Karel de Grote-Hogeschool
start met voorsprong

Departement Industriële Wetenschappen en Technologie
Masterproef 2012-2013

Voertuigsensoren in functie van wegdekbeheer

Ken Vanherpen

Promotoren: Dr. Paul De Meulenaere, KdG-IWT
Dr. Peter Hellinckx, KdG-IWT
Dr. Wouter Hendrickx, KdG-IWT

Proefschrift tot het behalen van de graad van
Master of Science in de Industriële Wetenschappen
Elektronica-ICT afstudeerrichting Automotive Engineering
Antwerpen, juni 2013

Karel de Grote-Hogeschool
start met voorsprong

Departement Industriële Wetenschappen en Technologie
Masterproef 2012-2013

Voertuigsensoren in functie van wegdekbeheer

Ken Vanherpen

Promotoren: Dr. Paul De Meulenaere, KdG-IWT
Dr. Peter Hellinckx, KdG-IWT
Dr. Wouter Hendrickx, KdG-IWT

Proefschrift tot het behalen van de graad van
Master of Science in de Industriële Wetenschappen
Elektronica-ICT afstudeerrichting Automotive Engineering
Antwerpen, juni 2013

VOORWOORD

Na het behalen van mijn diploma *Bachelor in de Elektronica-ICT* eind juni 2011, vond ik de tijd rijp om via een schakeljaar een gooi te doen naar het behalen van een Masterdiploma. Daar ik zowel een passie heb voor elektronica-ICT als voor wagens, vond ik mijn gading in de opleiding *Master in de Industriële Wetenschappen Elektronica-ICT afstudeerrichting Automotive Engineering* aan de Karel-de-Grote Hogeschool.

Kort na het succesvol voltooien van het schakeljaar diende er beslist te worden welk thesisonderwerp er zou behandeld worden tijdens het daaropvolgende masterjaar. Mijn interesse ging meteen naar het onderwerp *"Voertuigsensoren in functie van wegdekbeheer"*, en na een informatief gesprek met dr. De Meulenaere was het mij duidelijk dat dit mijn masterthesis zou worden. Deze thesis zou een afwisseling worden van theorie en praktijk met het nodige accent op de ingenieurswetenschappen, welke kadert in het grotere Sensovo-project van het VIM namelijk *"Wegdekbeheersysteem"*.

Een vlotte start werd genomen in september 2012, waardoor al vrij snel duidelijk werd welke richting deze thesis diende uit te gaan. Toch was deze vlotte start niet het voorteken voor het verdere verloop. Op weg naar dit resultaat dienden allerhande hindernissen overwonnen te worden, zowel op persoonlijk als op professioneel vlak, welke ik niet allemaal alleen heb kunnen afhandelen.

Daarom zou ik van de gelegenheid willen gebruik maken om enkele personen te bedanken. Allereerst zou ik dr. Paul De Meulenaere willen bedanken voor de technische ondersteuning betreffende automotive engineering, het verschaffen van inzicht in de fysica en het afhandelen van alle contractuele verplichtingen die geleid hebben tot het verkrijgen van licenties. Mijn dank gaat ook uit naar dr. Peter Hellinckx en dr. Wouter Hendrickx, beide als intern onderzoeker belast met het goede verloop van het Sensovo-project. Door hun onderzoekservaring heb ik praktisch inzicht verkregen in de materie, geleerd om op een effectieve manier nieuwe en/of veranderende onderzoeksresultaten te analyseren en om in teamverband te werken aan een dergelijk groot project. Ook had ik graag grad. Hans Vandenbulcke willen bedanken. Zonder zijn praktische expertise betreffende voertuignetwerken en het voorzien van enkele testvoertuigen, had het praktische gedeelte van deze thesis nooit zo vlot verlopen.

Uiteraard dien ik ook familie en vrienden te bedanken voor hun steun en geduld tijdens deze drukke periode. Dankzij hun begrip heb ik mij ten volle concentreren op deze thesis, waarvoor dank!

OPDRACHTOMSCHRIJVING

Momenteel wordt er voor het opsporen van defecten in het wegdek gebruik gemaakt van dure ARAN-meetvoertuigen, die elke 2 jaar een volledige analyse maken van het Vlaamse wegennet. Het probleem hierbij is dat defecten meestal in een laat stadium gelokaliseerd worden, waardoor de kostprijs om deze te herstellen groter wordt.

Aangezien de huidige generatie voertuigen voorzien zijn van tal van sensoren ter ondersteuning van de bestuurder en zijn comfort, dient de vraag gesteld te worden of deze sensoren eventueel aangewend kunnen worden om defecten in het wegdek te detecteren [1]. Enkele mogelijke interessante systemen met hun bijhorende sensoren zijn hierbij:

Systemen in het voertuig	*Respectievelijke sensoren*
ABS	Wielsnelheidssensoren
ESP/ESC/ DSC	Stuurwielhoek-sensoren, wielsnelheidssensoren, laterale acceleratie, rotatie van het voertuig (yaw)
Active Steering	Wielsnelheidssensoren, rotatie van het voertuig (yaw)

Tabel 0-1: Voertuigsystemen en hun sensoren

Het in 1983 door Bosh ontwikkelde CAN-protocol, welke door ISO gestandaardiseerd is sinds 1993, dient verplicht aanwezig te zijn in alle voertuigen die op de Europese markt zijn sinds 2004 (reeds sinds 2001 in het geval van benzinevoertuigen). Hierdoor kunnen alle aanwezige voertuigsensoren en -systemen hun data plaatsen op een CAN-bus, om zo met elkaar te communiceren. De bedoeling is dan ook om de data op deze CAN-bus te onderscheppen [2], en verbanden te zoeken met defecten in het wegdek. Het is namelijk aannemelijk dat bijvoorbeeld bij het rijden in een put, bepaalde sensordata op de CAN-bus een bepaald afwijkend gedrag vertonen specifiek door deze gebeurtenis [1].

Indien zich een dergelijke situatie voordoet, en met andere woorden een defect in het wegdek gedetecteerd wordt, moeten deze gegevens samen met hun GPS-locatie verstuurd worden naar een centraal meldingspunt.

DOELSTELLINGEN

In een eerste fase zal er met behulp van het simulatiepakket AMESim [3] een theoretische analyse gemaakt worden van mogelijk interessante sensoren, waarvan hun data een afwijkend gedrag (een uniek patroon) vertonen bij een defect in het wegdek. Specifiek zal dit defect in het wegdek een put zijn van 20 cm diameter en 4 cm diepte.

Nadat uit simulaties besloten is welke sensoren interessant zijn ter detectie van defecten in het wegdek, dienen deze sensorgegevens opgespoord te worden op de CAN-bus van een voertuig. Daarvoor dient de CAN-bus geanalyseerd te worden met behulp van analysesoftware, om zo de mogelijk interessante sensoren te lokaliseren op de CAN-bus [2]. Nadat hun ID is opgespoord, dient het theoretisch patroon omgezet te worden naar een praktisch patroon.

De eerder (aan)genomen theoretische conclusies dienen getoetst te worden met de werkelijk gemeten gegevens op de CAN-bus. Enkele belangrijke te beantwoorden vragen in deze fase van het onderzoek zijn dan ook:

Open punt 1: *Hoe nauwkeurig zijn de gegevens op de CAN-bus, en volstaat deze om defecten in het wegdek te detecteren?*

Open punt 2: *Naast nauwkeurigheid zal ook de frequentie waarop de data op de CAN-bus geplaatst wordt van belang zijn. Zal deze wel voldoende hoog zijn?*

Open punt 3: *Is er een standaardoplossing voor de verschillende types van wagens, daar elke constructeur zijn eigen codering op de CAN-bus gebruikt? Met andere woorden: is er wel een universeel patroon?*

Open punt 4: *Is het voldoende om een defect in het wegdek te detecteren met behulp van een enkele set aan meetgegevens, afkomstig van één en hetzelfde voertuig?*

Het is duidelijk dat het voorgaande van cruciaal belang is voor het verdere verloop van de thesis. Nadat dit in zekere mate succesvol is afgerond, kan er overgegaan worden naar de tweede belangrijke fase.

Hierin moet het mogelijk afwijkende gedrag (kortweg het patroon) gekoppeld worden aan een locatie met behulp van GPS-coördinaten, waarna alle meetgegevens via een draadloze verbinding verstuurd moeten worden naar een centraal punt. Men denkt eraan om hiervoor gebruik te maken van een reeds bestaande oplossing, namelijk "*uCAN*" [4], [5], [6], [7], [8]. Hoewel deze uCAN-module interessant lijkt voor dit project, zijn er mogelijk enkele nadelen verbonden aan de huidige configuratie:

Open punt 5: *uCAN stuurt constant gegevens door naar de server, waar de data verwerkt wordt. De enorme hoeveelheid aan constante datastroom zou in dit geval de kosten voor de draadloze verbinding via GPRS de hoogte doen inschieten.*

Een mogelijk denkbare oplossing voor dit probleem zou zijn om enkel data te versturen in het geval er effectief een (mogelijke) put gedetecteerd wordt, waarna er eventueel op de server een statische analyse dient gemaakt te worden op basis van meerdere ontvangen datagegevens. Deze oplossing doet opnieuw enkele vragen rijzen:

Open punt 6: *Is het mogelijk om dergelijke logica te implementeren in de uCAN-module? Hierbij moet rekening gehouden worden met het feit of er dan voldoende opslag- en rekencapaciteit aanwezig is in de uCAN-module.*

Open punt 7: *Indien het mogelijk is om dergelijke logica te implementeren in de uCAN-module, hoe gaan we ervoor zorgen dat er nog altijd (OTA-)updates mogelijk zijn? Elk type voertuig zal namelijk zijn eigen logica hebben, daar de ID's van gelijkaardige berichten op de CAN-bus steeds verschillend zijn, waardoor het universele karakter van de uCAN-module verdwijnt.*

ABSTRACT

Road quality control requires frequent observations of road deteriorating, in order to warn drivers of potentially dangereous situations and to plan repairs. In the past, road inspection was often done manually, or using expensive measurement vehicles. This paper presents an automated detection approach based on cheap and readily available sensors, namely the car's CAN-bus data, and a smartphone's sensors. Using vehicle simulations together with real-life measurements, an algorithm was devised to do an automated identification of road defficiencies while driving. The construction of this algorithm, together with results of experiments in the field, are presented in this work.

INHOUDSOPGAVE

LIJST MET TABELLEN

LIJST MET FIGUREN

LIJST MET GEBRUIKTE AFKORTINGEN

ABS	Anti-Lock Braking System
ANWB	Koninklijke Nederlandse Toeristenbond
API	Application Programming Interface
ARAN	Automatic Road Analyzer
ATOP	Automotive Telematics On-Board Unit Platform
bCall	Breakdown Call
BRP	Baud Rate Prescaler
CAN	Controller Area Network
CAN-H	CAN High
CAN-L	CAN Low
CiA	CAN in Automation
CRC	Cyclic Redundancy Check
CSMA/BA	Carrier Sense Multiple Access with Bitwise Arbitration
CSV	Comma-Seperated Values
DB	Data Byte
DSC	Dynamic Stability Control
eCall	Pan-European in-vehicle emergency call system
ECU	Electronic Control Unit
ESC	Electronic Stability Control
ESP	Electronic Stability Program
EU	European Union
GPRS	General Packet Radio Service
GPS	Global Positioning System
INTRO	Intelligent Roads
ISO	International Standard Organization
LIN	Local Interconnect Network
MOST	Media Oriented Systems Transport
MSB	Most Significant Bit
NRZ	Non-Return-to-Zero
OBD	On-Board Diagnostics
OSI	Open Systems Interconnection
PCM	Powertrain Control Module
RJW	Resynchronization Jump Width
TCS	Traction Control System
VIM	Vlaams Instituut voor Mobiliteit

1 INLEIDING

Deze thesistekst is opgedeeld in enkele hoofdstukken, in overeenstemming met het gevolgde pad om de voorgenoemde doelstellingen te bereiken.

Hoofdstuk ‘2 Vooronderzoek’ handelt over het technisch vooronderzoek. Er wordt een overzicht geschetst van de verschillende CAN-netwerken in een voertuig, het CAN-protocol wordt toegelicht en CAN-berichten worden meer in detail geanalyseerd. Verder wordt onderzocht welke voertuigsystemen, en hun bijkomstige sensoren, relevant kunnen zijn in het licht van deze thesis.

In hoofdstuk ‘3 Simulatie’ wordt aangetoond op welke manier simulaties, in het simulatiepakket AMESim [3], zijn aangewend om een geschikt algoritme te bekomen. Door een geïsoleerde put in het wegdek te modelleren, wordt aangetoond dat bepaalde sensorgegevens unieke patronen vertonen bij deze gebeurtenis. Een fusie van deze sensorgegevens levert een robuust algoritme, welke in staat is om putten in het wegdek te detecteren.

Een testopstelling werd ontwikkeld om het, uit simulatie verkregen, algoritme in de praktijk te testen. Hoofdstuk ‘4 Praktische uitvoering’ beschrijft hoe het algoritme in de praktijk op zijn juistheid geverifieerd kan worden. Na deze verificatie en het finetunen van het algoritme, worden enkele testritten besproken welke de werking van het algoritme aantonen.

2 VOORONDERZOEK

2.1 Onderzoeken van CAN

2.1.1 Overzicht

De huidige generatie voertuigen bevatten zo'n 70 ECU's die elk een of meerdere elektrische systemen[1] of subsystemen aansturen in het voertuig. Enkele voorbeelden van dergelijke ECU's zijn:

- Engine Control Unit
- Body Control Module
- Speed Control Unit
- Brake Control Module

Essentieel is de communicatie tussen de verschillende ECU's. Zo zal de Brake Control Module informatie doorsturen naar de Engine Control Unit, om in geval van doorslippen van de wielen het koppel/toerental te verlagen. Om deze communicatie tot stand te brengen heeft Bosch in de jaren '80 het CAN-protocol ontwikkeld [9], welke later in 1993 gestandaardiseerd is door de ISO in de ISO-11898 norm [10]. CAN is een asynchroon serieel netwerk gebaseerd op het OSI zeven lagen model, maar gebruikt hiervan slechts de onderste twee lagen. Op deze twee onderste lagen wordt verder in deze uiteenzetting dieper op ingegaan.

Door het grote aantal ECU's, wordt een voertuig opgedeeld volgens een classificatie zoals voorgesteld in 'Tabel 2-1'. Elk systeem aanwezig in een voertuig wordt vervolgens ingedeeld conform deze classificatie.

