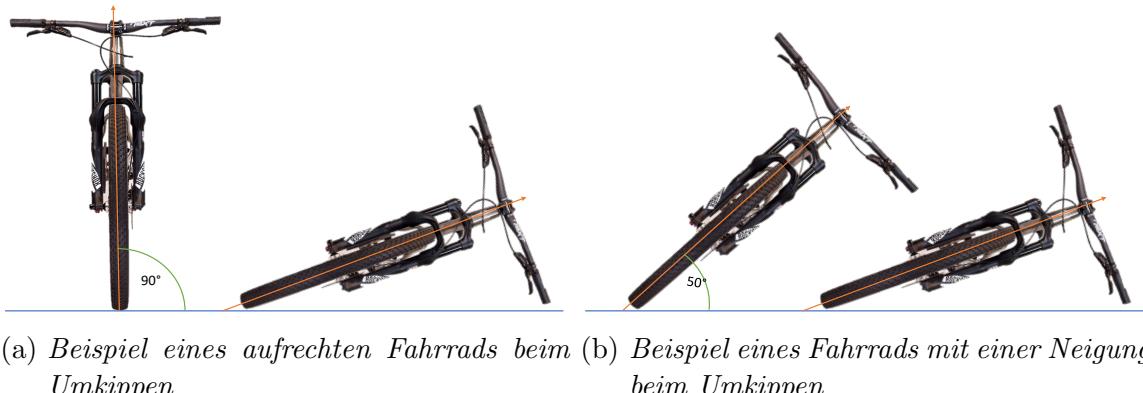


Abbildung 2.13: Beispiel eines Fahrrads beim Umkippen mit einer aufrechten Position und mit einer Neigung

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

Beispiele 1: Kein GroundHit

In diesem Beispiel ist eine Testfahrt ohne GroundHit durchgeführt und schließlich analysiert. Nach der Fahrt und bei späterer Simulation wurde entdeckt, dass die Kalibrierung nicht richtig war und musste manuell im Code angepasst werden. Eine Rotationsmatrix, die in der Regel durch die Kalibrierung gerechnet und umgesetzt wird, wurde dazu manuell erstellt und verwendet. Um eine falsche Nachkalibrierung zu verhindern, wurde der zuständige Teil deaktiviert (Abbildung 2.14). Es ist wichtig zu wissen, dass die Position des Geräts sich während der Fahrt kaum verändert hat.

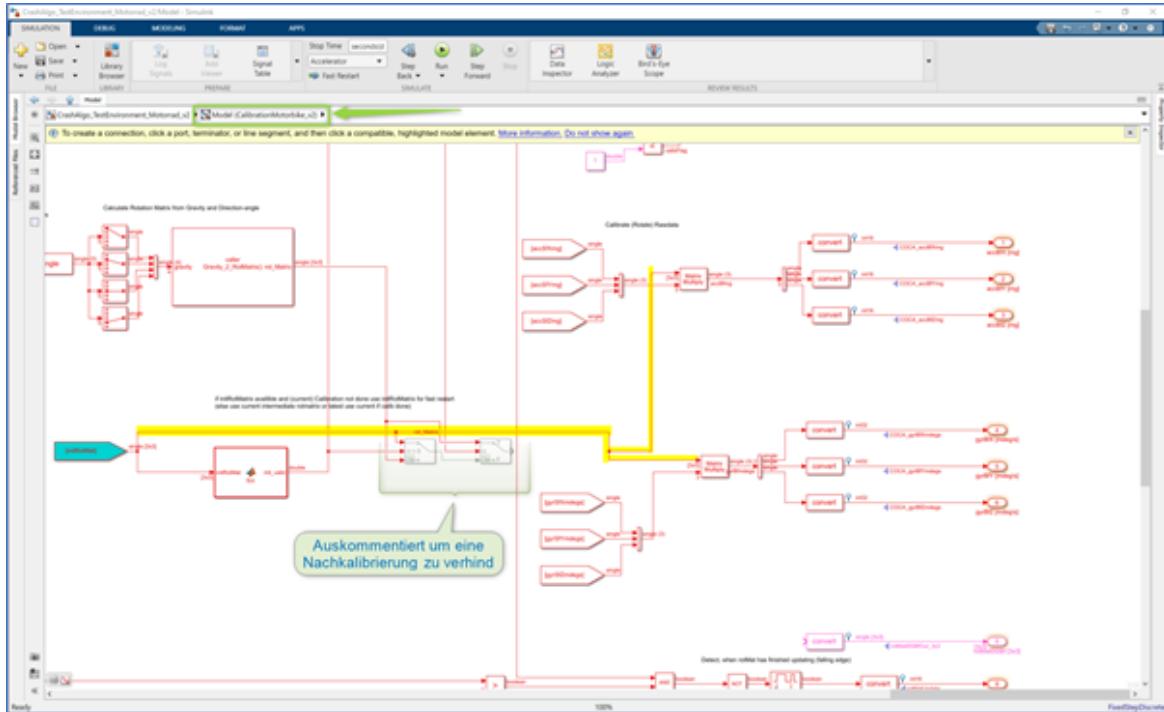


Abbildung 2.14: Kalibrierungsmodul - Nachkaklibrierung deaktiviert

In der Abbildung 2.15 und Abbildung 2.16 ist der Fall vorgestellt, in dem kein Groundhit erkannt wurde. Während dieser Fahrt fand kein Unfall statt.

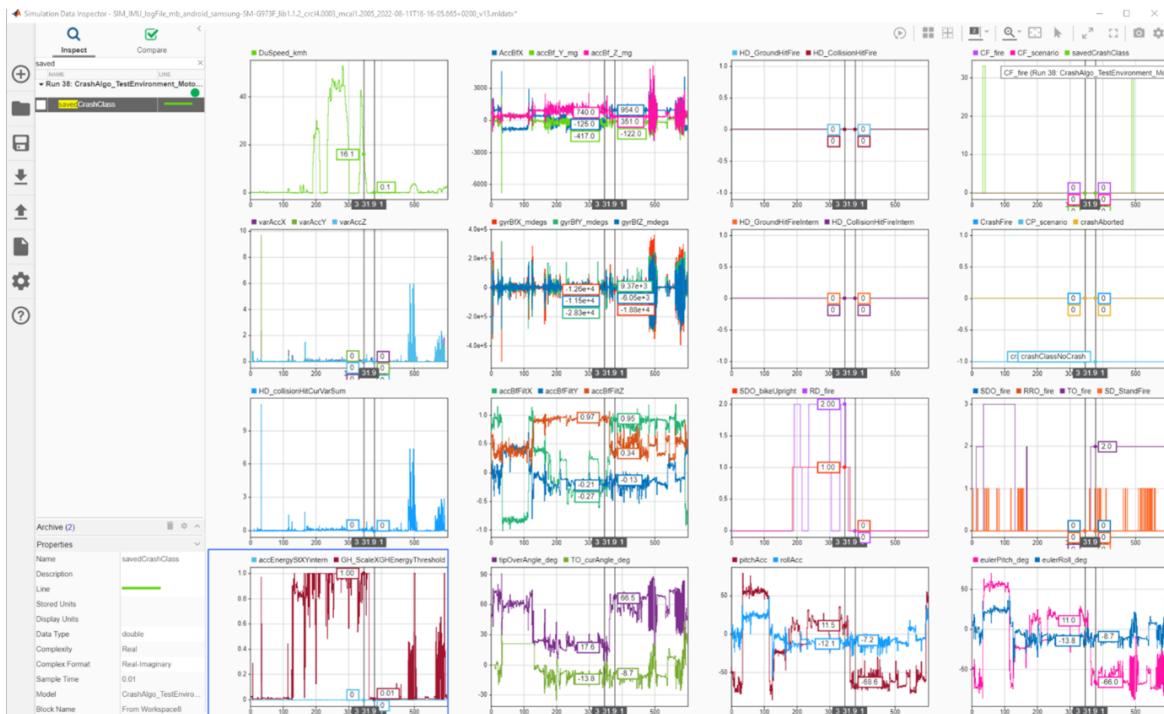


Abbildung 2.15: Testfahrt ohne GroundHit - Full View

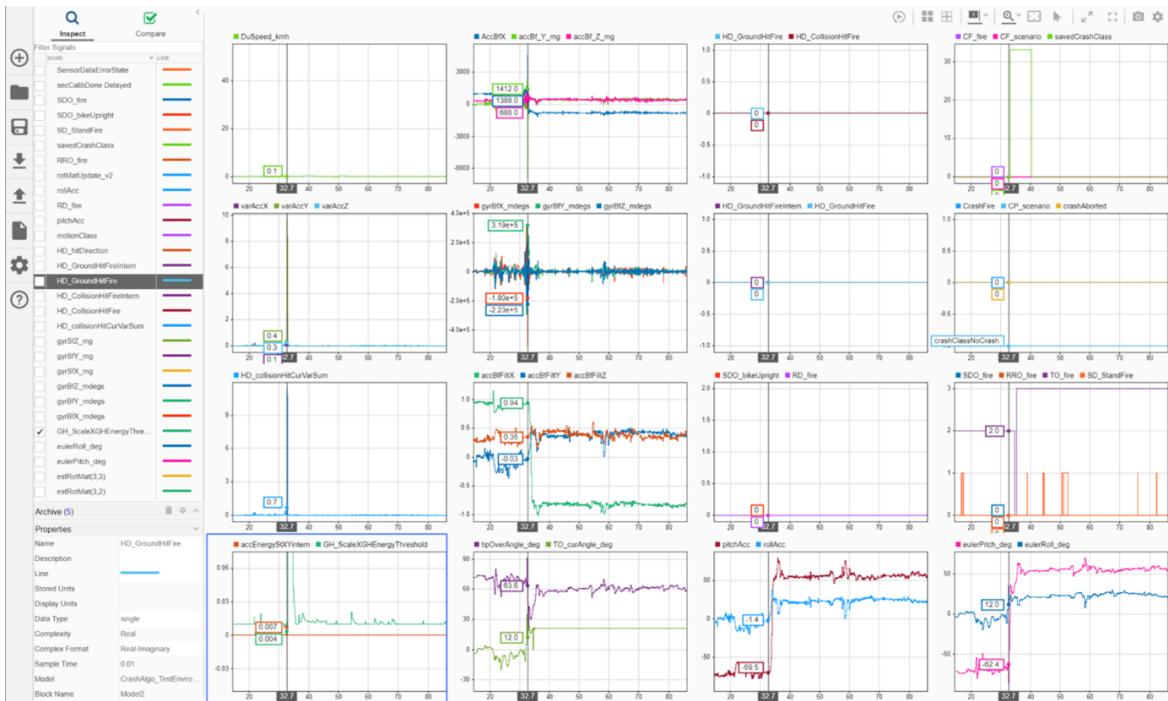


Abbildung 2.16: Testfahrt ohne GroundHit - auf die Energie gezoomt

In der Abbildung 2.16 ist zum Zeitpunkt 32,7 s Folgendes abzulesen:

- $RideFire = 0$, keine Fahrt
- $accEnergyStXYintern = 0,007$
- $GH_{SaleXGHEnergythreshold} = 0,004$
- $TO_{curAngledeg} = 12$
- $eulerPitch_{deg}$ springt von ca. -60 auf ca. +60 (Das handy wurde wahrscheinlich um 180 gedreht. Das könnte auch durch die Acc-Daten bestätigt werden)

An dieser Stelle ist $accEnergyStXYintern > GH_{SaleXGHEnergythreshold}$, was eigentlich einen GroundHit-Alarm auslösen soll. (Während des Testens hat das Smartphone an der Stelle einen falsch-postiven Alarm erkannt) Grund dafür ist, dass die Kalibrierung nicht 100% richtig war. In den simulierten Daten wurde kein Unfall erkannt, da die Kalibrierung richtig war, und der TO-Winkel ($TO_{curAngledeg}$) nicht über 45 war.