		Body		
Powertrain (aandrijflijn)	*Chassis*	*Comfort*	*Passieve veiligheid*	*Multimedia (Infotainment)*
Motor-management	Stuur-bekrachtiging	Ruitenwissers	Airbags	Radio
Elektronische transmissie	ESP	Verlichting	Veiligheids-gordels	Navigatie
	ABS	Zetelverwarming		

Tabel 2-1: Classificatie van de voertuigsystemen

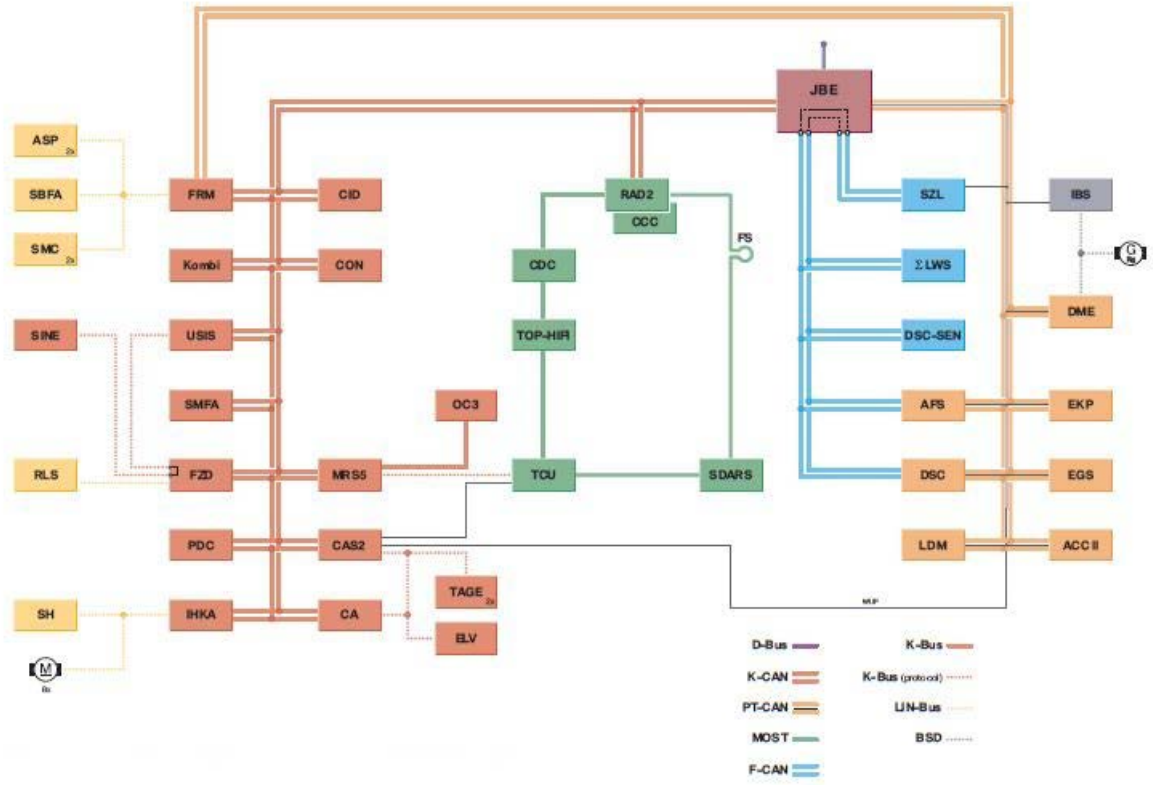
Deze vorm van classificatie is algemeen gekend in de automotive-sector, en heeft daardoor enkele gevolgen:

[1] Onder systemen in een wagen verstaat men die systemen waarvoor elektronica en software nodig is om deze operationeel te maken

- Bedrijven worden opgedeeld in verschillende divisies volgens bovengenoemde classificatie

- Een voertuig wordt volgens bovengenoemde classificatie onderverdeeld in verschillende netwerken

Dit laatste gevolg wordt mooi geïllustreerd in ‘Figuur 2-1’, waar een overzicht wordt gegeven van de verschillende voertuignetwerken aanwezig in een BMW 3-reeks uit 2006.



Figuur 2-1: BMW 3-reeks "Bus System Overview"

We onderscheiden in bovenstaande figuur volgende netwerken met hun respectievelijke afkortingen:

- PT-CAN (Power Train-CAN)

- MOST, is een ringnetwerk welke exclusief gebruikt wordt voor multimediatoepassingen

- K-CAN (Comfort-CAN)

- LIN, is netwerk bestaande uit maar 1 draad (met de wagen als massa). Het is daardoor een zeer eenvoudig netwerk, welke traag is en redelijk wat fouten heeft. Het wordt dan ook gebruikt voor voertuigsystemen waar betrouwbaarheid niet de grootste vereiste is, zoals het bedienen van de zijspiegels

- FlexRay, is het neusje van de zalm en met een snelheid van 10Mbit/s is het beduidend sneller dan CAN. Het wordt gebruikt daar waar de betrouwbaarheid zeer hoog moet zijn, bijvoorbeeld bij de dynamische ophanging of als backbone

- Ethernet, wordt steeds vaker gebruikt in voertuigen. Er wordt dan ook verwacht dat dit in de toekomst zal doorbreken, ter vervanging van de diversiteit aan voertuignetwerken [11].

De JBE-module die te zien is in 'Figuur 2-1', is een zogenoemde "gateway". Een gateway in de automotive verbindt meerdere netwerken met elkaar, om op die manier bijvoorbeeld boodschappen door te sturen van de PT-CAN naar de K-CAN. Op een gateway zit ook een apart klein CAN-netwerk naar de OBD-stekker.

Merk op dat er naast een CAN-netwerk nog andere soorten netwerken terug te vinden zijn in de huidige generatie voertuigen zoals FlexRay, LIN, MOST, … [12]. Hoewel deze netwerken zeer interessante materie zijn, wordt er in deze thesis niet verder op ingegaan daar de focus ligt op het gebruik van de CAN-bus. In wat volgt wordt er dieper ingegaan op het gebruik van de onderste twee lagen van het OSI-model volgens de ISO-11898 norm.

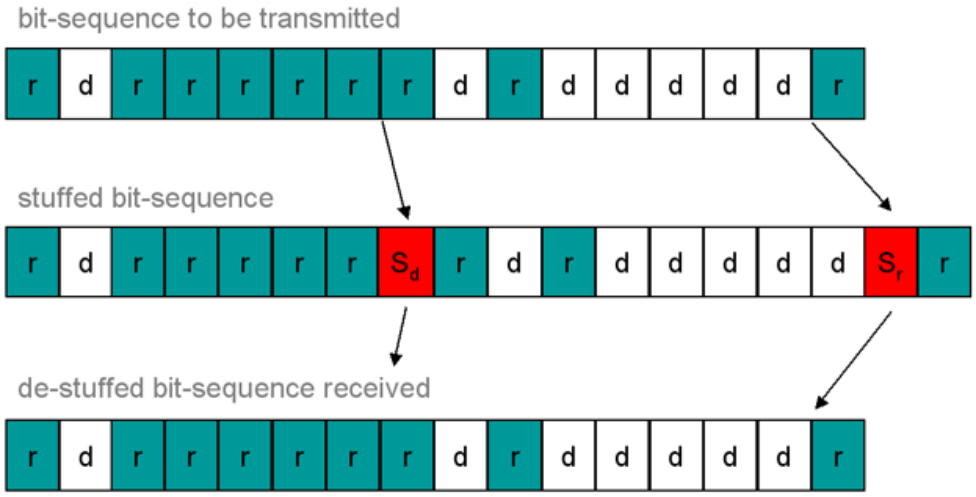
2.1.2 De fysische laag

2.1.2.1 Bit codering en bit stuffing

CAN maakt gebruik van de NRZ-modulatiemethode. Deze zeer eenvoudige techniek bestaat er simpelweg in dat een 1 wordt voorgesteld door een 1, en een 0 wordt voorgesteld door een 0. Er gebeurt met andere woorden geen modulatie, waardoor er slechts 2 toestanden mogelijk zijn gedurende 1 bittijd. Deze 2 mogelijke toestanden worden dominant en recessief genoemd.

Een nadeel van de NRZ-modulatiemethode is het gebrek aan flanken indien de bus zich gedurende een lange tijd in dezelfde toestand bevindt. Wanneer er te veel bits met eenzelfde polariteit op de bus worden geplaatst, zijn er onvoldoende neergaande flanken om de CAN-modules te synchroniseren. Dit probleem wordt verholpen door na iedere reeks van 5 bits met eenzelfde logisch niveau, een complementaire bit tussen te voegen alvorens het bericht te verzenden. Deze techniek wordt "bit stuffing" genoemd, waarbij de toegevoegde bit de "stuff bit" wordt genoemd. De toegevoegde bit wordt bij ontvangst uiteraard opnieuw verwijderd.

'Figuur 2-2' toont hiervan een voorbeeld. Bovenaan wordt de originele datastroom voorgesteld, welke een node wenst te versturen. De tweede datastroom is deze waar een stuff bit aan toegevoegd wordt, na 5 dominante of recessieve bits. Tot slot wordt de stuff bit terug verwijderd bij de ontvangende node, wat te zien is bij de derde datastroom. Uiteraard dienen de eerste en de laatste datastroom aan elkaar gelijk te zijn.



Figuur 2-2: Bit stuffing[2]

d = dominante data bit
r = recessieve data bit
S_d = dominante stuff bit
S_r = recessieve stuff bit

2.1.2.2 Bittiming en synchronisatie

In een CAN-netwerk heeft elke node zijn eigen klok. Door toedoen van kleine temperatuur- en spanningswijzigingen kunnen er kleine afwijkingen optreden tussen de verschillende klokken. Toch moeten alle nodes aanwezig op de CAN-bus gesynchroniseerd zijn op de data die op dat ogenblik verstuurd wordt via de bus. Mocht dit niet het geval zijn, zou de gegevensoverdracht of de arbitrage mislukken.

Om dit alles in goede banen te leiden, wordt de bittijd onderverdeeld in vier verschillende segmenten, zoals voorgesteld in 'Figuur 2-3':

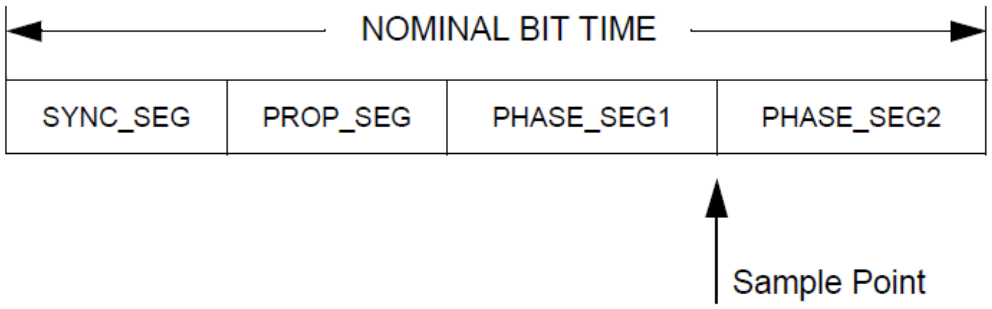
- Synchronization segment: Indien de stijgende of dalende flank in dit segment valt, dan zijn de nodes gesynchroniseerd.

- Propagation segment: Wegens de soms grote afstand tussen twee nodes, kunnen vertragingen optreden. Deze fysische vertragingen worden in dit segment opgevangen.

- Phase buffer segmenten: Door de segmenten in te korten of te verlengen, kan het sample point verschoven worden, om zo faseverschuivingen te compenseren en hersynchronisatie mogelijk te maken.

- Sample point: Dit punt ligt tussen de twee fasebuffersegmenten, en is het moment waarop de waarde van een bit ingelezen wordt.

[2] http://www.softing.com/home/en/industrial-automation/products/can-bus/more-can-bus/bit-encoding/bit-stuffing-rule.php



Figuur 2-3: CAN bittijd

Verder wordt elk segment opgedeeld in een programmeerbaar aantal tijdskwanta. Elke node mag zelf beslissen over de lengte van een tijdskwantum, afhankelijk van de periode van de interne klok en de programmeerbare BRP. De waarde van de BRP kan gelegen zijn tussen 1 en 32.

Het synchroniseren kan verlopen op twee manieren:

- Harde synchronisatie: Deze synchronisatie vindt plaats na een idle-toestand van de bus, gekenmerkt door een interframe space bestaande uit een recessieve bits. Daar de start van een frame steeds dominant is, zullen de ontvangers zich synchroniseren op deze overgang.

- Hersynchronisatie: Gedurende de transmissie kan de synchronisatie verloren gaan. Door het verlengen of het inkorten van de bittijd met een maximale lengte bepaald door RJW, kan ervoor gezorgd worden dat de volgende bitwisseling terug in het synchronization segment valt.

2.1.2.3 High Speed CAN (ISO-11898-2) en Low Speed CAN (ISO-11898-3)

Zowel High Speed CAN als Low Speed CAN maken gebruik van een twisted pair kabel, met onder andere volgende vereiste specificaties:

- Shielded of unshielded twisted pair kabel

- Nominale impedantie van 120 Ω

- Lijnvertraging van 5 nsec/m

CAN maakt steeds gebruik van differentieel signalen om data te versturen. De spanningsniveaus van deze signalen verschillen echter tussen High Speed CAN en Low Speed CAN, zoals onderstaande tabellen duidelijk maken.

	Recessief	*Dominant*
CAN-H	2,50 V	3,50 V
CAN-L	2,50 V	1,50 V

Tabel 2-2: Spanningsniveaus bij High Speed CAN

	Recessief	*Dominant*
CAN-H	5,00 V	1,40 V
CAN-L	0 V	3,60 V

Tabel 2-3: Spanningsniveaus bij Low Speed CAN

High Speed CAN kan snelheden halen tot 1Mbit/s met een theoretisch maximale buslengte van 40 m. Aan beide uiteinden van de bus dienen afsluitweerstanden van 120 Ω geplaatst te worden, om reflecties te verminderen. High Speed CAN is bovendien de meest gebruikte vorm van CAN.

Low Speed CAN heeft in tegenstelling tot High Speed CAN een maximale snelheid van slechts 125 kbit/s, en dit met een maximale buslengte van zo'n 500 m. Inderdaad, de maximale buslengte is omgekeerd evenredig met de datasnelheid van de bus.