Beispiele 2: Unfall mit GroundHit

In der Abbildung 2.17 sind die simulierten Daten einer echten Fahrt mit einem Unfall dargestellt. Es ist zu bemerken, dass die $GH_{SaleXGHEnergythreshold}$ von dem Winkel ($tipOverAngle_{deg}$) abhängig ist. Bei einem hohen Winkel sinkt die $SaleXGHEnergythreshold$ (die linke unterste Darstellung der Abbildung 2.17).

2 Grundlagen



Abbildung 2.17: Crash mit GroundHit - ID 2806 - Full View



Abbildung 2.18: Crash mit GroundHit - ID 2806 - auf die Energie gezoomt

In der Abbildung 2.18 sind die Daten eines Unfalls (ID: 2806) dargestellt und auf die

Unfallphase gezoomt. In der Abbildung ist zu erkennen, dass die $GH_{SaleXGHEnergythreshold}$ bei einem hohen Winkel ($tipOverAngledeg$) sinkt. An der Stelle wo die $accEnergyStXYintern > GH_{SaleXGHEnergythreshold}$ ist (ca. 591,35 s), ist der Winkel fast gleich 45 - \circ , dadurch löst ein Alarm der GroundHit aus. Das entspricht die Erwartungen.

2.5.5 CollisionHit

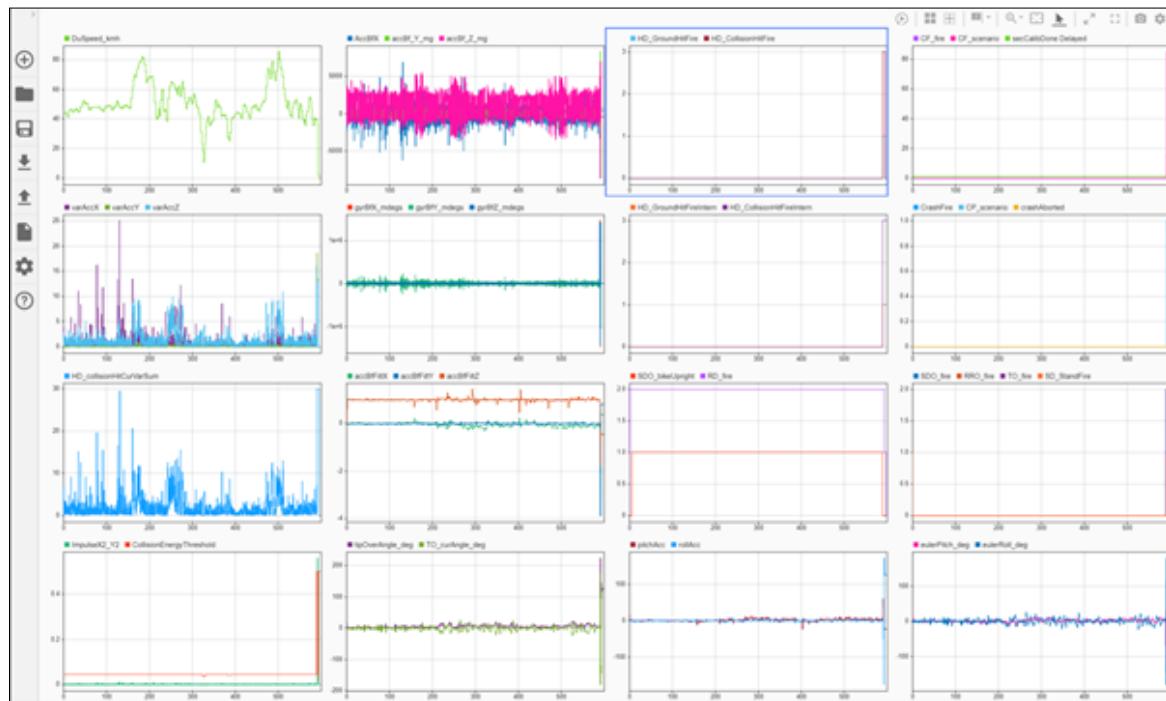


Abbildung 2.19: Crash mit CollisionHit - ID 2546 - Fullview

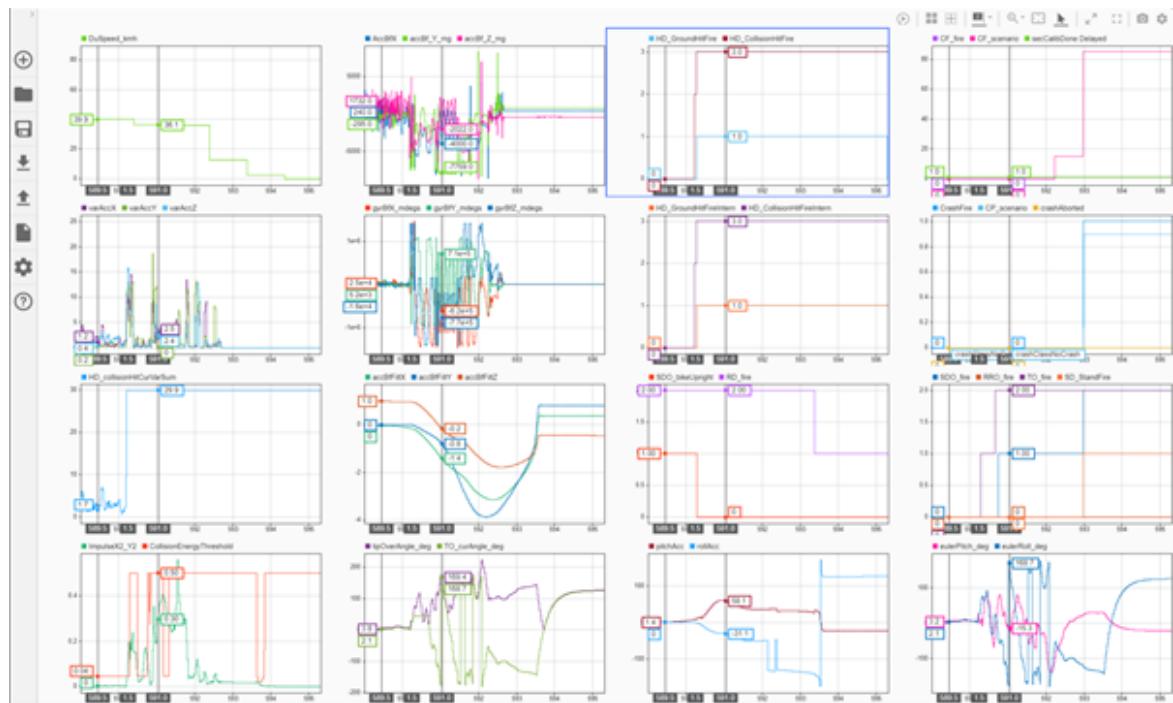


Abbildung 2.20: Crash mit CollisionHit - ID 2546 - gezoomt

3 Unfallerkennung im Pocket-Mode

Die Anzahl der App-Nutzer (Unfallerkennungsalgorithmus) war im November 2021 ca. 1190. Die Anzahl der Motorräder zum gleichen Zeit 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 Features abgedeckt werden.

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

Die zwei wichtigsten relevanten Fragen sind:

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

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

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

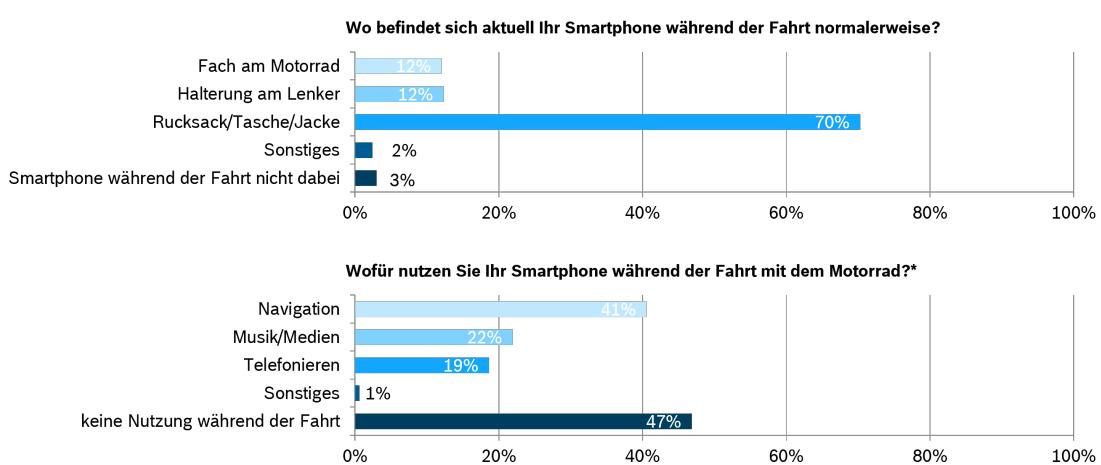


Abbildung 3.1: Spiegelinstitute - Umfrage - Pocket-Mode

Aus diesem Grund ist die Entwicklung einer Unfallerkennung im Pocket-Mode wichtig, wo das Smartphone nicht mehr unbedingt am Lenker befestigt werden muss. Die Weiterentwicklung der Unfallerkennung wird mit agilen Methoden erfolgt, Seite 19.

3.1 Kritische Szenarien/Fälle

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

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

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

A ID	B Name	C Beschreibung	D Erkennung durch den Algo		F Bemerkungen	G Geplante (mögliche) Maßnahmen
			Handy am Körper	Handy am Motorrad		
3 1	Oberkörper bewegen	Umdrehen, Nach hinten schauen, Lenker mit einer Hand halten	-	IO	Kein Einfluss auf das Handy	Keine
2		Nach vorne und hinten lehnen	IO	-	Kein GH, keine CH, Nur geringe Winkeländerung um die X-Y-Achsen	Keine
4		Seitliches Lehen (rechts und links)	-	IO	Kein Einfluss auf das Handy	Keine
5		Ab- und Aufsteigen	NIO	-	Bewegungsabhängig, enthält Winkeländerung (TO) und manchmal Groundhit. Falsch positiv bei GH. Kein GH -> kein Unfall	Erstmal Testen
6		Laufen	Handy am Körper	NIO	-	Lauferkennungsmodul einbauen und die Unfallerkennung während des Laufen deaktivieren
7		Handy in der Hand nehmen/nutzen	Eintippen, telefonieren, bewegen...ect.	NIO	-	Erkennen (durch phone-lifting-Funktion?) und Unfallerkennung deaktivieren
8		Wheelie fahren	Nur auf das Hinterrad fahren; Extremer Fall (Not intended use)	NIO	-	In AGB ausschließen
9		Auf der Motorradsitzbank stehen	Extremer Fall (Not intended use)	NIO	-	In AGB ausschließen
10		Auf dem Motorrad (Fußraste) stehen	Beim Fahren Körper dehnen/strecken	-	Kein Einfluss auf das Handy	Testen
11		An der Ampel stehen und Fuß runter	Fahren, dann (stark) bremsen und Fuß runter	IO	Kein GH, keine CH, Keine kritische Winkeländerung um die X-Y-Achsen	Keine
12		Normales Fahren	Beschleunigen, bremsen, Kurven fahren... ect.	IO	Im aktuellen Algo abgedeckt	Keine
13		Handy in der Tasche rutscht	Mit Winkeländerung	NIO	Fahren und Winkeländerung -> Unfall	Schnelles Nachkalibrierung oder in AGB bekannt machen (Handy befestigen)
14			Keine Winkeländerung (gleiche Position)	IO	Keine Winkeländerung -> kein Unfall	Testen ob GH oder CH erkannt werden
15				IO	Im aktuellen Algo abgedeckt	Keine
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						

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

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

3.2 Lauferkennung

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

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

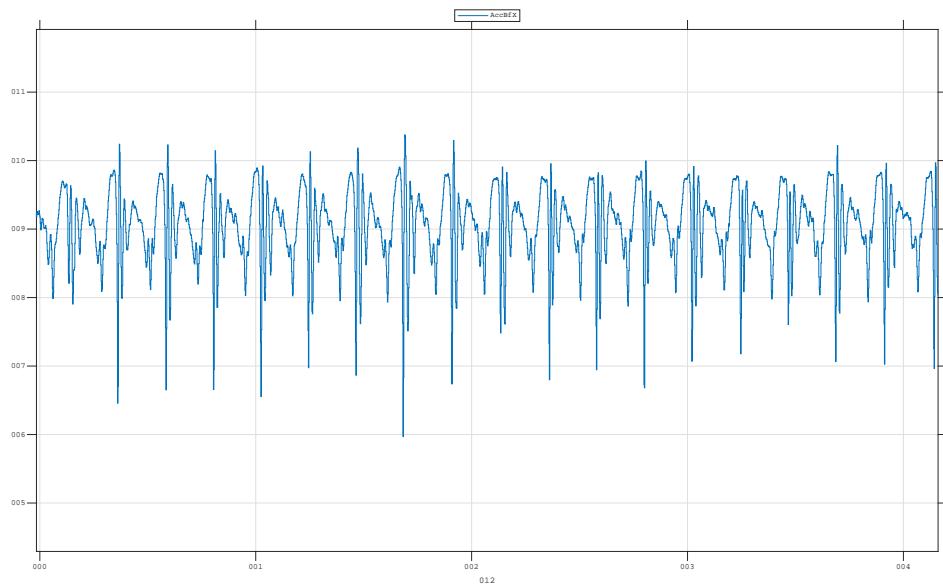


Abbildung 3.3: Beispiel: Beschleunigungssignal beim Laufen (**x-Achse 0-25 Sekunden, y-Achse (-2500 - 0))

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

3.2.1 Lauferkennung - Spitzenzähler

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

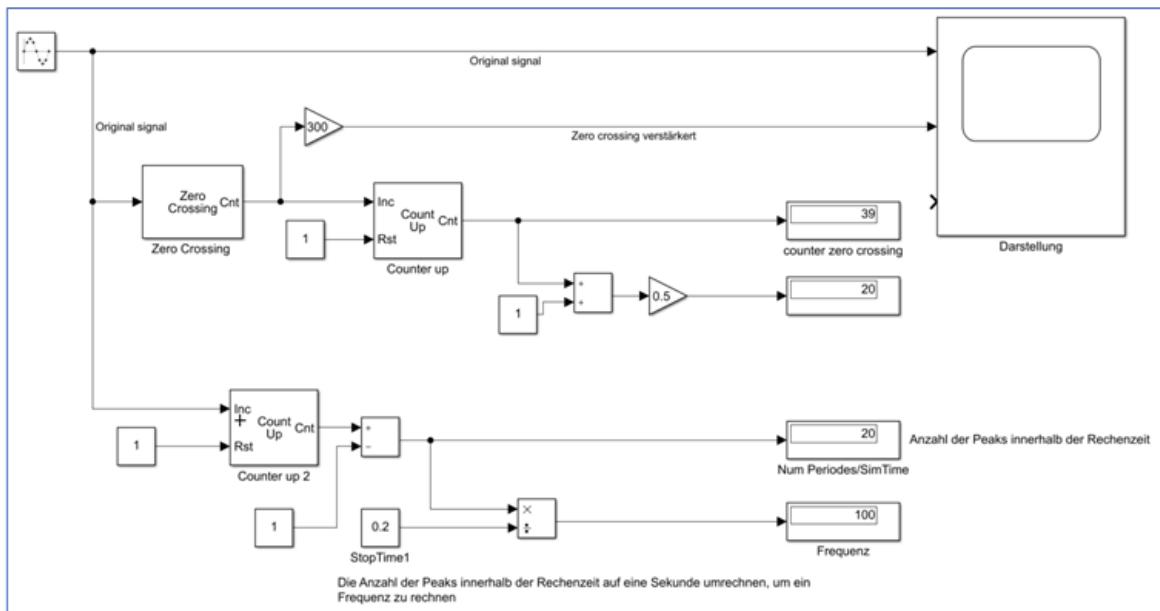


Abbildung 3.4: Testbeispiel - Lauferkennung - Peaks Zähler

Die Abbildung 3.5 zeigt die Spezifikationen des generierten Signals und die Abbildung 3.6 stellt mithilfe eines Scopes (Simulink-Block) das entsprechende Signal grafisch dar sowie wie oft das Signal die Nulllinie überschneidet (blau). Aus der Grafik ist die Anzahl der Peaks einfach zu ermitteln und diese beträgt in diesem Fall 100 Hz umgerechnet. Eine zeitliche Frequenz wird folgendes berechnet:

$$Freq_{Hz} = \frac{Freq_{(rad/sec)}}{2\pi}$$

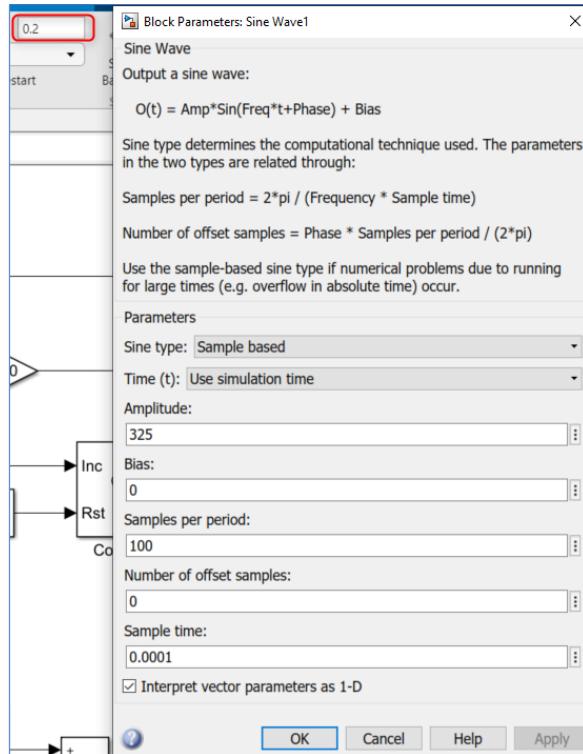


Abbildung 3.5: Testbeispiel - Lauferkennung - Sinussignalgenerator - Spezifikationen