Tot slot dient nog vermeld te worden dat er nog twee andere protocollen bestaan, naast deze zonet besproken, zijnde:

- Single Wire CAN, beschreven in de SAE J2411 standaard
- Point-to-Point CAN, beschreven in de ISO 11992 standaard

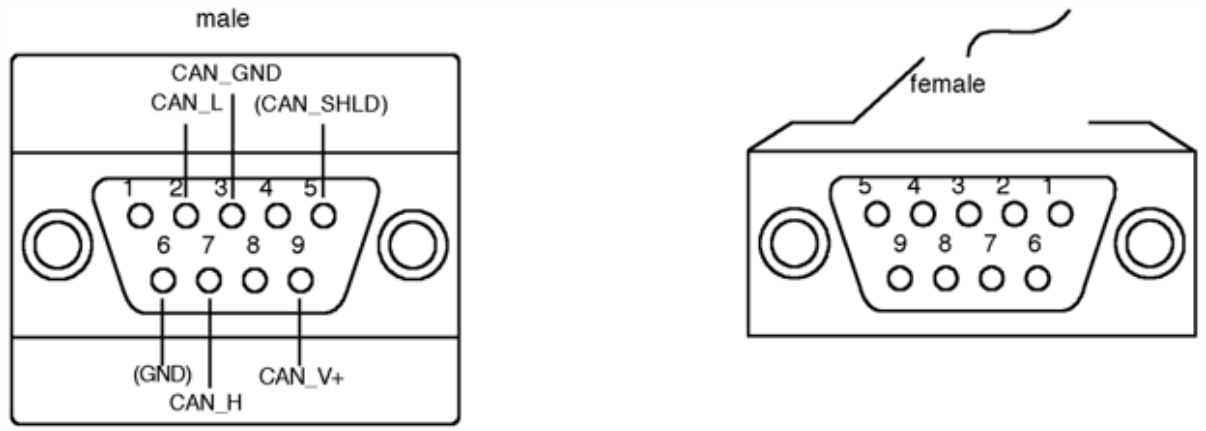
Daar deze twee protocollen niet aan bod komen in deze thesis, wordt hier niet verder op ingegaan.

2.1.2.4 Connectoren

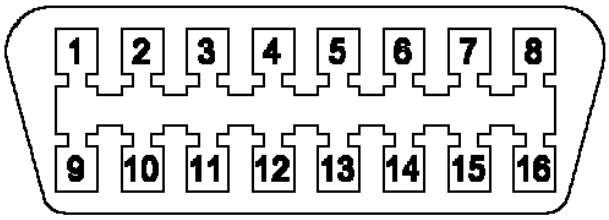
Het CAN-protocol definieert het fysische medium niet, waardoor een vrije keuze mogelijk is. Doch zijn er enkele standaarden ontworpen om enige uniformiteit te bekomen, waarvan hier de belangrijkste worden aangehaald.

De meest gebruikte connector is de 9-pin D-sub connector, zoals afgebeeld in 'Figuur 2-4', waarbij de allocatie van de pinnen is gedefinieerd volgens de DS-102 CiA standaard.

In deze thesis werd er echter meer gebruik gemaakt van de OBD-II connector. Het betreft een DLC3 connector, zoals afgebeeld in 'Figuur 2-5', waarbij de allocatie van de pinnen is gedefinieerd volgens de J1962 SAE standaard. Deze connector bevindt zich meestal onder het stuurwiel, of in de dichte omgeving van de bestuurderszijde, en wordt gebruikt voor diagnoseanalyse van het voertuig.



Figuur 2-4: 9-pin D-sub connector (CAN)



Figuur 2-5: DLC3 connector (CAN)

Pin	*Beschrijving*
4	Chassis Ground
5	Signal Ground
6	CAN-H (J2284)
7	K Line (ISO 9141-2)
14	CAN-L (J2284)
15	L Line (ISO 9141-2)
16	Battery Power

2.1.3 De datalink laag

2.1.3.1 Multimaster berichtgeoriënteerde communicatie

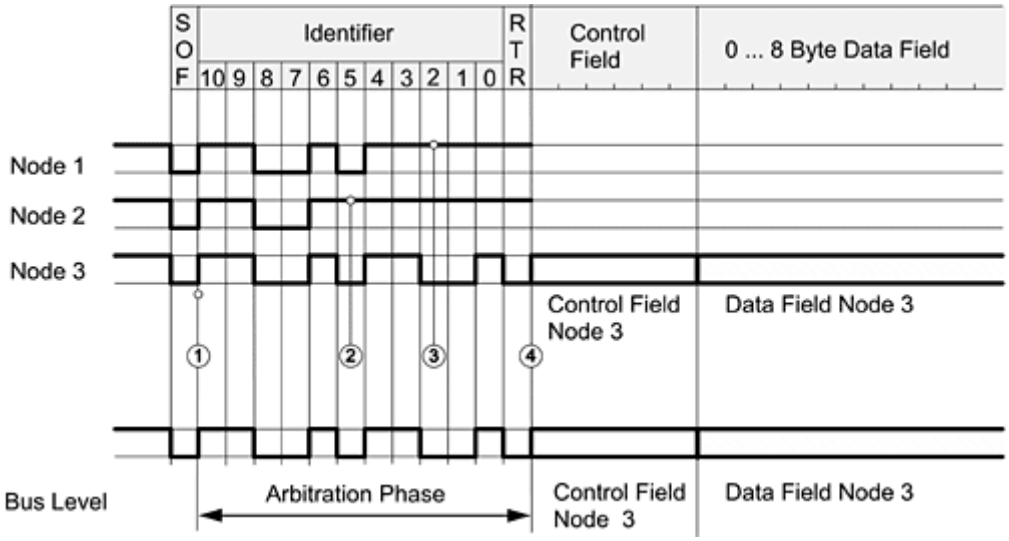
Het CAN-concept is gebaseerd op het multimaster-principe. Dit principe laat iedere node toe om op eigen initiatief berichten op de bus te plaatsen.

Verder is het CAN-concept gebaseerd op berichtgeoriënteerde communicatie. Dit wil zeggen dat een bericht niet getypeerd wordt met source en destination adressen, maar door een unieke identifier. Dit impliceert dat CAN gebruik maakt van het broadcast principe. Een bericht dat verstuurd wordt door een node, bestemd voor een andere node, zal zodus ontvangen worden door alle nodes op het netwerk. Via een interne lokale filter, een acceptance filter genoemd, beslist elke node voor zichzelf of het bericht voor hem bestemd was en indien verder verwerking van het bericht noodzakelijk is.

2.1.3.2 CSMA/BA

Het CAN protocol maakt gebruik van het CSMA/BA principe om toegang tot de bus te regelen. Zoals de naam zegt zal een node die een bericht wenst te verzenden, eerst luisteren of de bus in gebruik is (Carrier Sense). Wanneer dit het geval is, zal de node enige tijd wachten alvorens opnieuw een poging te ondernemen tot het verzenden van zijn dataframe. Multiple Access duidt erop dat er meerdere nodes op de bus aanwezig zijn.

Het Bitwise Arbitration mechanisme bij CAN maakt gebruik van het verschil tussen dominante (0) en recessieve (1) niveau's in een dataframe. Hierbij kan een dominante bustoestand een recessieve bustoestand overschrijven. 'Figuur 2-6' illustreert dit mechanisme. Bij de start zullen node 1, 2 en 3 simultaan starten met de arbitrage van de bus. Op punt 2 merkt node 2 dat de bus zich in een dominante toestand bevindt, in plaats van de eigen recessieve toestand, waardoor hij de arbitrage verliest. Op punt 3 doet dezelfde situatie zich voor, ditmaal voor node 1, waardoor deze eveneens de arbitrage verliest. Op het einde van de arbitrage fase zal enkel node 3 verder gaan met het versturen van zijn bericht, daar deze node de arbitrage gewonnen heeft.



Figuur 2-6: Bitwise Arbritation mechanisme (CAN)[3]

Merk op dat enkel het frame-ID gebruikt wordt voor de arbitrage. Een bijkomend gevolg is dat het frame met het laagste ID-nummer, de hoogste prioriteit heeft.

2.1.3.3 CAN frames

Een CAN-bericht wordt in een zogenoemd frame verstuurd. Het CAN-protocol definieert vier verschillende types, zijnde:

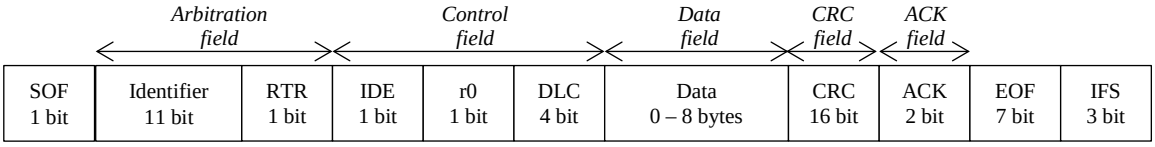
- Data frame, waarin de actuele data wordt verstuurd
- Remote frame, waarmee een node een verzoek doet aan een andere node tot het verzenden van data met een specifieke identifier. Dergelijk frame bevat dus geen data
- Error frame, waarmee een node een fout op de bus signaleert
- Overload frame, waarmee een node een aanvraag doet om de datarate van de bus te verlagen om zo meer verwerkingstijd te verkrijgen

[3] http://www.ixxat.com/can-controller-area-network-introduction_en.html

Deze uiteenzetting zal zich echter beperken tot de bespreking van het data en het remote frame, daar enkele deze frames relevant zijn voor deze thesis [12].

Data frame

'Figuur 2-7' illustreert het "CAN standaard data frame", ook wel gekend als "CAN2.0 A".



Figuur 2-7: CAN standaard frame (CAN2.0 A)

SOF (Start of Frame):

> De start van een frame begint steeds met een dominante bit, welke alle ontvangende nodes attent maakt op de start van een dataoverdracht. Deze nieuwe dataoverdracht kan enkel gestart worden indien de bus zich tot dan in de rusttoestand bevond.

Arbitration field:

> Dit veld bestaat uit de 11 bit identifier welke het adres van het bericht voorstelt zodat andere nodes kunnen afleiden of dit bericht voor hun bedoeld is of niet. De identifier wordt bij een data frame steeds gevolgd door een dominante RTR-bit (Remote Transmission Request-bit).

Control field:

> De IDE-bit (Identifier Extension-bit) geeft weer welk dataformaat er verstuurd wordt, standaard of extended. In het geval van een standaard bericht zal deze bit dominant zijn, in het andere geval recessief. Het CAN extended dataframe komt later in deze uiteenzetting aan bod.
>
> De r0-bit is een gereserveerde bit voor eventuele toekomstige uitbreidingen, en is steeds recessief.
>
> Het DLC-veld (Data Length Code-veld), bestaande uit 4 bits, geeft aan uit hoeveel bytes het dataveld bestaat.

Data field:

> Dit veld bevat de eigenlijke te verzenden data. Deze data kan 0 tot 8 bytes groot zijn, waarbij de MSB eerst wordt verzonden.

CRC field (Cyclic Redundancy Check-veld):

> Het CRC-veld bestaat uit een veld van 15 CRC-bits, gevolgd door een recessieve bit genaamd de "CRC delimiter". Met dit veld wordt gecontroleerd of het verstuurde bericht correct ontvangen is.

De CRC wordt berekend door alle voorgaande bits (van “SOF” tot en met het “Data field”) te beschouwen als een groot binair getal. Dit binaire getal wordt gedeeld door de constante 1100010110011001, welke gekend is door elke node op de bus. De restwaarde van deze deling is de CRC. Deze berekening wordt zowel uitgevoerd bij de versturende als bij de ontvangende node. Wanneer deze berekeningen overeenstemmen, is het bericht correct verzonden.

ACK field (Acknowledge-veld):

Dit veld bestaat uit 2 bits waarvan de eerste een bevestigingsbit (ACK slot) en de tweede een bevestigingsafsluiting (ACK delimiter) is. Door de verzendende node worden deze beide bits als recessieve bits verzonden, terwijl ontvangende nodes de eerste bit dominant kunnen maken ter bevestiging van een correct ontvangen frame.

EOF (End Of Frame):

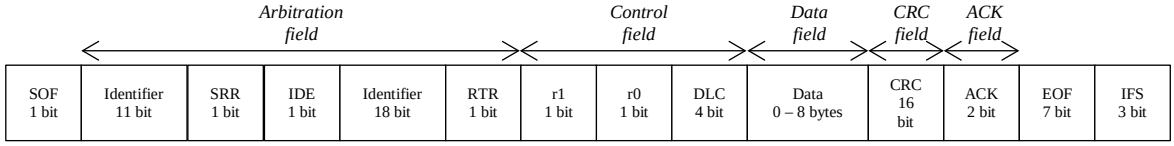
Het einde van een dataframe bestaat steeds uit 7 opeenvolgende recessieve bits.

IFS (InterFrame Space):

Dit verplichte veld tussen twee opeenvolgende dataframes moet steeds bestaan uit minstens drie opeenvolgende recessieve bits.

Merk op dat, voor er een nieuw dataframe kan worden verzonden, er steeds op zijn minst 11 recessieve bits op de bus moeten geplaatst zijn (ACK delimiter, EOF en IFS).

Zoals reeds aangehaald, bestaat er ook een “CAN extended data frame” (‘Figuur 2-8’) welke ook wel gekend is als “CAN2.0 B”. Het grote verschil met het standaard frame is de lengte van het identifierveld. Bij een standaard frame bestaat dit veld uit 11 bits, terwijl dit bij een extended frame bestaat uit 29 bits. Deze 29 bits is opgesplitst in een 11 bit identifier (base identifier) en een 18 bit uitbreiding (identifier extension).



Figuur 2-8: CAN extended frame (CAN2.0 B)

Zoals reeds gezegd, wordt het onderscheid tussen een standaard en een extended frame gemaakt door de IDE-bit. Bij een standaard frame is deze bit dominant, terwijl deze bij een extended frame recessief is. Merk daardoor op dat een standaard frame steeds voorrang heeft op een extended frame.

Een extended frame heeft verder nog enkele velden die verschillend zijn van een standaard frame.

SRR (Substitute Remote Request):

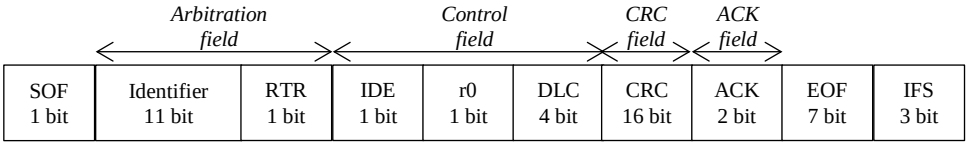
Na de 11 bit identifier volgt de SRR-bit, welke in de plaats komt te staan van de RTR-bit bij een standaard frame. Deze bit zal steeds een recessieve waarde hebben, waardoor een standaard frame steeds voorrang krijgt op een extended frame.

Control field:

Een extra recessieve bit “r1” is hier toegevoegd, welke tevens een gereserveerde bit is.

Remote frame

‘Figuur 2-9’ illustreert het “CAN remote frame”.



Figuur 2-9: CAN remote frame

Dit frame komt grotendeels overeen met een data frame, maar heeft geen data-veld daar dit frame informatie opvraagt in plaats van te verzenden. Verder zijn er nog enkele kleine verschillen.