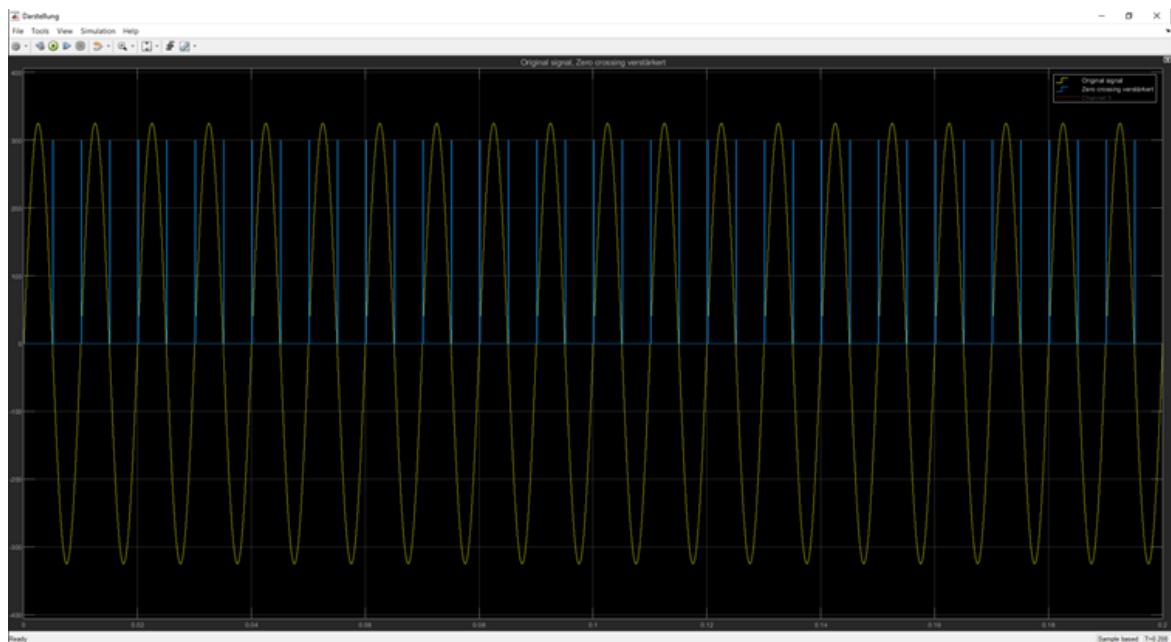


Abbildung 3.6: Testbeispiel - Lauferkennung - Sinussignal

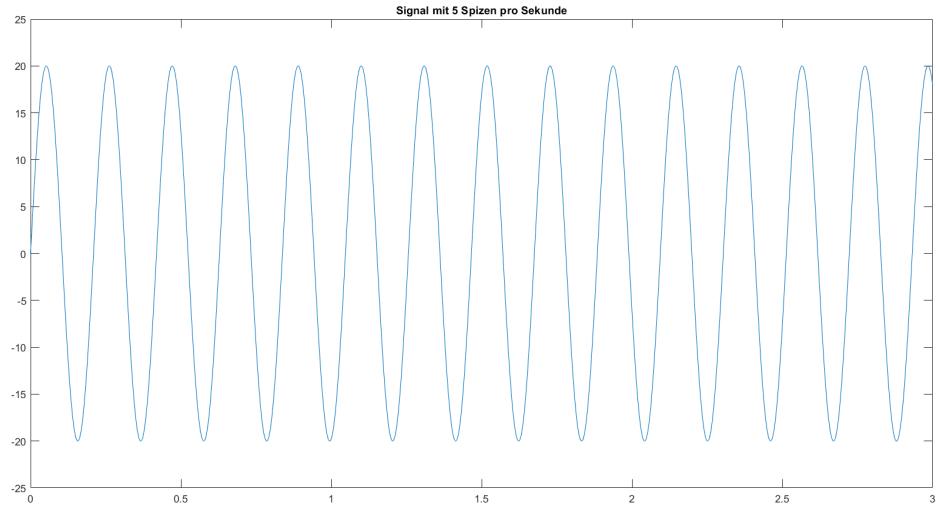
In dem Modell ist die Funktion 'Zero Crossing' verwendet. Diese zählt wie oft das Signal die x-Achse überquert. Diese Methode liefert das richtige erwartete Ergebnis,

wenn das Signal um die x-Achse dargestellt ist. Von der anderen Seite hilft diese Funktion nicht, wenn das Signal ein Offset hat, in dem dieses z.B. um die Linie $y = 300$ (Verschiebung auf der y-Achse), da in diesem Fall das Signal die x-Achse (amplitudenabhängig) nicht mehr schneidet. Das führt dazu, dass das Ergebnis nicht mehr zuverlässig ist.

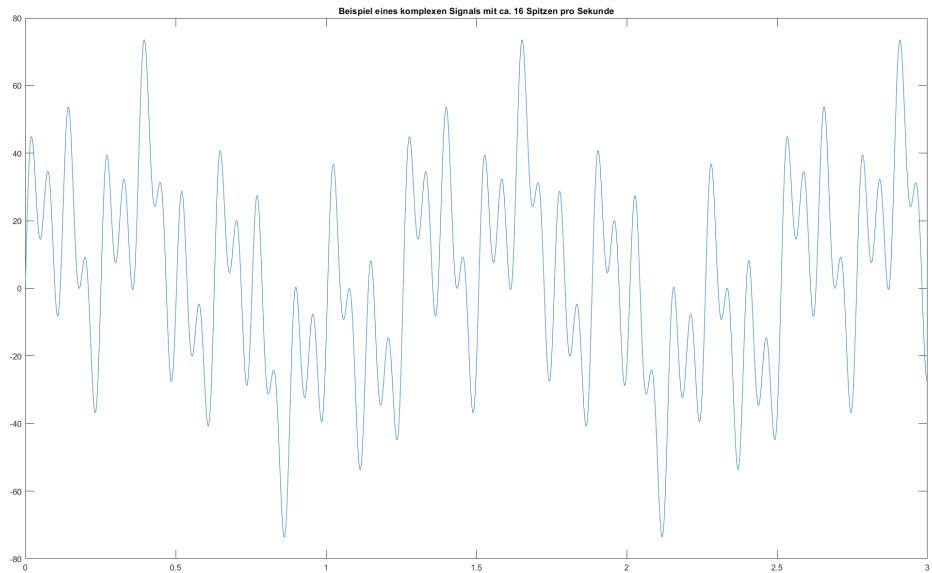
Eine andere Methode hat sich ergeben, dass die Funktion 'Counter up' in dem Modell verwendet wird. Dieses Block zählt wie oft das Signals in die positive Richtung zeigt. Das neue Implementierung hat ein zuverlässiges Ergebnis im Vergleich zum originalen Modell geliefert.

Beim Einsetzen des gleichen Vorgehens beziehungsweise Modell auf das richtige Laufsignal (Abbildung 3.3) wird eine Frequenz von ca. 11 Hz beim Laufen zurückgegeben.

Nach weiteren Auswertungen und Forschungen wird der Grund des Fehlers entdeckt. Es liegt an den Unterschied zwischen dem perfekten generierten Sinussignal und dem echten Laufsignal. Das echte Signal hat im Vergleich zum generierten viele Störungen (Rauschen). Diese lassen sich durch das benannte Modell nicht ausfiltert oder ignorieren, was zu einem falschen Ergebnis führt. Die Abbildung 3.7 stellt ein gutes Beispiel dieser Unterschied dar. Die Rauschen erhöhen die Anzahl der Spitzen.



(a) Beispiel eines einfaches Signal



(b) Beispiel eines komplexes Signal

Abbildung 3.7: Skizze eines einfachen ideales Signal sowie eines komplexeres Signal

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

3.2.2 Frequenzbasierte Lauferkennung

Das Laufsignal stellt ein wiederholtes Pattern dar, was auch durch die Frequenz erkannt wird. Das Modell muss diese Frequenz ermitteln und auswerten. Analog zum

Unterabschnitt 3.2.1 wird hier nochmal eine neue Hypothese festgelegt, die durch ein Testmodell überprüft werden soll.

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

Die Calimoto-App hat eine Abtastrate $f_s = 100 \text{ Hz}$. D.h. es gibt 100 Messwerte pro Messsekunde.

Bezogen auf die Nyquist-Frequenz (Gleichung 2.3) ist die Bandbreite beziehungsweise die minimale erkennbare Frequenz $f_n = 50 \text{ Hz}$.

Spectrum Analyzer

In der Abbildung 3.8 ist das generierte Sinussignal mit einer Frequenz von 100 Hz sowie seine Spezifikationen zu sehen. Es wird in diesem Modell ein Block 'Spectrum Analyzer' (rot markiert) als Referenz verwendet, was die einzelnen Frequenzen eines komplexen Signals zurückgibt. Der Benutzer kann die Spezifikationen des 'Spectrum Analyzer's einstellen. Die Ausgabe des Spectrum Analyzers ist in der Abbildung 3.9 zu sehen. In der Oberen Grafik werden die Intensität der Frequenz(en) (Spektrum genannt) abgebildet und in der unteren Grafik eine 3-D-Darstellung (Frequenz-Zeit-Intensität) (genannt Spektrogramm), wobei die Farbe die Intensität repräsentiert. Wenn die Abbildung näher betrachtet wird, ist eine Frequenz von 100 Hz gut sichtbar.

3 Unfallerkennung im Pocket-Mode

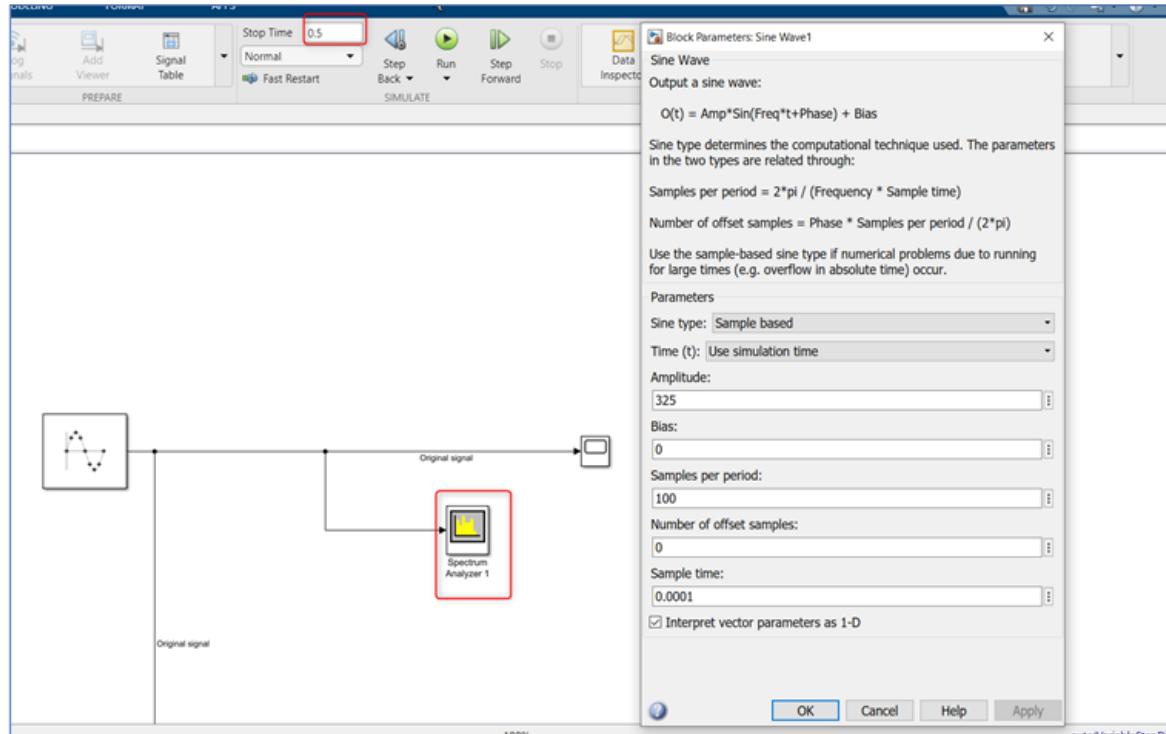


Abbildung 3.8: Testbeispiel - Frequenzbasierte Lauferkennung - Sinussignal

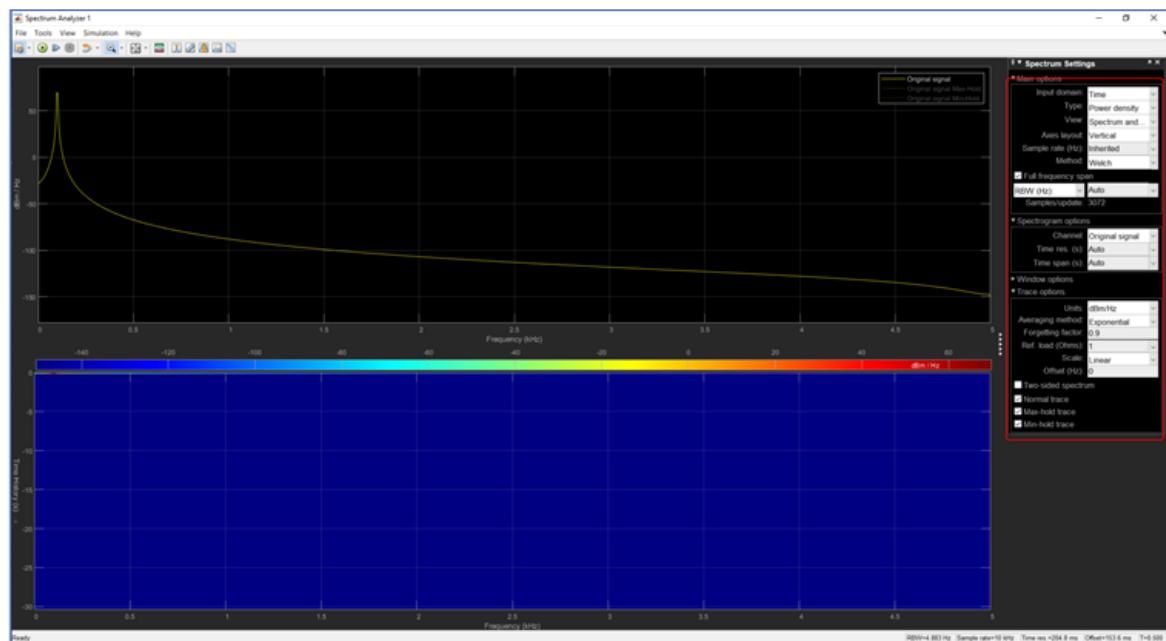


Abbildung 3.9: Testbeispiel - Frequenzbasierte Lauferkennung - Ausgabe des Spektrum-Analyzert und seine Spezifikationen

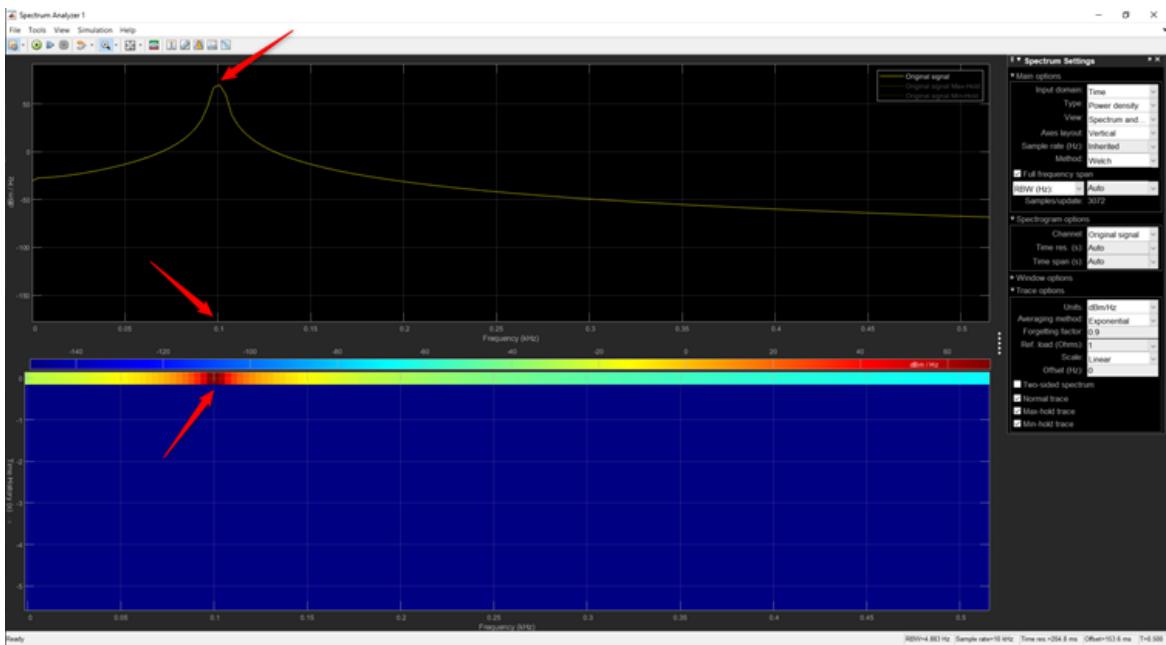


Abbildung 3.10: Testbeispiel - frequenzbasierte Lauferkennung - Ausgabe des Spektrum-Analyzert im Bereich zwischen 0-0.5 kHz

Mit anderen Einstellungen in dem Spektrum-Analyzer erhält der Benutzer eine aussagekräftigere Darstellung der Frequenz (siehe Abbildung 3.11)



Abbildung 3.11: Testbeispiel - Frequenzbasierte Lauferkennung - Ausgabe des Spektrum-Analyzert mit anderen Spezifikationen

Testmodell

Ein Testmodell ist in der Abbildung 3.12 abgebildet. Das Ziel ist die Funktionalität des Prinzips zu überprüfen, bevor dieses mit einem Echtesignal verwendet wird.

In dem Modell sind drei Sinussignale mit verschiedenen Frequenzen generiert, die zusammen summiert werden, um ein komplexes Signal zu erstellen. Die drei Sinussignale sowie deren Summe sind in der Abbildung 3.15 dargestellt.

Das Testmodell erstellt zuerst ein komplexes Signal mit bekannten Einzel- beziehungsweise Grundfrequenzen und konvertiert dieses zu einem diskreten Signals, da die FFT nicht auf ein kontinuierliches Signal anwendbar. Danach wird das FFT-Fenster durch das 'Buffer'-Block ermittelt und dann eine FFT für das entsprechende Fenster verwendet. Das Ergebnis der FFT wird weiterbearbeitet, in dem der Betrag gebildet und Spiegelung entfernt wird. Das Resultat ist eine 2-D-Matrix, wobei die x-Werte die Frequenzen auf einer Skala von 1-512 und die y-Werte die Intensität jeder Frequenz sind. Das Resultat ist in der Abbildung 3.14 dargestellt. Die X-Werte (beziehungsweise Indexe) werden extrahiert und in den Skala von 1-100 umgerechnet, in dem diese mit 100/512 multipliziert. Eine vereinfachte Ablaufschema ist in der Abbildung 3.13 abgebildet. Das Endergebnis des Modells ist eine sortierte Liste von den tatsächlichen Frequenzen. Die ersten drei Werte haben eine wesentliche große Intensität und sind somit die gesuchten Frequenzen mit kleiner Abweichung.