Arbitration field:

In tegenstelling tot een data frame, is de identifier die in het frame voorkomt de identifier van het gewenste bericht. De node welke de data bevat eigen aan deze identifier, zal de gevraagde data versturen. De RTR-bit zal in het geval van een remote frame steeds recessief zijn, opnieuw omwille van arbitrage met een standaard frame.

Control field:

Daar er geen data veld aanwezig is, zal de inhoud van dit veld overeenkomen met de lengte van de op te vragen data.

2.2 Onderzoeken van voertuigsystemen en hun bijhorende sensoren

De huidige technologische ontwikkelingen hebben er toe geleid dat er steeds meer elektronische voertuigsystemen, met hun desbetreffende sensoren, aanwezig zijn in een voertuig ten voordele van veiligheid en prestaties. Terwijl de nauwkeurigheid van deze systemen toeneemt, daalt hun kostprijs, waardoor steeds meer klasse van voertuigen deze standaard aanwezig hebben.

In ‘Tabel 2-1’ werden al enkele voertuigsystemen aangehaald die tegenwoordig aanwezig zijn in een voertuig. Daar deze systemen constant de omgeving monitoren, is het aannemelijk dat de sensoren op zeer regelmatige basis hun gegevens op de CAN-bus plaatsen. Indien deze gegevens kunnen onderschept worden op de CAN-bus, is het eventueel mogelijk om een algoritme te bedenken dat zich enkel voordoet indien het voertuig rijdt in een put in het wegdek.

Wat volgt is een bespreking van de, voor deze thesis, belangrijkste voertuigsystemen met hun overeenkomstige sensoren.

2.2.1 Anti-Lock Braking System

Een Anti-Lock Braking System (ABS) [13] is een veiligheidssysteem in voertuigen welke ervoor zorgt dat tijdens het remmen de wielen niet blokkeren, zodat voldoende grip met het wegdek behouden blijft.

Uit onderzoek blijkt een remsysteem het meest effectief te zijn indien de wielsnelheden tijdens het remmen 85 tot 90% bedragen van de voertuigsnelheid. Het verschil van 15% wordt het percentage slip genoemd van een bepaald wiel. Indien tijdens het remmen, de remkracht zo groot wordt dat de wielen blokkeren, wordt percentage slip verkregen van 100%. Een ABS vermijdt dit, zodat de remefficiëntie verhoogd wordt.

Dit is echter niet de hoofdtaak van een ABS. Indien een wiel geblokkeerd is biedt het geen laterale besturing meer aan het voertuig, en in het geval van meerdere geblokkeerde wielen kan het voertuig beginnen tollen. Stel bijvoorbeeld dat beide achterwielen blokkeren, dan zal het voertuig de neiging krijgen om de achterkant naar voren te brengen. Bij geblokkeerde voorwielen zal het voertuig onbestuurbaar worden. Dit is nu net de hoofdtaak van een ABS, ervoor zorgen dat een voertuig te allen tijde bestuurbaar blijft en zo de gewenste richting uitgaat tijdens het remmen.

Er is aangetoond [14] dat bij slechte wegomstandigheden zoals watergladheid, ijsvorming, … een ABS zorgt voor een verhoogde stuurcontrole en een kortere remweg. Echter, in het geval van sneeuw of grind zal een geblokkeerd wiel een hogere remefficiëntie hebben in vergelijking met een ABS, daar sneeuw- of grindophopingen net voor het geblokkeerde wiel het voertuig sneller afremmen.

Dit alles wordt mogelijk gemaakt door het gebruik van wielsnelheidssensoren op elk individueel wiel of op de differentieel van elke as, een ECU, een pomp en enkele kleppen. Er zijn twee type wielsnelheidssensoren op de markt, waarbij hun werking licht verschilt:

- Passieve sensoren: Zij bestaan uit een zacht ijzeren, permanent magnetische pin waarrond een fijne koperdraadspoel is gewonden. Deze sensor wordt dicht bij een tandenkrans geplaatst, welke op zijn beurt gemonteerd wordt op het ronddraaiende wiel. Wanneer de tandenkrans passeert langs de sensor, zullen de wisselende tanden een kleine inductieve spanning induceren in de koperen winding. Deze sinusvormige spanning wordt doorgestuurd naar de ECU, welke een maat is voor de wielsnelheid.

- Actieve sensoren (‘Figuur 2-10’): Dit zijn vandaag de dag de meest voorkomende sensoren bij een ABS. Het zijn Hall-sensoren werkende op een referentiespanning van 5 tot 12V, geleverd door de ECU. Opnieuw wordt bij dit type van sensoren een tandenkrans gemonteerd op het ronddraaiende wiel. De tandenkrans is in dit geval een multipool (noord-zuid, noord-zuid, …) magnetische ring, waardoor de wisselende tanden een magnetische flux induceren die voldoende groot is om de basis van de inwendige transistor aan te sturen. In tegenstelling tot de passieve sensoren, zal het gegenereerde signaal blokvormig zijn met steeds eenzelfde amplitude, waarbij de frequentie van dit blokvormig signaal een maat is voor de wielsnelheid.

Figuur 2-10: ABS met actieve wielsnelheidssensoren[4]

Indien wordt verondersteld dat elk wiel voorzien is van een wielsnelheidssensor, wat vandaag de dag bij de meeste voertuigen het geval is, kan de werking als volgt worden beschreven. De ECU kan op basis van de ontvangen signalen, afkomstig van de wielsnelheidssensoren, de individuele wielsnelheden monitoren. Indien de ECU merkt dat een individueel wiel een bepaalde slipdrempel overschrijdt, dus dat een individueel wiel trager draait, zal deze de remkracht op dit individueel wiel verlagen. Dit wordt gedaan door de pomp en kleppen zo bij te regelen, dat de rotatie van het desbetreffende wiel wordt verhoogd. Wanneer het wiel terug aan de gewenste snelheid roteert, en dus de optimale slip bereikt is, zal de remkracht opnieuw aangebracht worden (indien het rempedaal nog steeds is ingedrukt). Dit proces wordt meerdere keren herhaald, tot zo’n 15 keer per seconde, en wordt door de bestuurder ervaren

[4] http://repairpal.com/abs-wheel-speed-sensor

als een schokkend rempedaal. Dit schokkend effect is een gevolg van het pompend effect op de remkracht (verhogen/verlagen/verhogen) welke gecontroleerd wordt door de ABS.

ABS is een van de eerste elektronische systemen in een wagen, en is tegenwoordig standaard op alle in de EU verkochte voertuigen. Het is daarom zeer interessant om bij de detectie van putten in het wegdek, rekening te houden met de steeds aanwezige wielsnelheidssensoren. Daar een ABS snel moet ingrijpen in een noodsituatie, is het aannemelijk dat deze informatie met een hoge frequentie op de bus wordt geplaatst. Tevens worden deze wielsnelheidssensoren aangewend voor andere elektronische systemen, daar ABS de basis is geweest voor enkele meer geavanceerde elektronische systemen, waardoor deze informatie zeker terug te vinden is bij het loggen van de CAN-bus.

2.2.2 Traction Control System

Terwijl ABS ervoor zorgt dat een voertuig bestuurbaar blijft tijdens het remmen, zal een Traction Control System (TCS) [15] hetzelfde nastreven tijdens het accelereren.

Zoals reeds gezegd zal een ABS de remkracht kortstondig verlagen wanneer de ABS-ECU merkt dat een wiel geblokkeerd wordt. Bij TCS zal er kortstondig remkracht aangebracht worden op een van de aangedreven wielen indien TCS merkt dat een wiel sneller draait, en dus meer slip ondervindt, dan het andere. Van zodra het desbetreffende wiel een wielsnelheid heeft overeenkomstig met een gemiddelde van de andere wielsnelheden, wordt deze remkracht verwijderd. Opnieuw kan dit proces verschillende keren herhaald worden.

De ECU's van TCS en ABS werken in dergelijk geval nauw samen. Indien TCS beslist om een wiel af te remmen, dan zal de TCS-ECU een signaal geven naar de ABS-ECU om actie op het desbetreffende wiel te ondernemen. Tegenwoordig worden ABS en TCS vaak gecombineerd in eenzelfde ECU. Merk wel op dat een TCS enkel werkende is bij het accelereren, en dat zijn functie inactief is wanneer het rempedaal ingedrukt wordt. Sommige TCS systemen hebben ook de mogelijkheid om te communiceren met de ECU's van de motor en de transmissie (kortweg PCM genoemd in de automotive), waardoor het motorvermogen kan gereduceerd worden indien tijdens het accelereren de wielen onderhevig zijn aan slip. Het reduceren van het motorvermogen kan bijvoorbeeld gebeuren door de frequentie van de ontstekingsvonk te verminderen, brandstoftoevoer te verminderen, …

De laatste jaren is TCS eveneens standaard aanwezig in voertuigen verkocht in de EU. Daar het de omgekeerde werking heeft van ABS, gebruikt het dezelfde sensoren. Hierdoor levert dit systeem voor deze thesis geen meerwaarde wat betreft bruikbare sensorgegevens.

2.2.3 Electronic Stability Program

Een verdere uitbreiding van ABS en TCS is het Electronic Stability Program (ESP) [16]. ESP is een elektronisch systeem dat onder- en overstuur probeert te voorkomen, en zo een bestuurder helpt om de controle over het sturen te behouden.

ESP zal constant berekenen of een voertuig de richting uitgaat de welke de bestuurder aangeeft. Hiervoor gebruikt ESP, naast de wielsnelheidssensoren afkomstig van het ABS, bijkomend een stuurwielhoeksensor, een acceleratiesensor en een yaw-sensor. Een yaw-sensor is een sensor die geplaatst wordt in de omgeving van het zwaartepunt van een voertuig, en welke de verticale rotatie van een voertuig meet.

De ESP-ECU meet de gewenste stuurrichting aan de hand van een stuurwielhoeksensor. Door het verschil in wielsnelheden, afkomstig van het ABS, te meten wordt berekend of het voertuig de juiste richting uitgaat. Deze berekening wordt verder verfijnd door informatie van de yaw-sensor en deze van de acceleratiesensor welke een indicatie geeft van de laterale slip.

Wanneer ESP merkt dat een voertuig mogelijk de stuurcontrole zal verliezen, dan ESP ingrijpen om zo het voertuig terug op het gewenste pad te brengen. Dit kan door een signaal te sturen naar de ECU van het ABS met de opdracht om een of meerdere wielen af te remmen, al naargelang de berekeningen van de ESP -ECU. Stel bijvoorbeeld een bocht naar rechts waarbij onderstuur optreedt, dan zal ESP de nodige remkracht aanbrengen op het rechterachterwiel, en mogelijk ook het rechtervoorwiel, om zo alsnog de bocht te kunnen nemen. Indien dit onvoldoende blijkt, kan de ESP-ECU ook een signaal sturen naar de ECU van het TCS om via deze weg het motorvermogen te verlagen. Merk op dat het ESP ook in werking zal treden in het geval de wielen doorslippen bij het accelereren, het TCS zodus in werking treedt, en het voertuig dreigt schuin te trekken.

Naast ESP zijn er ook andere benamingen in omloop, al naargelang de fabrikant, maar elk met dezelfde functie. Voorbeelden van deze andere benamingen zijn Electronic Stability Control (ESC) en Dynamic Stability Control (DSC).

De laatste twee jaar worden voertuigen vanaf de middenklasse standaard uitgerust met ESP. Bij de goedkopere mini en compacte autoklasse is dit echter nog niet het geval. Met de neigende trend om alle klasse voertuigen standaard uit te rusten met ESP, is het zeker van belang om deze sensoren verder te onderzoeken. Het is aannemelijk dat de stuurwielhoeksensor en de yaw-sensor uitwijkingen zullen vertonen bij het rijden in een put. Een probleem kan zijn dat de informatie van deze sensoren niet op de CAN-bus aanwezig is, daar deze in de meeste voertuigen niet gebruikt wordt door andere ECU’s.

2.2.4 Active Steering

Active Steering is een systeem waarbij de relatie tussen de hoek van het stuurwiel en deze van de wielen veranderd, afhankelijk van de huidige voertuigsnelheid.

De basis van het systeem is een planetair tandwielstelsel dat aanwezig is in de stuurkolom. Een elektrische motor zorgt vervolgens voor de aanpassing van de stuurhoek van de voorwielen, in proportie met de voertuigsnelheid. Bij zeer lage snelheden, bijvoorbeeld wanneer een voertuig moet parkeren of bij scherpe trage bochten, zal Active Steering de stuurhoek vergroten. De voorwielen zullen zo zeer snel reageren op kleine bewegingen van het stuurwiel, wat ervoor zorgt dat de bestuurder kan manoeuvreren tussen kleine ruimtes zonder meerdere rotaties te moeten maken met het stuurwiel. Bij hoge snelheid zal Active Steering het aantal rotaties van het stuurwiel verhogen voor het bekomen van eenzelfde stuurhoek. Dit zorgt ervoor dat de bestuurder preciezer kan sturen wat meer stabiliteit en comfort moet bieden.

Active Steering kan ook samenwerken met ESP. Bijvoorbeeld, in het geval van overstuur kan de ECU van ESP een signaal geven aan de ECU van Active Steering om de stuurhoek van de wielen te veranderen en zo tegen te sturen.

Bij Active Steering wordt de directe connectie tussen het stuurwiel en de voorwielen echter niet verbroken. Daardoor kan in het geval van een uitval van de elektronica, de operatie van

de elektrische motor uitgeschakeld worden, waardoor het planetair tandwielstelsel vergrendeld wordt en er met een vaste verhouding kan gestuurd worden.

Active Steering wordt momenteel enkel gebruikt bij BMW, en maakt geen gebruik van bijkomstige sensoren op de CAN-bus. Interessant voor de toekomst, wanneer andere constructeurs dit systeem zouden implementeren, is dat er een interactie bestaat tussen de ECU's van ESP en Active Steering, waardoor de kans dat de ESP-sensorinformatie te onderscheppen is op de CAN-bus verhoogt.

2.2.5 (Semi-)Active Suspension

Een (Semi-)Active Suspension, in het Nederlands een (semi-)actieve ophanging genaamd, is een ophangingssysteem waarbij de verticale beweging van de wielen wordt gecontroleerd door middel van een ECU.

Het systeem elimineert quasi de rolbeweging waaraan een voertuig onderhevig is tijdens het nemen van een bocht, en de verticale acceleratie tijdens het accelereren en het remmen, waardoor de rijkwaliteit verhoogd wordt. Bijkomend wordt het besturen van een voertuig verbeterd door een betere tractie en controle, daar de (semi-)actieve ophanging ervoor zorgt dat elk wiel loodrecht op het wegdek staat.