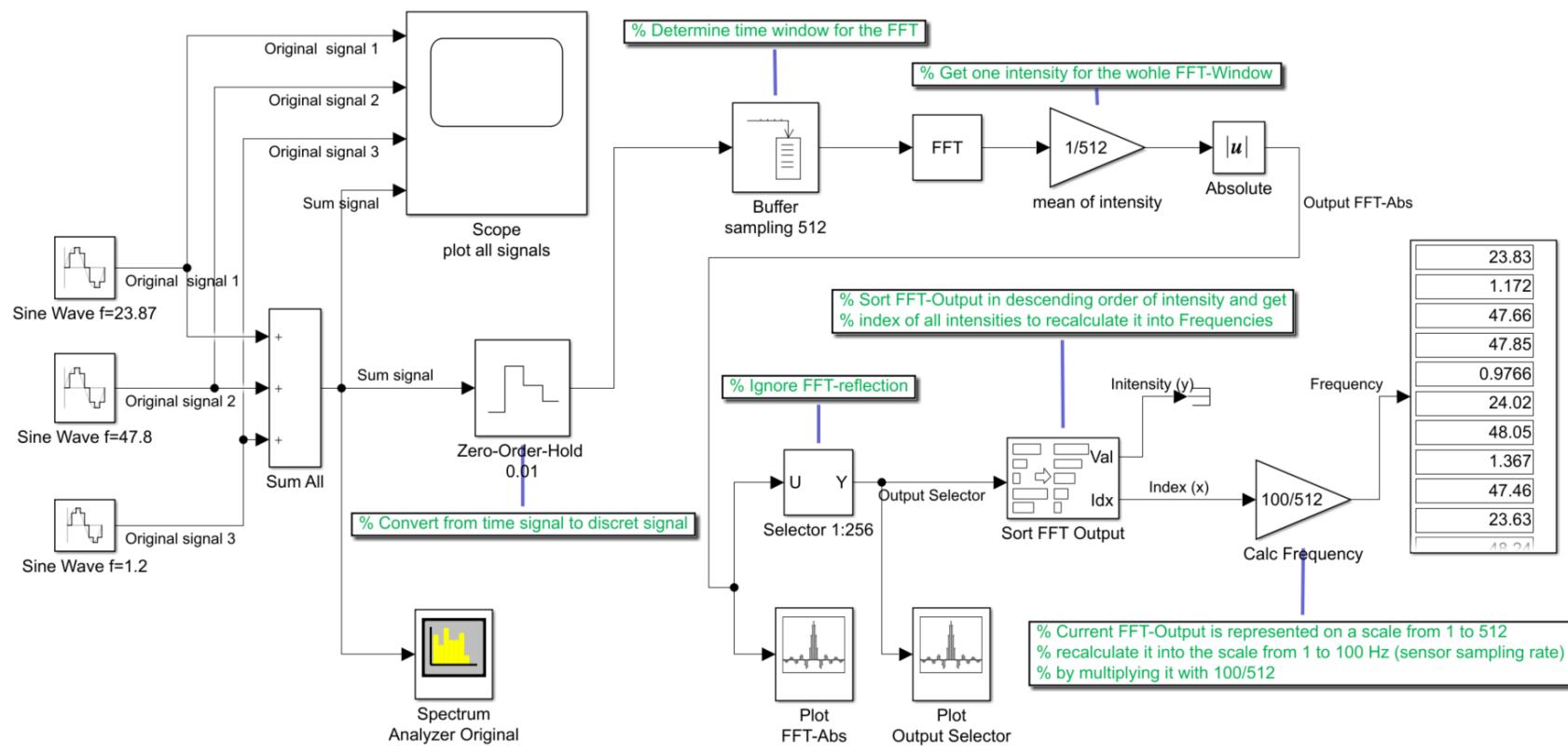


Abbildung 3.12: Testbeispiel - Frequenzbasierte Lauferkennung - FFT

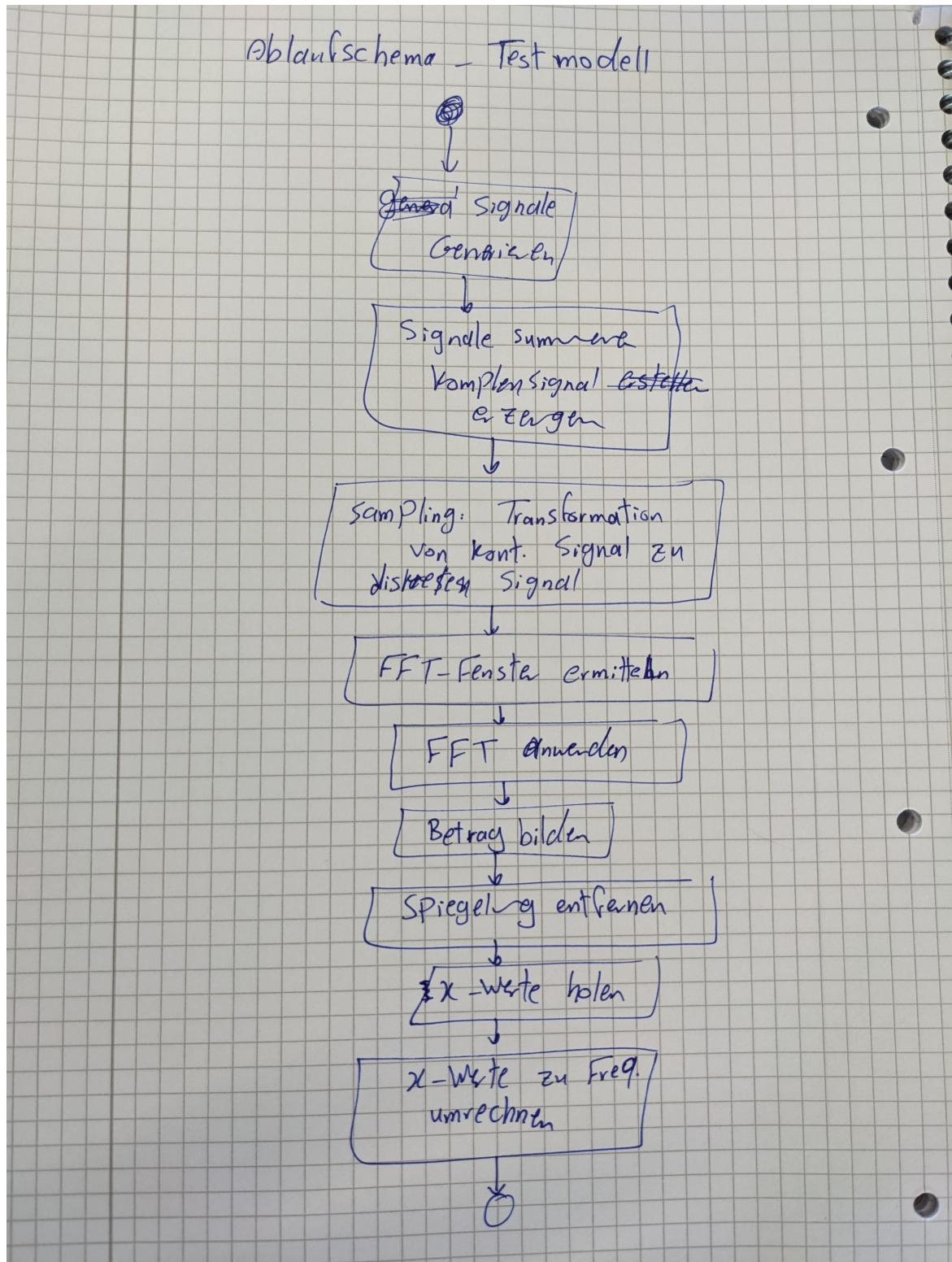


Abbildung 3.13: Ablaufschema des Testmodells

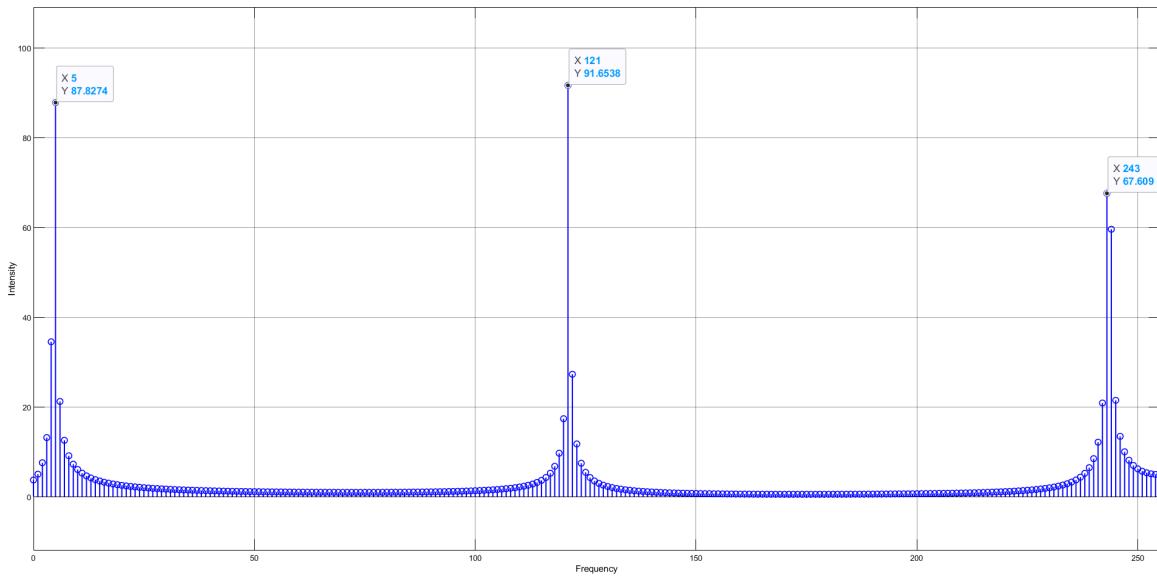


Abbildung 3.14: Das Ergebnis der FFT - Spiegelung entfernt und Beträge

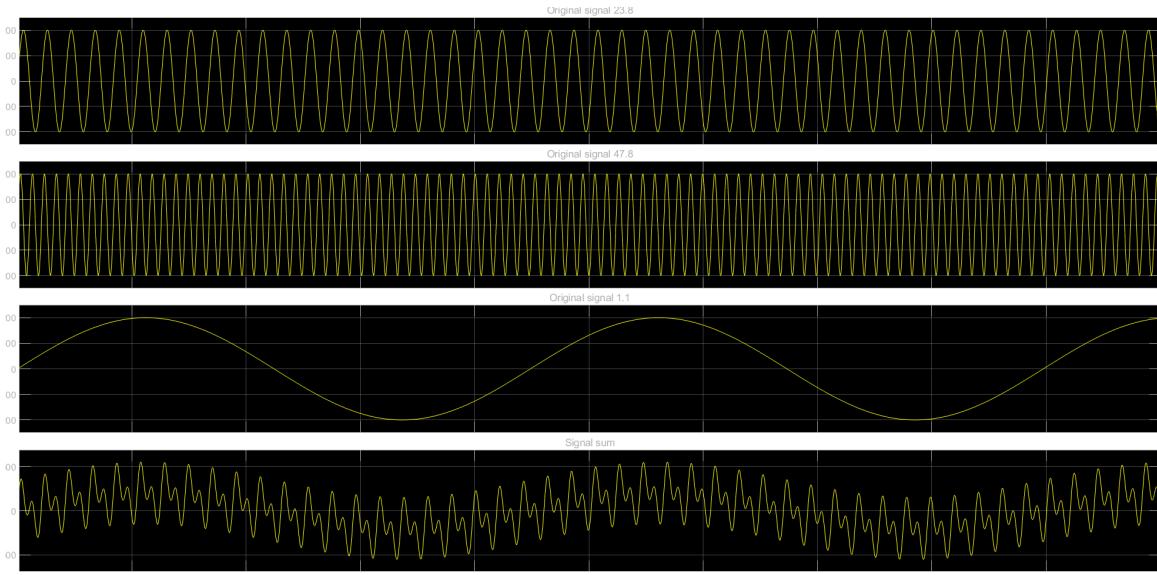


Abbildung 3.15: verschiedene Signale ($f = 23,8; f = 47,8; f = 1,1$) mit deren Summe f_g

Nachdem die Ergebnisse des Testmodells für Richtigkeit geprüft wurde, wird das Modell mit einem Echtesignal getestet.

Anwendung auf ein Echtesignal

Zum Anwenden auf ein Echtesignal wird ein Untermodell 'MotionDetection' erstellt, das die Bewegung (Laufen oder fahren) erkennt und zurückgibt. In der Abbildung 3.16 ist einen Teil des genannten Modells sichtbar.

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

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

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

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

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

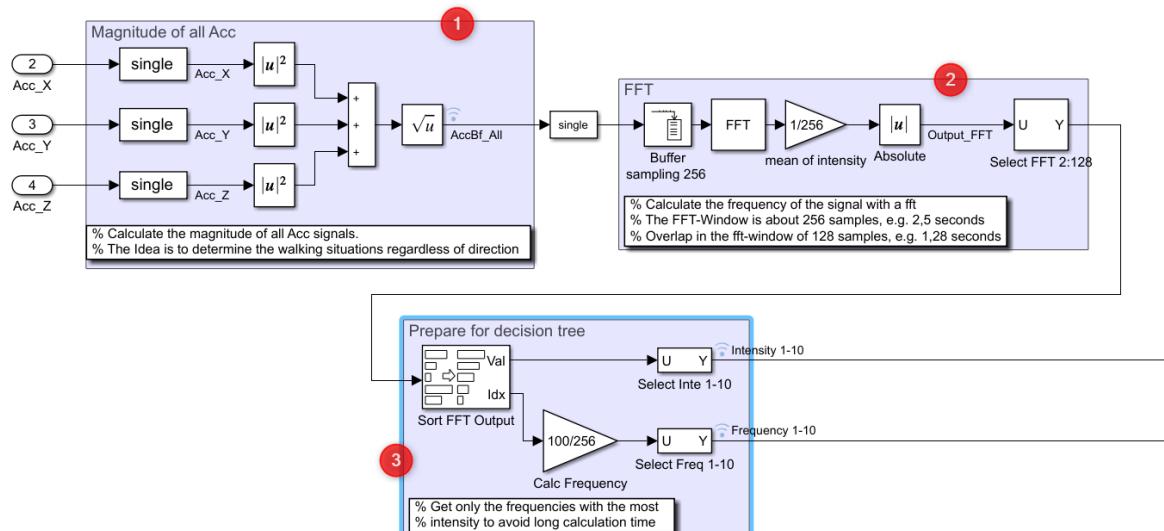


Abbildung 3.16: Lauferkennung - Frequenzbasierte - FFT - Echtmodell

Die vom dritten Teil ausgegangenen Daten werden zu einer Matlabdatei (Abbildung 3.17) geleitet, wo eine Entscheidung getroffen werden muss.

Entscheidungskriterien - Matlabskript

Die Abbildung 3.17 zeigt die Matlab-Funktion, die die Entscheidung übers Laufen trifft. Die Eingänge der Funktion sind die zehn Frequenzen mit den höchsten zehn Intensitäten sowie die aktuelle Geschwindigkeit. Die durch die Funktion letzte erkannte Aktivität sowie deren Zeitpunkt werden auch in die Funktion geleitet. Nach dem Durchlauf liefert die Matlab-Funktion eine Id-Zahl, die eine Aktivität entspricht. Der Aktivität-ID-Zusammenhang ist in der Tabelle 3.1 abgebildet.

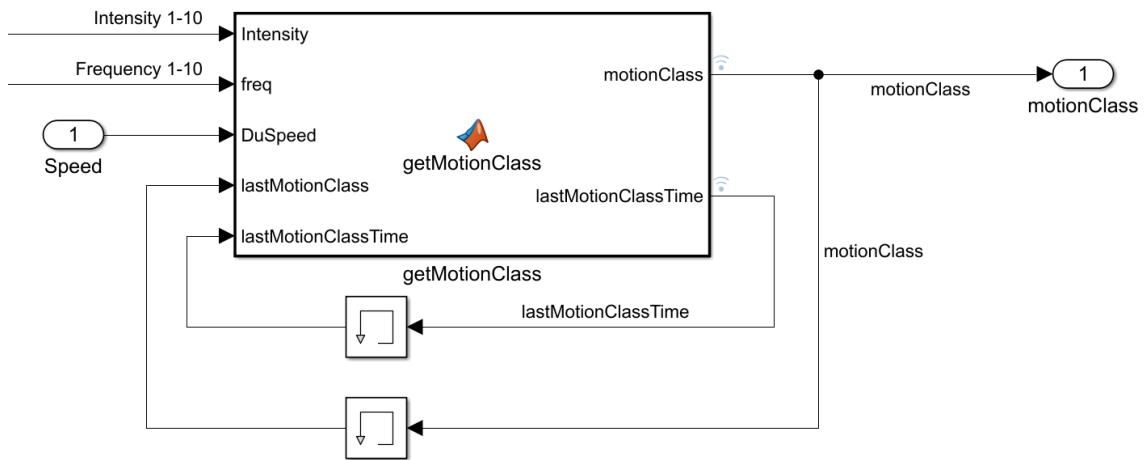


Abbildung 3.17: Ein- und Ausgänge des Entscheidungsskripts

Tabelle 3.1: *Ausgangsmöglichkeiten der Entscheidungsfunktion*

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

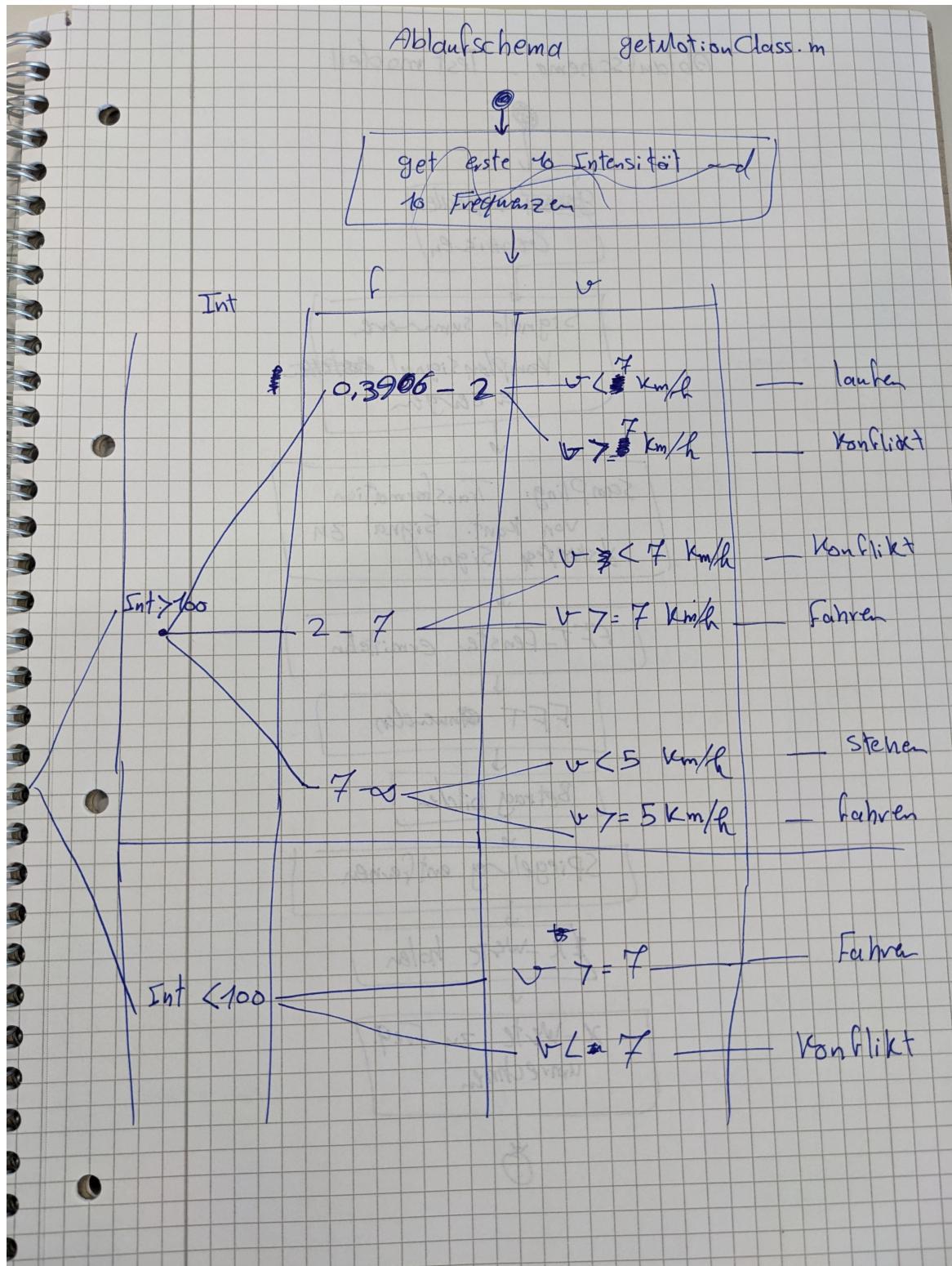


Abbildung 3.18: Entscheidungsbaum des Laufens

Die Abbildung 3.18 zeigt das einfache Entscheidungsbaum, wonach eine Lauferken-

nung erfolgt wird. Die Entscheidung erfolgt in einer Matlab-Funktion innerhalb vom Simulink-Modell (siehe Abschnitt A.1).

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

- Stehen beziehungsweise keine Bewegung
- Laufen
- Fahren
- Konflikt: Wenn die Entscheidungskriterien (Frequenz und Geschwindigkeit) verschiedene Aussagen liefern

Der Entscheidungsbaum fängt bei der Intensität an. Wenn die größte Intensität kleiner als der Schwellwert (100) ist, kann die dazugehörige Frequenz für die Entscheidung nicht vertrauend sein und werden die vier nachfolgenden Intensität untersucht. Sollte immer noch keine zulässige Intensität ergeben, wird sofort nach Geschwindigkeit geschaut und diese für die Entscheidung verwendet.

Z.B. Bei einer Geschwindigkeit von 30 km/h ist von einer Fahr auszugehen und bei 3 km/h ist das Laufen die richtige Entscheidung.

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 die Geschwindigkeitsüberprüfung verschiedene Aussagen zurückgeben, ist von einem 'Konflikt' auszugehen. In diesem Fall werden die vier nachfolgenden Frequenzen überprüft.

Testbeispiel

Die Signale in der Abbildung 3.19 sind aus einem Echtesignal ausgeschnitten, in dem der Fahrer nach einer Fahrt gelaufen ist. Das Fahren ist in der Abbildung 3.19a sowie das Laufen in der Abbildung 3.19b abgebildet.

Das Lauferkennung-Modell wurde mit dem entsprechenden Signal getestet. Das Modell hatte die richtigen Frequenzen erkannt. Für das erste Teilsignal aus der Abbildung 3.19a hat eine Frequenz von durchschnittlich 20 Hz geliefert. Das Teilsignal aus der Abbildung 3.19b hat eine Frequenz von etwa 1.8 Hz. Die Frequenzermittlung der Lauferkennung war sehr nah zur Realität. Nachdem die Frequenzen ermittelt wurden, sind diese ins Entscheidungsskript geleitet. Das Skript entscheidet dann mit einer Geschwindigkeitsprüfung, welche Aktivität (Fahren oder Laufen) durchgeführt wurde.