Het verschil tussen een actieve en semi-actieve ophanging ligt hem in de werking. Een actieve ophanging zal energie toevoegen aan de ophanging op de bewegingen op te vangen. Het chassis wordt op elk afzonderlijk wiel verlaagd of verhoogd. Dit kan zowel gebeuren op een hydraulische manier als met behulp van elektromagnetische motoren. Bij een semi-actieve ophanging wordt er geen energie toegevoegd om de bewegingen op te vangen. Bij dit type ophanging wordt de visceuze dempingscoëfficiënt aangepast. Dit wordt tegenwoordig bekomen door via een elektromagneet de in de vloeistof metalen deeltjes al dan niet te activeren, waardoor de stijfheid van de ophanging wordt geregeld. Door deze eenvoudigere uitvoering is een semi-actieve ophanging goedkoper dan een actieve ophanging.

Voor beide type ophangingen heeft de ECU, die de ophanging regelt, allerhande gegevens nodig afkomstig van verschillende sensoren. Onder deze sensoren vallen de wielsnelheidssensoren, stuurwielhoeksensor, acceleratiesensoren, yaw-sensor en sensoren geplaatst op de ophanging. Deze laatste categorie sensoren omvatten sensoren welke de hoogte van het voertuig opmeten, en de acceleratie opmeten waarmee de ophanging wordt ingedrukt.

Naast de acceleratiesensoren (lateraal, longitudinaal) en yaw-sensor, zijn de bijkomende sensoren die geplaatst worden op de ophanging, om de hoogte van het voertuig te bepalen en de acceleratie van de ophanging op te meten, zeer interessant in het licht van deze thesis. Deze sensoren staan het dichtst bij het wegdek, en zullen mogelijk de beste indicatie geven van een defect in het wegdek. Ook de frequentie waarmee de sensorgegevens op de CAN-bus worden geplaatst zou geen probleem zijn. Echter, opnieuw worden deze sensorgegevens enkele aangewend door de ECU welke de ophanging regelt en door geen enkel ander systeem, waardoor de kans klein is dat de sensorgegevens op de CAN-bus worden geplaatst.

(Semi-)Active Suspension wordt voorlopig enkel maar toegepast bij voertuigen van een hogere prijsklasse. Elke constructeur heeft bovendien zijn eigen benaming voor dit type ophanging. Volgend is een kort overzicht van de belangrijkste constructeurs en hun (semi-) actieve ophangingen.

Citroën

De *Hydractive 3* is een hydropneumatische ophanging welke gebruik maakt van de wielsnelheidssensoren en de sensoren die de hoogte van het voertuig bepalen. Dit systeem is standaard aanwezig op het model C5. De meer geavanceerde *Hydractive 3+* ophanging maakt gebruik van enkele bijkomende sensoren zoals stuurwielhoeksensoren, acceleratiesensoren (longitudinaal en lateraal), acceleratiesensoren op de ophanging, … Het is standaard verkrijgbaar op het C6 model, en op sommige C5 modellen (afhankelijk van de motoruitvoering).

Mercedes-Benz

Bij Mercedes-Benz wordt gebruik gemaakt van een actieve ophanging, genaamd *Active Body Control (ABC)*, welke op een vergelijkbare manier werkt als het *Hydractive 3+* systeem van Citroën. Het is enkel verkrijgbaar op de topmodellen van het Mercedes-Benz gamma, zoals de CL-class, S-class en SL-class.

Op de goedkopere modellen zoals de C-class, E-class en GLK-class, wordt een hydraulische variant genaamd *Agility Control* optioneel geleverd. *Agility Control* is een geavanceerd veer-demper systeem welke werkt zonder elektronica. Het is daarom zelfs goedkoper dan een semi-actieve ophanging.

BMW

BMW noemt zijn versie van een (semi-)actieve ophanging *Active Roll Stabilization (ARS)/Dynamic Drive*. Het is als optie verkrijgbaar bij de huidige 3-reeks, en standaard op de 5-reeks, X5 en X6 modellen.

Lexus

Het *Adaptive Variable Suspension (AVS)* bij Lexus is momenteel optioneel verkrijgbaar bij het RX-model, en standaard aanwezig bij het GS en LS-model.

Ford

Als optie op de Galaxy en de Mondeo is een (semi-)actieve ophanging mogelijk, welke bij Ford *Continuous Damping Control (CDC)* wordt genoemd.

Opel

FlexRide (IDS+), zoals Opel zijn versie van een (semi-)actieve ophanging kenbaar maakt, is momenteel enkel als optie verkrijgbaar op de modellen Astra en Insignia.

Volvo

Volvo gebruikt de benaming *Four-C Active Chassis*, en is momenteel standaard aanwezig op de modellen S60R en V70R.

2.2.6 Conclusie

Uit de voorgaande bespreking blijkt dat er tal van sensoren aanwezig zijn in een voertuig die kunnen aangewend worden om, via een bepaald algoritme, defecten in het wegdek te

detecteren. Het blijkt echter dat maar weinig van deze sensoren standaard aanwezig zijn in een voertuig, daar hun bijhorende systemen vaak als optie worden aangeboden door de constructeurs. Verder wil de aanwezigheid van deze systemen en hun bijhorende nuttige sensoren niet impliceren dat de sensorgegevens te onderscheppen zijn op de CAN-bus.

Daar het de bedoeling is van deze thesis om putten in het wegdek te detecteren met een zo algemeen mogelijk algoritme, toepasbaar in de meeste voertuigen van de huidige generatie, wordt er gefocust op de standaard aanwezige voertuigsystemen met hun bijhorende sensoren, namelijk:

- ABS: wielsnelheidssensoren
- TCS: wielsnelheidssensoren
- ESP: wielsnelheidssensoren, stuurwielhoeksensor, acceleratiesensor en yaw-sensor

In dit stadium van het onderzoek werd reeds een kleine simulatie uitgevoerd met het programma CarMaker [17], waarbij een voertuig een gesimuleerde route volgde. Het voertuig accelereerde hierbij eerst van 0 km/u naar 90 km/u, waarna het een rechte baan volgde met enkele putten in het wegdek. Het programma liet toe om enkele datavariabelen te loggen, waarbij volgende datavariabelen het meest interessant leken om een put in het wegdek te detecteren:

- Acceleratie: longitudinaal, lateraal en verticaal (m/s^2)
- Roll van de wagen (rad)
- Snelheid waarmee een vering wordt ingedrukt (m/s)
- Stuurhoek op de individuele voorwielen (rad)
- Wielsnelheden (rad/s)

Bedoeling was om reeds vroeg na te gaan of het überhaupt wel mogelijk is om met voertuiggegevens putten in een wegdek te detecteren. Na grondige analyse van voornoemde datavariabelen, die overeenkomstig zijn met gelijkaardige sensorgegevens, is er geconcludeerd om naar een algoritme te zoeken waarbij een combinatie van sensorgegevens gehanteerd wordt. De reden waarom deze manier van werken aangeraden is, kan het best aangetoond worden met een voorbeeld.

Bij het rijden in een put zal de verticale acceleratie een piek vertonen. Een drempelwaarde op bijvoorbeeld 5 m/s^2 zou kunnen betekenen dat, bij een overschrijding van deze waarde, er zich een put in het wegdek bevindt. Bij het accelereren wordt deze drempelwaarde echter ook overschreden, waardoor er zogezegd een put in het wegdek wordt gedetecteerd.

Door de verticale acceleratie te combineren met een parameter welke eveneens een piek vertoont bij het rijden in een put, maar niet bij het accelereren, zou een algoritme kunnen gevonden worden welke putten in een wegdek kan detecteren.

2.3 Onderzoeken van de uCAN-module

Een onderdeel van deze thesis is het koppelen van meetgegevens aan GPS-coördinaten en deze, in het geval van een defect aan het wegdek, draadloos te versturen naar een centraal punt. Om dit te realiseren werd gebruik gemaakt van een uCAN-module [4].

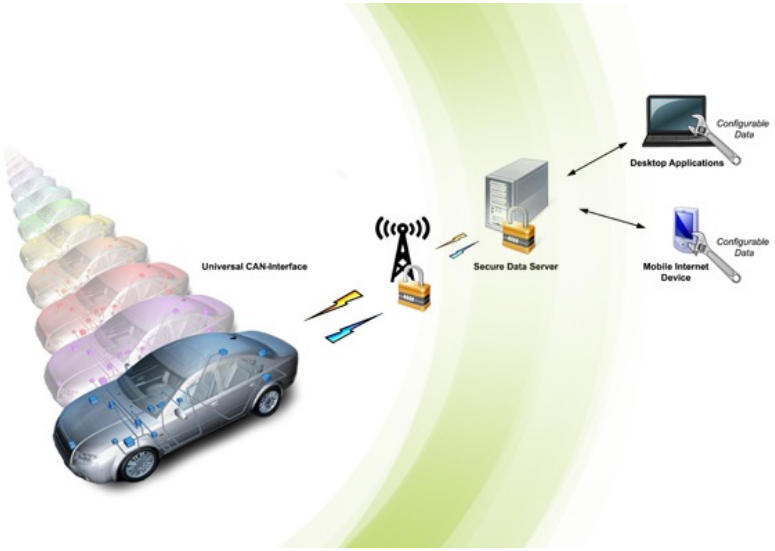
In wat volgt wordt een korte beschrijving gegeven van deze uCAN-module: zijn functionaliteit, wie het heeft ontwikkeld, en zijn beperkingen voor deze thesis.

2.3.1 Definitie

uCAN is in wezen een klein doosje, aangesloten op de CAN-bus van een voertuig, wiens primaire taak is *het maken van een verbinding tussen het voertuig/object en de applicatie van de leverancier van aftermarket-systemenen zoals ritregistratie, Ecodriving, eCall, bCall,…* [18].

2.3.2 Werking

De werking van de uCAN wordt geïllustreerd in ‘Figuur 2-11’.



Figuur 2-11: Werking uCAN[5]

In het voertuig wordt de uCAN’s universele In-car module ingebouwd, welke gebouwd is rond de ATOP-chip ontwikkeld door NXP [5]. De ruwe CAN-data afkomstig van de CAN-bus van het voertuig wordt via een beveiligde GPRS-verbinding doorgestuurd naar een centrale server. Op deze server staat een database met de CAN-gegevens van vrijwel alle auto’s op de markt [7]. De gegevens worden op de server geïnterpreteerd en doorgestuurd naar de verschillende applicaties. Via een back office toepassing kunnen de gebruikers toegang verkrijgen tot hun data en configureren zij hun In-car module aan de hand van de voertuiginformatie die zij willen ontvangen.

[5] http://www.ucan-solutions.com/nl/oplossingen/ucan-automotive/

2.3.3 Partners en hun bijdrage

uCAN is tot stand gekomen door een samenwerking van enkele bedrijven, met elk hun specifieke bijdrage. Wat volgt is een overzicht van deze coöperanten.

2.3.3.1 NXP Semiconductors

NXP Semiconductors levert de ATOP-chip [5], [19] voor de uCAN. ATOP (Automotive Telematics On-Board Unit Platform) is een black box met elektronica voor:

- Mobiele verbinding voor spraak- en dataconnectie
- GPS lokalisatie sensoren
- Verbinding met een voertuig zijn elektronische systemen (sensoren)
- Verwerkings- en opslagcapaciteit voor optionele applicaties

ATOP is origineel ontwikkeld voor het in de toekomst verplichte eCall [20], maar kan voor meerdere opties gebruikt worden, zoals: vehicle location tracking, remote diagnostics, fleet management, … De ATOP-chip levert bovendien ook bijkomende beveiliging om gegevens te beschermen.

Het is hiermee duidelijk dat de ATOP-chip een zeer groot aandeel heeft in de uCAN. Het lijkt erop dat de hele uCAN-module rond deze ATOP-chip gebouwd is, gezien de gelijkaardige functionaliteiten tussen de ATOP-chip (als standalone beschouwd) en de functionaliteiten van de uCAN-module.

2.3.3.2 LaQuSo

LaQuSo [8] is een samenwerkingsverband tussen enkele Nederlandse universiteiten. Deze spin-off staat in voor de beveiliging van de gegevens en zorgt dat de privacy van het systeem en zijn gebruikers verzekerd is.

2.3.3.3 Beijer Automotive

Beijer Automotive [7] heeft expertise in de verschillende soorten berichten op de CAN-bus, en dit voor de meest verkochte voertuigen binnen de EU. Dit alles dankzij een eigen database, welke opgebouwd is gedurende de voorbije 10 jaar door eigen reverse engineering.

Zij claimen Europa's grootste CAN-database te hebben, waarin zo goed als alle Europese automodellen aanwezig zijn. Elk signaal dat aanwezig is op de CAN-bus kan door Beijer Automotive aangeboden worden in de door de klant gewenste vorm/specificatie.

Naast deze database, hebben zij reeds geruime tijd kant en klare CAN-interfaces welke vergelijkbaar zijn met de uCAN-module, doch niet zo uitgebreid. Deze CAN-interfaces zijn meestal loggers van eenvoudige signalen in de wagen zoals snelheid, toerental, contact aan/uit, …

Uit deze informatie blijkt dat Beijer Automotive de tweede hoofdrolspeler is van het uCAN-module. Zij hebben de expertise in huis om interfaces aan te sluiten op de CAN-bus, en een uitgebreide database om signalen op de CAN-bus te herkennen.

2.3.3.4 TASS

TASS [6], tot 2007 onderdeel van Philips, levert embedded software voor technisch hoogwaardige producten. Ook voor de uCAN-module ontwikkelen zij de embedded software voor de aansturing van de boordmodule, welke zorgt voor het vertalen van de CAN-signalen naar bruikbare informatie en het verzenden via een GPRS-verbinding naar de centrale server. Verder zorgen zij ook voor de uCAN-serversoftware welke, zoals reeds eerder vermeld, zorgt voor de ontvangst en interpretatie van de gegevens en het doorsturen ervan naar de verschillende (aftermarket) applicaties.

Enkele reportages op de website van TASS [6] leveren ons wat meer informatie met het afgeronde pilotproject waarbij 120 taxi's en 50 ANWB-voertuigen voorzien werden van een uCAN-module. Dit pilotproject, bekend onder de naam "SMART in-car", is een onderdeel van het door de Nederlandse overheid gesubsidieerde project "Brabant in-car II" waarbij verschillende innovatieve in-car toepassingen werden getest. De geselecteerde voertuigen sturen periodisch CAN-gegevens door zoals:

- Status van de ruitenwissers
- Status van de lichten
- Status van de airconditioning
- Status van de remmen
- Toerental
- Het aantal gordels in gebruik
- Wielsnelheid/ABS

Dit laatste is voor deze thesis zeer interessant te noemen, daar de wielsnelheden kunnen gebruikt worden als één van de parameters om putten in een wegdek te detecteren. In dit pilotproject was het oorspronkelijk de bedoeling dat men de gladheid van een wegdek zou meten aan de hand van deze gegevens. Helaas is hierover in het eindrapport [21] niets over terug te vinden, waardoor eventueel mag besloten worden dat dit onderdeel van het pilotproject niet of onvoldoende geslaagd is.