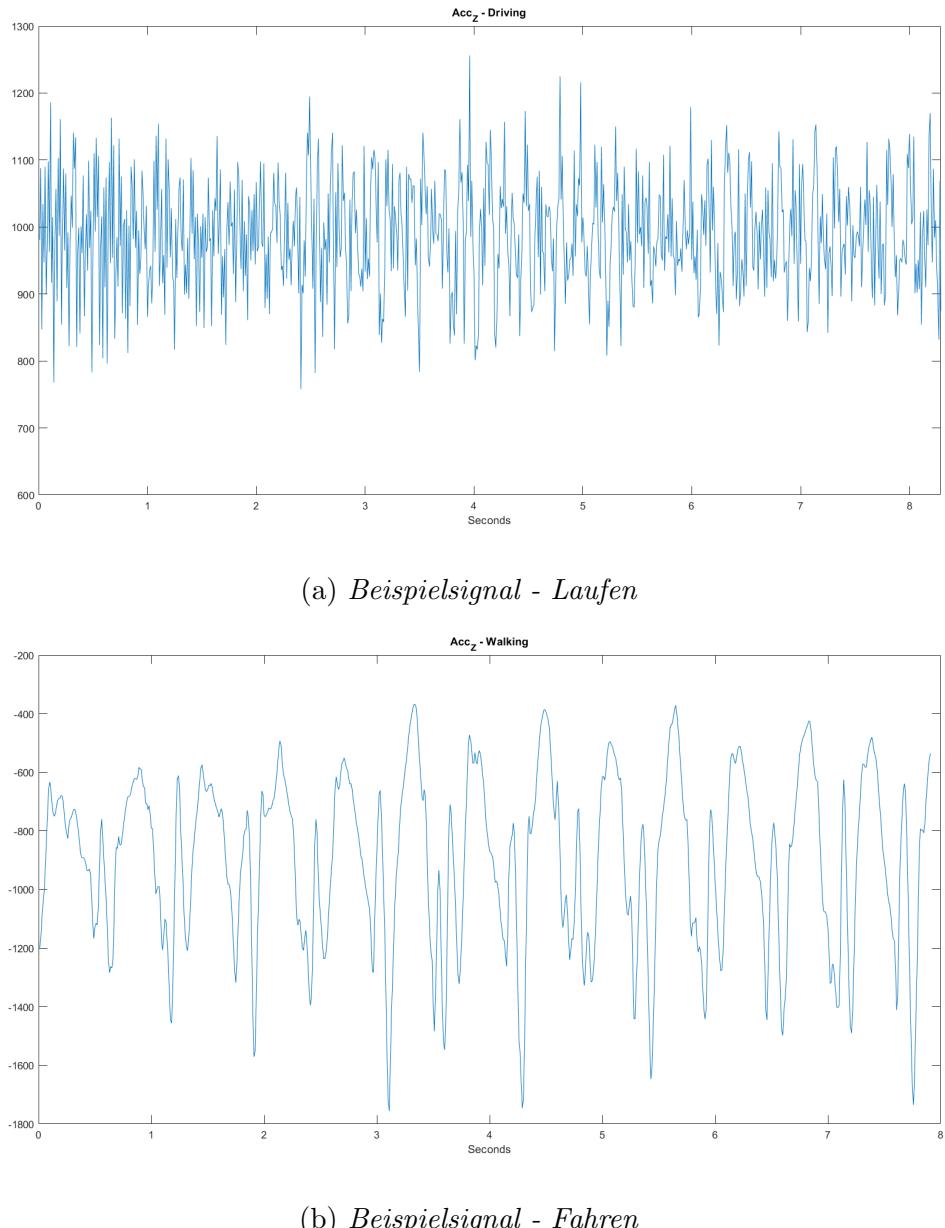


Abbildung 3.19: Abgeschnittene Teile eines Beispieldesignals

Sinnvolle Werte auswählen

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

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

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

- **Frequenzbereich:** das entspricht der Frequenzbereiche vom Laufen und vom Fahren.
- **Geschwindigkeit:** In dem Skript ist ein Wert von 7 km/h als Schwellwert zwischen Fahren und Laufen ausgewählt. Der Wert ist in der Straßenverkehrsordnung als Schrittgeschwindigkeit anerkannt[Bussgeldkataloge, 2022]. Erfahrungsmäßig läuft der Mensch in einer Geschwindigkeit zwischen 5 und 10 km/h.

3.3 Auf- und Absteigen

Beim Auf- und Absteigen gibt's starke Winkeländerung im Raum (3D). Wenn einen GH erkannt wird, sollte einen Alarmauslösung ergeben. Ohne GH sollte der Fall nicht als Unfall erkannt.

3.4 An Ampel stehen und Fuß runter

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

Die erste Position ist das Bein während einer Fahrt mit der horizontalen Beinstellung. Wenn der Fuß am Boden ist, steht das Bein einer vertikalen Postion.

- Signalbeispiel mit Phasenmarkungen (Horizontal und Vertikal)

Nach dem Testen: Keine falsche Alarmauslösungen. Grund ist, dass die Winkeländerung nicht 90 grad beträgt sondern nur ca. 20-30 grad (Abbildung 3.20). Die Person hat sein Bein beim Sitzen nicht genau horizontal sondern leicht nach Unten geneigt. Und wenn der Fahrer sein Fuß runter setzt, ist diese auch nicht genau vertikal sondern bisschen gebogen mit einem Winkel von ca. 10-20 Grad zum Vertikallinie. D.h. die Winkeländerung nicht 90 Grad ist.



(a) Beinposition während einer Fahrt



(b) Beinposition Beim Stehen

Abbildung 3.20: Beinpositionen während einer Fahrt und gestreckt

3.5 Verifikation des Algorithmus'

3.5.1 Groundtruth-Daten sammeln, Tatsächliche Aktivitätsdaten

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

dem richtigen Signalteil zugeordnet werden können, ist eine Synchronisierung zwischen dem Signal und dem dazugehörigen Video erforderlich.

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

Die erwähnte Labels muss der Benutzer vorhin definieren und in das Tool einladen. In der Abbildung 3.22 ist die Tabelle der definierten Labels sichtbar. Die Tabelle hat sieben verschiedene Klassen, die sich im Tool mit den F-Tasten einfach eingeben lassen. 'Undefined' entspricht unbekannter Eingabe, wenn das Motorrad nicht im Video sichtbar ist. 'Post-Crash' repräsentiert der Motorradstand nach einem Aufprall beziehungsweise einem Sturz bis zum Ruhestand.

In der Abbildung 3.21 ist ein verkürztes Beispiel der exportierten Tabelle, in der die neue Spalte 'GroundTruthId' sichtbar ist.

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

Abbildung 3.21: Beispiel der exportierten Tabelle mit der neuen Spalte

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

Abbildung 3.22: *Die definierte Labels*

4 Ergebnisse

5 Ausblick

- ...

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

Literaturverzeichnis

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

- [MostrandomComps 2019] MOSTRANDOMCOMPS: *Motorcycle Crash Compilation.* https://www.youtube.com/watch?v=dMsYT_7xNfg&ab_channel=MostrandomComps. Version: November 2019. – Aufgerufen am 06.09.2022
- [Moto-Passion 2018a] MOTO-PASSION: *Animals Vs Bikers.* https://www.youtube.com/watch?v=KiRydLwtAyc&ab_channel=MotoPassion. Version: Juli 2018. – Aufgerufen am 06.09.2022
- [Moto-Passion 2018b] MOTO-PASSION: *CRAZY Driver Vs Biker.* https://www.youtube.com/watch?v=HFig7J40mT8&ab_channel=MotoPassion. Version: Dezember 2018. – Aufgerufen am 06.09.2022
- [Moto-Passion 2018c] MOTO-PASSION: *Motorcycle Crashes and Mishaps.* https://www.youtube.com/watch?v=jLm0b4Mq-pQ&ab_channel=MotoPassion. Version: Juli 2018. – Aufgerufen am 06.09.2022
- [Moto-Passion 2018d] MOTO-PASSION: *Motorcycle Crashes on the Road.* https://www.youtube.com/watch?v=S_lizETnMpk&ab_channel=MotoPassion. Version: Juli 2018. – Aufgerufen am 06.09.2022
- [N. Kehtarnavaz 2006] N. KEHTARNAVAZ, C. G.: DSP System Design using LabVIEW and Simulink: A Comparative Evaluation. (2006). <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1660510>
- [Nasa 2019] NASA: *How Does GPS Work?* <https://spaceplace.nasa.gov/gps/en/>. Version: Juni 2019. – Aufgerufen am 01.08.2022
- [NHTSA] NHTSA: *Ratings, Biomechanics and Trauma, Motorcycle Safety.* <https://www.nhtsa.gov>. – Aufgerufen am 01.08.2022
- [NTI-Audio] NTI-AUDIO: *Fast Fourier Transformation FFT - Basics.* <https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft>
- [health organization 2022] ORGANIZATION, World health: *Road traffic injuries.* <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>. Version: Juni 2022. – Aufgerufen am 01.08.2022
- [Rogers] ROGERS, K.: *gyroscope.* Encyclopedia Britannica. <https://www.britannica.com/technology/inertial-guidance-system>. – Aufgerufen am 01.08.2022
- [Sathish 2021] SATHISH: *How to Measure Acceleration in Smartphones using Accelerometer?* <https://blog.contus.com/how-to-measure-acceleration-in-smartphones-using-accelerometer/>. Version: November 2021. – Aufgerufen am 01.08.2022

[Schnee u. a. 2021] SCHNEE, Jan ; STEGMAIER, Jürgen ; LI, Pu: A probabilistic approach to online classification of bicycle crashes. In: *Accident Analysis and Prevention* 160 (2021), 106311. <http://dx.doi.org/https://doi.org/10.1016/j.aap.2021.106311>. – DOI <https://doi.org/10.1016/j.aap.2021.106311>. – ISSN 0001-4575

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

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

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

[Weisstein] WEISSTEIN, E. W.: *Fast Fourier Transform.* <https://mathworld.wolfram.com/FastFourierTransform.html>

A Anhang

A.1 Entscheidungsfunktion

Screenshot vom Skript mit einer Erklärung (Abbildung A.1)

```
1  function [motionClass, lastMotionClassTime] = fcn(Intensity, freq, DuSpeed, lastMotionClass, lastMotionClassTime)
2  % status = -1;
3  for iFq = 1:5
4      if Intensity(iFq) > 100
5          lastMotionClassTime = 0;
6          % lastClassTime = 0;
7          if freq(iFq) >= 7
8              % high frequency (most likely driving), check with speed
9              if DuSpeed < 5
10                  % high frequency but very low speed => sitting on motorbike and standing still
11                  motionClass = 0;
12              else
13                  % there is movement
14                  motionClass = 2;
15              end
16              break
17          elseif freq(iFq) >= 0.3906 && freq(iFq) <= 2
18              % walking, (most likely walking) check with speed
19              if DuSpeed <=7
20                  % walking speed
21                  motionClass = 1;
22              break
23          else
24              % speed > 7 kmh => not walking => conflict => check next frequency
25              motionClass = -1;
26              continue;
27          end
28      elseif freq(iFq) > 2 && freq(iFq) < 7
29          % undeclared. Maybe walking and maybe driving. Double check with the speed. If the speed is bigger than 7 km/h than he must be driving.
30          % If the speed's smaller than 7 km/h it should stay undeclared, he might be driving with no GPS-Signal (e.g. no speed) or walking
31          if DuSpeed >= 7
32              motionClass = 2;
33          break;
34      else
35          motionClass = -1;
36          continue;
37      end
38
39      elseif freq(iFq) < 0.3906 % undeclared f < 0.3906 Hz
40          % not intended to get here
41          % (sensor couldn't sense frequencies smaller than 0.396)
42          motionClass = -2;
43          break
44      else
45          % not intended to get here either; all usecases were defined in the
46          % previous elseifs
47          motionClass = lastMotionClass;
48          break
49      end
50  else
51      if DuSpeed >= 7
52          motionClass = 2;
53          break;
54      else
55          if lastMotionClassTime <=100
56              lastMotionClassTime = lastMotionClassTime + 1;
57              motionClass = lastMotionClass;
58          else
59              lastMotionClassTime = 0;
60              motionClass = -1;
61          end
62          continue
63      end
64  end
65 end
66
67 end
68
```

Abbildung A.1: Entscheidungsskript durch mehrere Kriterien (Frequenz und Geschwindigkeit)

- Tabelle von Jan

Eidesstattliche Erklärung

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

Ort und Datum: Stuttgart, den _____ Unterschrift: _____