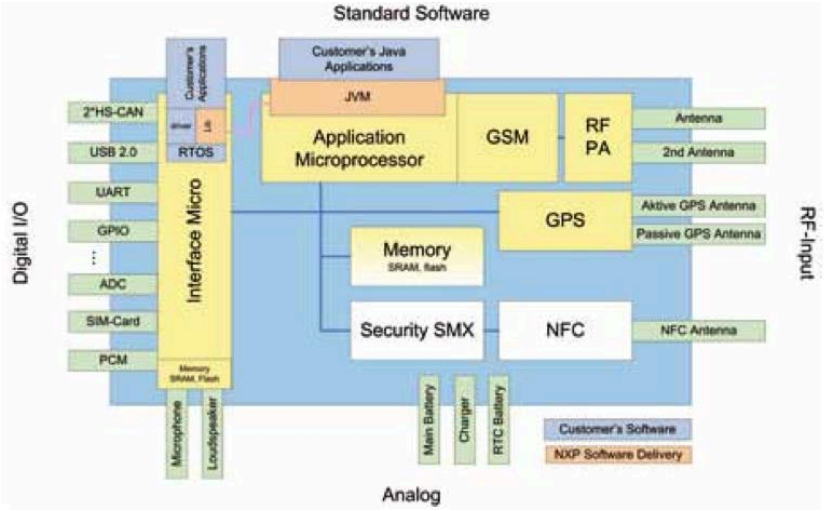
2.3.4 Conclusie

Daar de uCAN-module nog niet op de markt is en geen technische informatie wordt vrijgegeven, is het belangrijk om te vermelden dat de ingewonnen informatie zich berust op de beschikbare commerciële informatie en daarop gebaseerde persoonlijke research.

Desalniettemin lijkt deze uCAN-module interessant voor deze thesis. Er worden namelijk CAN-gegevens verzameld en via een GPRS-verbinding verstuurd naar een centraal punt, in dit geval een centrale server waarop de gebruiker kan inloggen. Bijkomend voordeel is dat deze uCAN-module voorzien is van een accelerometer, zodat de verticale acceleratie van een wagen kan opgemeten worden, wat van belang zal zijn bij het detecteren van putten in het wegdek.

Er is echter een mogelijk nadeel aan het uCAN-systeem. uCAN stuurt periodisch een pakket aan CAN-gegevens door naar de server, met in dit pakket de gegevens vanaf het vorige verzendpunt tot aan het huidige verzendpunt, waardoor de verstuurde data als constant mag beschouwd worden. Voor het Sensovo-project [22] wordt er met deze werkwijze een grote hoeveelheid aan data verstuurd, zeker indien het de bedoeling is om een heel wagenpark uit te rusten met deze technologie. Deze grote hoeveelheid aan data is enerzijds misschien niet nodig daar men er enkel in geïnteresseerd is bij een mogelijke put in het wegdek, anderzijds zijn er hoge kosten verbonden voor de dataoverdracht via een GPRS-verbinding.

Een mogelijke oplossing voor dit probleem zou zijn om enkel de CAN-data te versturen indien er een put in het wegdek gedetecteerd wordt. Het is namelijk aannemelijk dat er zich een afwijkend gedrag voordoet op bepaalde sensorsignalen, wat impliceert dat er eveneens een afwijkend gedrag zich zal voordoen op de respectievelijke CAN-data. De detectie van deze afwijkingen zou in dit geval in het voertuig zelf moeten plaatsvinden, waarna de data doorgestuurd wordt naar de centrale server. Om dit mogelijk te maken moet de embedded software van de uCAN-module, welke de ATOP-chip aanstuurt (‘Figuur 2-12’), aangepast worden waardoor een samenwerking met TASS zich opdringt.



Figuur 2-12: ATOP-chip

Op de server moet de software vervolgens beslissen of dit een “false negative” is, dan wel een effectieve melding van een put. Dit zou kunnen gebeuren door gegevens/meldingen van meerdere voertuigen te combineren.

Deze oplossing vereist echter dat er voldoende opslag- en rekencapaciteit aanwezig is in het voertuig zelf. Het grote nadeel van deze oplossing is dat elk voertuigmodel zijn eigen specifieke software nodig heeft in de uCAN-module, daar elk model zijn eigen specifieke CAN identifiers heeft, waardoor de uCAN-module geen universele module meer is. Daardoor zullen updates voertuigspecifiek moeten toegepast worden.

3 SIMULATIE

Bij het onderzoeken van voertuigsystemen en hun sensoren, werd in paragraaf ‘2.2.6 Conclusie’ reeds aangehaald dat met behulp van simulatie, afwijkingen in de sensorgegevens, gerelateerd aan een put in het wegdek, opgespoord kunnen worden. Zo werd reeds geconcludeerd dat er op zoek moet gegaan worden naar een fusie van sensorgegevens, om zo een algoritme te bekomen welke een put in het wegdek detecteert.

Aanvankelijk werd er gebruik gemaakt van het softwarepakket CarMaker [17], waarbij een voertuig uit de aanwezige bibliotheek een, door de gebruiker te configureren, weg volgde aan een gewenste (variërende) snelheid. CarMaker kan door de gebruiker naar wens aangepast worden, daar dit softwarepakket gebaseerd is op MathWorks Simulink [23]. Echter is de complexiteit van CarMaker dusdanig groot, dat het aanbrengen van de nodige wijzigingen enorm veel tijd in beslag neemt. Bijkomend nadeel aan dit pakket, in het licht van deze thesis, is de beperkte beschikbaarheid van sensoren en de mate waarmee deze de werkelijkheid benaderen.

Daar de Karel de Grote-Hogeschool goede contacten heeft met LMS International, en zij eveneens partner zijn binnen het Sensovo-project [22], werd besloten om gebruik te maken van hun simulatiepakket AMESim [3]. AMESim is een uitgebreid simulatiepakket, welke aangewend kan worden binnen verschillende takken van de mechatronische-wereld waaronder ook automotive. Het maakt gebruik van een gebruiksvriendelijk simulatieplatform, en dankzij de uitgebreide bibliotheken, met vooraf gedefinieerde en gevalideerde componenten, kan de gebruiker zelf de graad van detail bepalen van het simulatiemodel. Het vergt enige tijd om met het simulatiepakket vertrouwd te geraken, maar na het volgen van enkele workshops bij LMS International en het doornemen van de uitgebreide documentatie, kan er op termijn veel tijd uitgespaard worden bij het gebruik ervan.

3.1 Simulatiemodel

Er is voor deze thesis geopteerd om te vertrekken van een bestaand simulatiemodel, welke beschikbaar is in de bibliotheek “Solution Demos”, en deze aan te passen. Hiervoor zijn volgende redenen aan te brengen:

- LMS International beveelt sterk aan om voor deze thesis te vertrekken van een bestaand model. Deze modellen zijn ontworpen en gevalideerd door de eigen ingenieurs, of aangeboden door externe bedrijven welke het gebruiken in de industrie

- De leercurve wordt hierdoor sterk ingekort

- De focus van deze thesis ligt niet op het modelleren van een voertuig, zodat een hoge graad van detail en complexiteit niet wenselijk is. Het vanaf nul modelleren van een voertuig zou bovendien dusdanig veel tijd in beslag nemen, waardoor de focus op het zoeken naar een robuust algoritme ter detectie van putten verloren gaat

Na grondige analyse van de aanwezige simulatiemodellen in de bibliotheek “Vehicle Systems Dynamics, Solution Demos”, werd besloten om als basis het model “Global Analysis” te

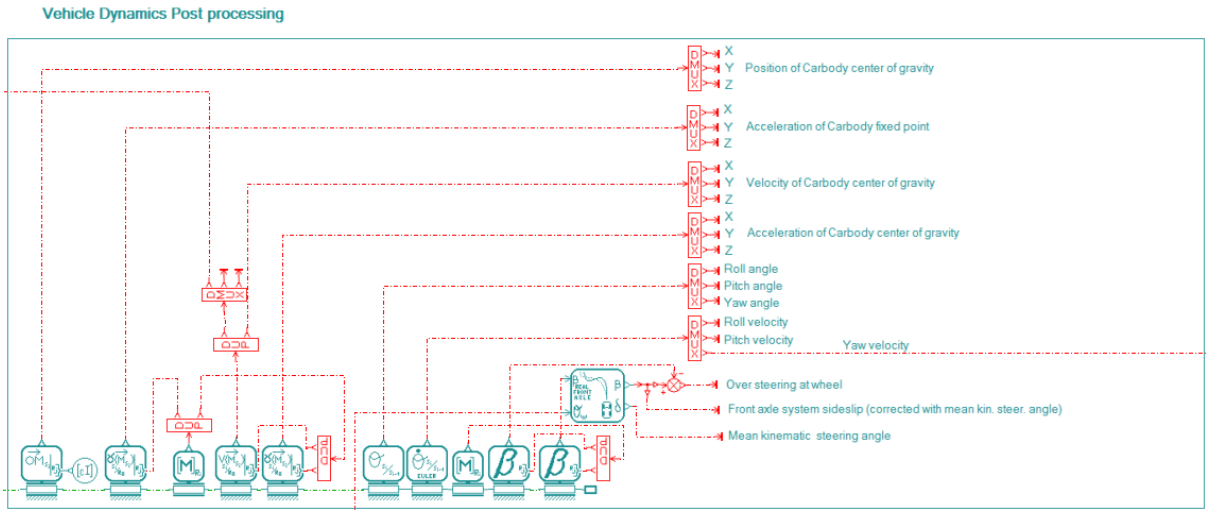
gebruiken. Dit basismodel kan gebruikt worden om het rij- en stuurgedrag van een voertuig te analyseren, waarbij het chassis en de subsystemen van het gemodelleerde voertuig modulair zijn opgebouwd. Door het aanbrengen van enkele wijzigingen aan het basismodel, kunnen de gewenste gegevens gesimuleerd worden en daaruit volgend conclusies getrokken worden.

3.1.1 Bespreking

Een uitgebreide documentatie van het basismodel is terug te vinden in AMEHelp, getiteld "Vehicle Dynamics Global Analysis". Wat volgt is een korte bespreking van de voor deze thesis belangrijkste kenmerken van het simulatiemodel, en van de aangebrachte wijzigingen. Een kopie van het uiteindelijke simulatiemodel is toegevoegd als bijlage 'I Simulatiemodel'.

3.1.1.1 Sensoren

Het grote voordeel van het basismodel zijn de reeds gemodelleerde sensoren, welke voorgesteld zijn in 'Figuur 3-1'.



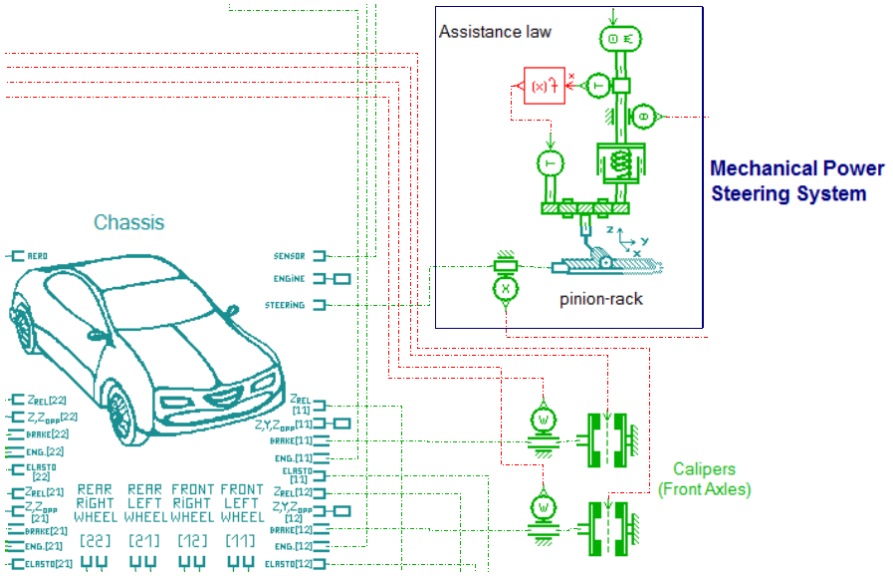
Figuur 3-1: Gemodelleerde sensoren

Met name de sensorgegevens van de accelerometer, geplaatst in het zwaartepunt van het gemodelleerde voertuig, zullen later van belang blijken te zijn voor de detectie van putten in het wegdek. De giersnelheid ("Yaw velocity") wordt teruggekoppeld naar het ABS/ESP-systeem welke verder in deze bespreking aan bod komt. De terugkoppeling van de longitudinale voertuigsnelheid naar een controlenetwerk, zorgt ervoor dat het model de voertuigsnelheid regelt naar de door de gebruiker gemodelleerde snelheid.

De sensoren welke de positie, snelheid, acceleratie en slip opmeten, kunnen naar wens op het gemodelleerde voertuig geplaatst worden. Het parametriseren gebeurt via de globale parameters en/of de parameters van de component. De gegevens van deze sensoren worden uitgedrukt volgens een driedimensionaal cartesisch assenstelsel, waarbij de linkshandige oriëntatie gebruik wordt.

Om de wielsnelheden te kunnen opmeten, zijn er wielsnelheidsensoren aangebracht op elk individueel wielmodel, waar 'Figuur 3-2' een voorbeeld van is. Eveneens zullen deze sensoren van belang zijn bij het modelleren van het ABS. Op dezelfde figuur wordt de

stuurkolom afgebeeld, waar eveneens een extra sensor op aangebracht is om de stuurhoek op te meten. De sensorgegevens zullen gebruikt worden door het gemodelleerde ESP.



Figuur 3-2: Gemodelleerde wielsnelheidsensoren en stuurwielhoeksensor

3.1.1.2 Remsysteem

Het gemodelleerde basismodel bevat geen ABS/ESP-systeem, en zodus ook geen wielsnelheidsensoren. Aangezien elk voertuig van de huidige generatie uitgerust is met een dergelijk systeem, werd het gemodelleerde remsysteem daarmee uitgebreid.

De oorspronkelijke bedoeling was om na te gaan of het ABS/ESP-systeem in werking treedt bij het rijden in een put. Na verloop van tijd bleek dat dit niet het geval is. Bovendien is het reverse engineeren van het ABS- en ESP-signaal op de CAN-bus moeilijk en als onveilig te beschouwen indien dit niet gebeurt op een afgesloten stuk weg.

Het modelleerwerk is daarom niet verloren. De werking van het ABS/ESP-systeem is nagegaan en blijkt te werken. Dit model kan zodus als basis gebruikt worden in een opvolging van deze thesis.

3.1.1.3 Versnellingsbak

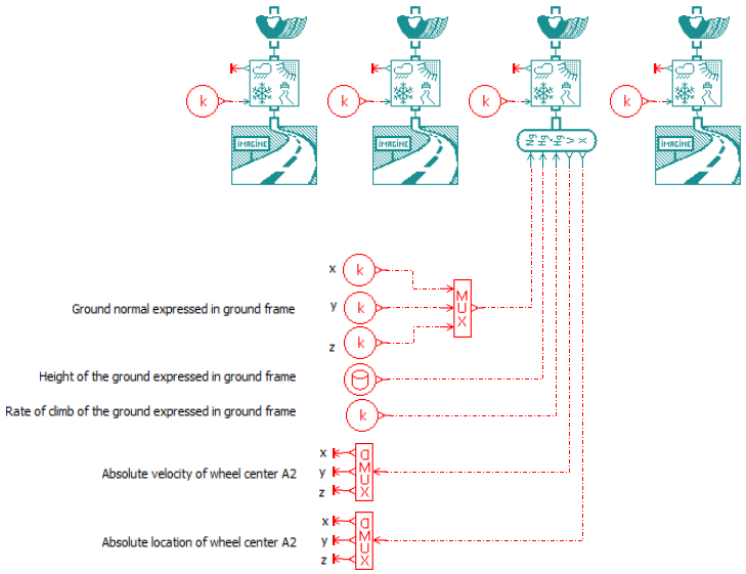
Daar er in het praktisch gedeelte van dit eindwerk zal gereden worden met een voertuig voorzien van een manuele 5-versnellingsbak, is ervoor gekozen om dit gedeelte in detail te modelleren.

De hoge graad van detail maakt het mogelijk om controle te krijgen over de schakelmomenten, welke de realiteit benaderen. Hierdoor kan ook de invloed van het schakelen op het uiteindelijke algoritme geanalyseerd worden, daar het schakelen oorzaak zal zijn van een verticale acceleratie.

3.1.1.4 Modelleren van het wegdekprofiel

De grootste uitdaging was een manier uitzoeken om een put in het wegdek te simuleren. Na een grondige analyse van het basismodel, blijkt dat elk wielmodel zeer gedetailleerd is uitgewerkt. Zo kan bijvoorbeeld per individueel wielmodel de kinematica, plaatsing van het wiel op het wegdek, grip en het te volgen wegdekprofiel ingesteld worden.

Deze laatste te modelleren parameter is in deze thesis van belang. Het basismodel is zo opgebouwd dat het gemodelleerde voertuig rijdt over een vlak wegdekprofiel. Door een wijziging aan te brengen op een wielmodel, zoals voorgesteld in 'Figuur 3-3', kan een put in het wegdek gesimuleerd worden per wielmodel.



Figuur 3-3: Modelleren wegdekprofiel

Voor de komende simulaties is geopteerd om één wiel een dergelijk wegdekprofiel te laten volgen. Met behulp van een tabel kan het exacte tijdstip, de duur, en de diepte van een put in het wegdek ingesteld worden.

3.1.2 Parametriseren van het model

Het parametriseren van het simulatiemodel vergt enig denkwerk, daar het snelheidsprofiel, de instellingen van de versnellingsbak en het moment van een put in het wegdek met elkaar moeten gesynchroniseerd zijn.

Om dit proces te vereenvoudigen in de toekomst is een Excel-bestand aangemaakt, welke terug te vinden is op de CD-ROM onder de bestandsnaam "Parameters_Template", en dit in de folder 'Simulatie'. Een duidelijke gebruikshandleiding is voorzien in het Excel-bestand, en de cellen die niet hoeven gewijzigd te worden zijn beveiligd. Indien gewenst kan deze beveiliging zonder paswoord gedeactiveerd worden. Bij wijze van voorbeeld is de template reeds ingevuld met de gegevens welke gebruikt zijn voor deze thesis.

3.1.2.1 Snelheids- en wegdekprofiel

Er is gekozen om tijdens de simulaties het gemodelleerde voertuig van stilstand te laten versnellen tot 120 km/u. Tijdens deze versnelling worden enkele snelheden gedurende 10 seconden aangehouden. De aangehouden snelheden komen overeen met de meest gebruikelijke snelheidslimieten welke gelden op het Belgische wegennet. Tijdens het aanhouden van deze snelheden zal het rechtervoorwiel door een put rijden met een lengte van 40 cm en een diepte van 4 cm. Dit maakt het mogelijk om in het praktische gedeelte een vergelijking te maken met de simulatie.

In 'Tabel 3-1' wordt de daarbij horende parametrisatie weergegeven voor het snelheids- en het wegdekprofiel. Daar AMESim werkt met de basiseenheden van het SI-stelsel, dienen de snelheden omgezet te worden naar m/s. De lengte van een put dient bij het modelleren van het wegdekprofiel omgezet te worden naar een tijdspanne waarbij deze oneffenheid zich voordoet. Dit komt doordat de ingang, welke het profiel van het wegdek bepaalt, een grafiek is waarbij de hoogte wordt weergegeven in functie van de tijd. Het spreekt voor zich dat de duur van een put afhankelijk is van de snelheid van het voertuig.

Snelheidsprofiel				*Put*		
km/u	*m/s*	*begin (s)*	*einde (s)*	*tijdstip (s)*	*lengte (m)*	*duur (s)*
30	8.33333	10	20	12	0.4	0.0480
50	13.88889	30	40	32	0.4	0.0288
70	19.44444	50	60	52	0.4	0.0206
90	25.00000	70	80	72	0.4	0.0160
100	27.77778	90	100	92	0.4	0.0144
120	33.33333	130	150	132	0.4	0.0120

Tabel 3-1: Paremetrisatie snelheids- en wegdekprofiel

3.1.2.2 Versnellingsbak en koppeling

Door het gekozen snelheidsprofiel in te vullen in de bijgeleverde template, zullen de parameters voor de manuele versnellingsbak en de koppeling berekend worden. Deze berekende waarden dienen vervolgens ingegeven te worden bij hun overeenkomstige ingangen in het simulatiemodel.

3.1.2.3 Globale parameters

In het praktische gedeelte van deze thesis wordt er getest met een Ford Focus, met als bouwjaar 2007. Bedoeling is om het, met behulp van simulatie, gevonden algoritme in de praktijk uit te testen. Daarom dient de simulatie afgestemd te worden op de praktijk.

Als globale parameters kan de gebruiker vooraf gedefinieerde gegevens inladen, overeenkomstig met een geselecteerd klasse voertuig. Voorbeelden van deze vooraf gedefinieerde waarden zijn: massa van het voertuig, wielbasis, spoorbreedte, parameters van de ophanging en de veren, …

Daar een Ford Focus behoort tot een compacte middenklasse, is ervoor gekozen om als globale parameters “Compact_Car.gp” in te laden als template. Vervolgens zijn volgende parameters aangepast, overeenkomstig met de technische gegevens van het testvoertuig:

- Massa van het voertuig: 1240 kg
- Wielbasis: 2640 mm
- Spoorbreedte: 1535 mm vooraan en 1531 mm achteraan

3.1.3 Integratie met Matlab

AMESim biedt de mogelijkheid aan om via MathWorks Matlab [24] een simulatie te starten en alle simulatievariabelen te importeren na afloop. Deze functionaliteit is voor deze thesis bijzonder nuttig. AMESim biedt immers wel de mogelijkheid aan om de simulatievariabelen te plotten, en zelfs meerdere simulatievariabelen te plotten in één en hetzelfde plotvenster. Wiskundige berekeningen met de simulatievariabelen zijn in de AMESim-omgeving echter niet mogelijk. Daar er gezocht wordt naar een algoritme bestaande uit een fusie van sensorgegevens, zullen de wiskundige bewerkingen en het toepassen van het algoritme gebeuren in de Matlab-omgeving.

Om AMESim commando’s te kunnen gebruiken in Matlab, dient in de startfolder van Matlab (welke standaard onder “Mijn Documenten” terug te vinden is), een “startup” m-file aangemaakt te worden met de volgende regel aan code:

```
addpath(fullfile(getenv('AME'),'scripting','matlab','amesim'));
```

Indien in de Matlab-omgeving het commando “help amesim” wordt ingegeven, zou een lijst van de AMESim functies moeten verschijnen.

De code om een simulatie te starten, en na afloop de gewenste variabelen te importeren in de Matlab-omgeving, is toegevoegd als bijlage, getiteld ‘II functie “getAmeResults”’ De code wordt als een functie aan de gebruiker aangeboden, met als mee te geven parameters: de projectnaam van het te simuleren model, de simulatietijd en het samplefrequentie. Deze functie maakt tijdens zijn uitvoering gebruik van de functie “reorganizeAmeResults”, welke opnieuw toegevoegd is als bijlage, getiteld ‘III functie “reorganizeAmeResults”’. Deze laatste functie heeft als enige doel de geïmporteerde simulatievariabelen te converteren in kolomvectoren en vervolgens in één grote datamatrix te plaatsen. Dit maakt dat verdere wiskundige bewerkingen eenvoudiger worden. De datamatrix en de daarbij horende kopteksten worden als parameters naar de Matlab-omgeving teruggekeerd. Een voorbeeld van het aanroepen is:

```
[ data headers ] = getAmeResults( 'simulationModel', 150, 50 );
```

3.2 Bepalen van het algoritme

Eenmaal de parametrisatie en de integratie met Matlab op punt staan, kan er gezocht worden naar een algoritme om putten in het wegdek te detecteren. In wat volgt wordt de werkwijze tot het bekomen van een geschikt algoritme uitgelegd, en zal het uiteindelijke resultaat omschreven worden.

3.2.1 Stappenplan

Het zoeken naar een geschikt algoritme neemt enige tijd en complexiteit met zich mee. Om deze reden is een stappenplan gevolgd, welke meerde keren doorlopen werd, en resulteerde in het uiteindelijke algoritme.

3.2.1.1 Stap 1: selectie van de sensoren

Een eerste stap in het proces naar het vinden van een algoritme is de selectie van de sensoren. Het is hierbij cruciaal om deze selectie zorgvuldig, en doordacht te maken. Het heeft bijvoorbeeld geen nut om sensoren welke de rotatie opmeten van de aandrijfas, mee op te nemen in het algoritme, daar deze geen nut zullen hebben bij de detectie van een put in het wegdek. Een theoretische studie van welke sensoren nuttig zijn voor de toepassing, zoals in paragraaf ‘2.2 Onderzoeken van voertuigsystemen en hun sensoren, is dan ook wenselijk.

3.2.1.2 Stap 2: keuze van de samplefrequentie

Na het selecteren van de sensoren welke nuttige data kunnen leveren voor het te onderzoeken gegeven, in dit geval het detecteren van putten in het wegdek, dient de frequentie waarmee de simulatiegegevens worden gesampled, te worden bepaald. Deze samplefrequentie komt overeen met de frequentie waarmee de gegevens op de CAN-bus beschikbaar zijn. De gekozen samplefrequentie wordt in AMESim het “print interval” genoemd, en wordt uitgedrukt in seconden.

3.2.1.3 Stap 3: selectie van een traject

Vervolgens dient een keuze gemaakt te worden van het, door het gemodelleerde voertuig, te volgen traject. Dit traject dient zorgvuldig gekozen te worden. Een algoritme gevonden op basis van een gesimuleerd traject welke bestaat uit een enkele rechte baan, is een algoritme welke in de realiteit met een lage betrouwbaarheid zal functioneren.

Om dit probleem te voorkomen, is het aangewezen om de hierop volgende stappen parallel uit te voeren voor twee scenario’s, zijnde:

- Het voertuig volgt een rechte baan
- Het voertuig volgt een bocht

In beide scenario’s moet het te testen gegeven, in dit geval putten in het wegdek, op gekende tijdstippen aanwezig zijn. Een voorbeeld hiervan is het gemodelleerde wegdekprofiel in paragraaf ‘3.1.2.1 Snelheids- en wegdekprofiel’, waar op gekende tijdstippen een put in het wegdek gemodelleerd is. Hetzelfde geldt voor het snelheidsprofiel, zodat in de komende stappen eventueel iets kan besloten worden voor bepaalde snelheden. Het is namelijk

mogelijk dat er verschillende algoritmen nodig zijn, afhankelijk van de snelheid van een voertuig.

3.2.1.4 Stap 4: analyse en selectie van de sensorgegevens

Na het selecteren van een geschikt traject, dienen de gegevens van de geselecteerde sensorgegevens uit stap 1 geanalyseerd te worden, en dit per gekozen traject. Er dient gezocht te worden naar verschijnselen in de datagegevens, welke zich enkel voordoen bij het te testen gegeven, in dit geval het rijden in een put. Deze verschijnselen kunnen eventueel nogmaals worden onderverdeeld, afhankelijk van bijvoorbeeld de voertuigsnelheid.

Nadat de sensorgegevens geanalyseerd zijn per traject, dienen deze sensorgegevens geselecteerd te worden welke zichtbaar hetzelfde verschijnsel vertonen bij de beide trajecten.

3.2.1.5 Stap 5: zoeken van een algoritme

Zoals reeds aangehaald is het zoeken naar een geschikt algoritme tijdrovend en complex. Een eenduidige regel tot het bepalen van een algoritme bestaat niet. Uit de ervaring van deze thesis blijkt echter dat volgende richtlijnen, en zelfs een combinatie ervan, een grote kans tot slagen geven:

- Fusie van de geselecteerde sensorgegevens
- Toepassen van drempelwaarden op de geselecteerde sensorgegevens
- Toepassen van wiskundige bewerkingen
- Toepassen van wiskundige filters (bv. Kalman filter)

Zoals reeds aangehaald in stap 4 kan het eventueel voorkomen dat meerdere algoritmen, welke afhankelijk zijn van de voertuigsnelheid, leiden tot de oplossing van een probleem.

3.2.1.6 Stap 6: toepassen van het algoritme

Eenmaal een algoritme in de vorige stap werd verkregen, dient dit algoritme toegepast te worden op de volledige set aan meetgegevens, bekomen uit simulatie. In de praktijk is het onwaarschijnlijk dat men beschikt over een set aan meetgegevens welke een gehele rit omvat. Dit zou immers leiden tot de behoefte aan een grote opslagcapaciteit, daar uit praktische metingen is gebleken dat er per minuut bijna 2 Mb aan data wordt verzameld. Het is daarom aan te raden om te werken met een zogenaamd sliding window, zodat de set aan meetgegevens stapsgewijs wordt geëvalueerd, overeenkomstig met de praktijk.

Dit dient te gebeuren voor beide gemodelleerde trajecten. Daar het wegdekprofiel werd gemodelleerd met putten op gekende tijdstippen, is het algoritme eenvoudig te testen.

Indien op deze gekende tijdstippen een put wordt gedetecteerd, werkt het algoritme correct en kan er overgegaan worden tot het testen in de praktijk. In het andere geval dient dit stappenplan opnieuw te worden doornomen, vertrekkende van stap 2.

3.2.2 Toepassen van het stappenplan

3.2.2.1 Stap 1: selectie van de sensoren

Bij het modelleren is reeds gebleken dat er tal van sensorgegevens beschikbaar zijn in het simulatiemodel. Met het theoretische vooronderzoek in het achterhoofd, is de keuze aan sensorgegevens echter beperkt tot:

- Wielsnelheidssensoren
- Stuurwielhoeksensor
- Acceleratiesensor: longitudinaal, lateraal en verticaal
- Yaw-sensor

Door praktische analyse van de CAN-bus van het testvoertuig is echter gebleken dat de sensorgegevens afkomstig van de stuurwielhoek en de yaw, niet op de bus aanwezig zijn. Dit gegeven beperkt de keuze aan sensorgegevens, en zodus ook de samenstelling van het algoritme, tot volgende selectie:

- Wielsnelheidssensoren
- Acceleratiesensoren: longitudinaal, lateraal en verticaal

Waarbij nog dient opgemerkt te worden dat de gegevens van de accelerometer afkomstig zullen zijn van een smartphone, zoals later in het praktische gedeelte zal blijken.

3.2.2.2 Stap 2: keuze van de samplefrequentie

Uit een eerdere thesis [2] kan er afgeleid worden dat er gemiddeld een frequentie van 40 Hz mag verwacht worden. Analyse van de CAN-gegevens van het testvoertuig heeft aangetoond dat de informatie met betrekking tot de wielsnelheidssensoren, op de CAN-bus geplaatst worden met een frequentie van 75 Hz. Aangezien de simulatieresultaten in de praktijk zullen worden geverifieerd, is voor een samplefrequentie van 75 Hz gekozen.

3.2.2.3 Stap 3: selectie van een traject

Om aan de voorwaarden van het stappenplan te voldoen, is er gekozen voor twee scenario's:

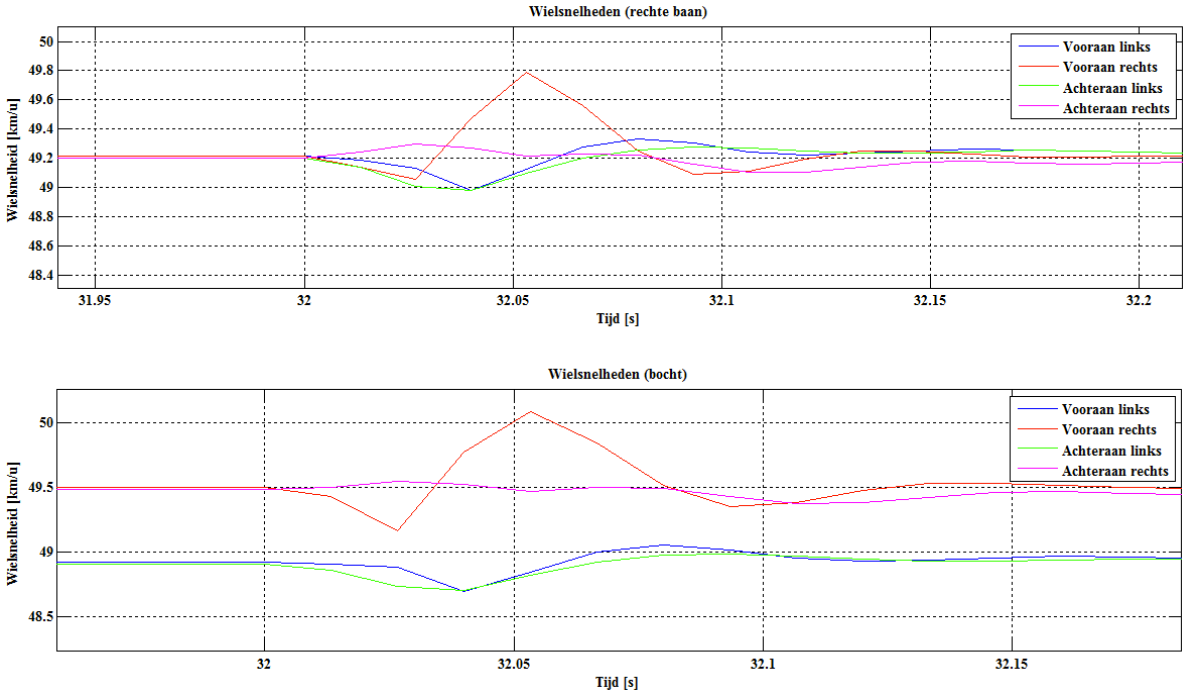
- In een eerste scenario zal het gemodelleerde voertuig een rechte baan volgen, zonder enige bochten
- Een tweede scenario zal het voertuig na 5 seconden, in een cirkel rijden, waarbij de draaihoek ingesteld is op 30° rechtsdraaiend. Op deze manier zal het voertuig als het ware in een bocht rijden op het moment dat er zich een put in het wegdek voordoet

Het snelheids- en wegdekprofiel, zoals gemodelleerd in paragraaf '3.1.2.1 Snelheids- en wegdekprofiel', wordt voor beide scenario's toegepast.

3.2.2.4 Stap 4: analyse en selectie van de sensorgegevens

Voor het analyseren van de wielsnelheden, is het aangewezen om deze te plotten voor beide geselecteerde trajecten. Deze grafieken zijn als bijlage toegevoegd onder ‘IV Wielsnelheden (rechte baan)’ en ‘V Wielsnelheden (bocht)’.

Wat meteen in het oog springt bij deze grafieken, zijn de pieken op de tijdstippen waar een put in het gemodelleerde wegdek zich voordoet. Interessant wordt het wanneer er ingezoomd wordt op een dergelijke piek. Bij wijze van voorbeeld wordt dit gedaan voor de piek op het tijdstip van 32 s, voorgesteld in ‘Figuur 3-4’. Op dit tijdstip werd in paragraaf ‘3.1.2.1 Snelheids- en wegdekprofiel’ het wegdek gemodelleerd met een put. Deze put bevindt zich aan het rechtse voorwiel, wanneer het gemodelleerde voertuig een snelheid heeft van 50 km/u.



Figuur 3-4: Wielsnelheden op het tijdstip van 32 s

Uit de figuur is waar te nemen dat de wielsnelheid van het rechtse voorwiel een relatief grote toename ondergaat, zeker in vergelijking met de overige wielsnelheden, welke interessant is voor het algoritme. Deze toename is te verklaren door het feit dat het desbetreffende wiel in eenzelfde tijd een grotere afstand moet afleggen, ten opzichte van de overige wielen, en dit ten gevolge van de put in het wegdek.

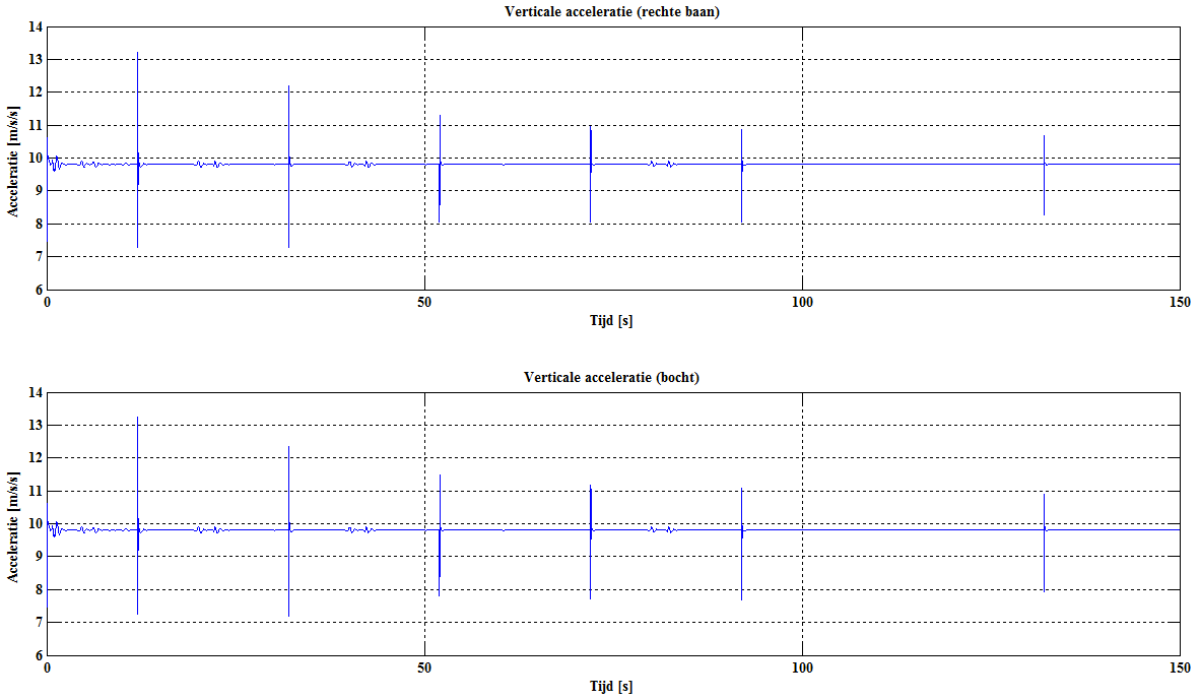
Een vergelijking van de wielsnelheden op de tijdstippen 72 s en 132 s, waarbij zich eenzelfde put in het wegdek voordoet bij hogere snelheden, is toegevoegd als bijlage ‘VI Vergelijking van de wielsnelheden’. Het eerder omschreven verschijnsel doet zich steeds voor, hoewel de piek minder uitgesproken wordt naarmate de snelheid toeneemt.

Andere gegevens welke geanalyseerd dienen te worden, zijn deze van de accelerometer. Uit bijlage ‘VII Gegevens accelerometer’ is hierover te concluderen dat de versnellingen in longitudinale, laterale en verticale richting duidelijke pieken vertonen op de momenten dat er

zich een put in het wegdek voordoet. Echter is waar te nemen dat enkel de verticale acceleratie eenzelfde verloop kent voor beide geselecteerde trajecten. Deze verticale acceleratie is eveneens afgebeeld in ‘Figuur 3-5’.

Het lijkt dan ook opportuun om in het algoritme de verticale acceleratie te betrekken. De redenen hiervoor zijn evident:

- De verticale acceleratie ondervindt geen invloed van het gevolgde traject
- Het is de enige acceleratie welke weinig invloed ondervindt van versnellingen en vertragingen van het voertuig



Figuur 3-5: Verticale acceleratie

Uit deze stap kan zodus worden geconcludeerd dat de volgende sensorgegevens van belang zullen zijn voor het opstellen van een algoritme:

- Wielsnelheden
- Verticale acceleratie

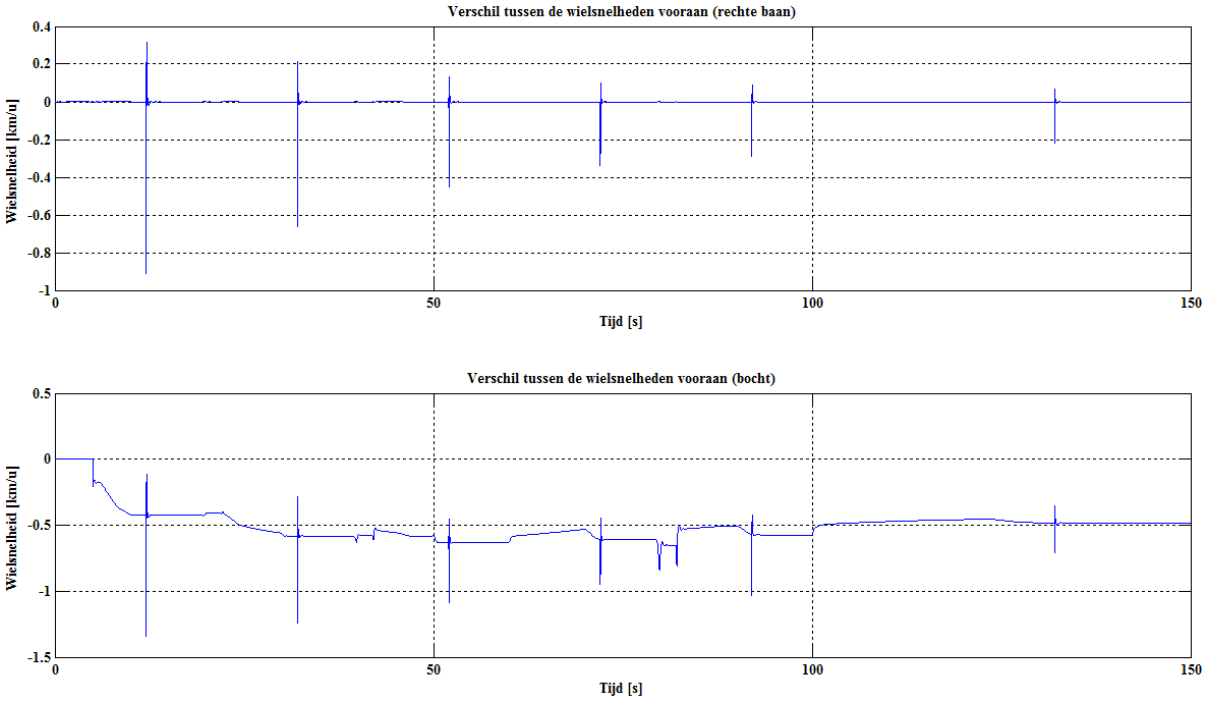
3.2.2.5 Stap 5: zoeken van een algoritme

Vooreerst wordt er aandacht besteed aan de sensorgegevens met betrekking tot de wielsnelheden. In de vorige stap is reeds geconcludeerd dat de wielsnelheid van het wiel welke in een put rijdt, een piek vertoont welke groter is in vergelijking met de overige wielsnelheden.

Uit ‘Figuur 3-4’ kan verder nog besloten worden dat de overige wielsnelheden eveneens beïnvloed worden op het moment dat het rechter voorwiel in een put rijdt. Zo zal het linker

voor- en achterwiel hun wielsnelheid verlagen, terwijl het rechter achterwiel zijn snelheid zal verhogen.

Een eerste conclusie is dan ook om het verschil te nemen tussen de wielsnelheden links en rechts, en dit zowel voor- als achteraan. Dit verschil zal bijgevolg nul zijn bij een vlak wegoppervlak, en een waarde verschillend van nul bij het rijden in put. Door het plaatsen van een statische drempelwaarde zou op deze manier een put kunnen worden gedetecteerd. Zoals 'Figuur 3-6' echter aantoont, waar het verschil tussen de wielsnelheden vooraan is afgebeeld, geldt deze theorie enkel indien het voertuig een rechte baan volgt.



Figuur 3-6: Verschil tussen de wielsnelheden vooraan

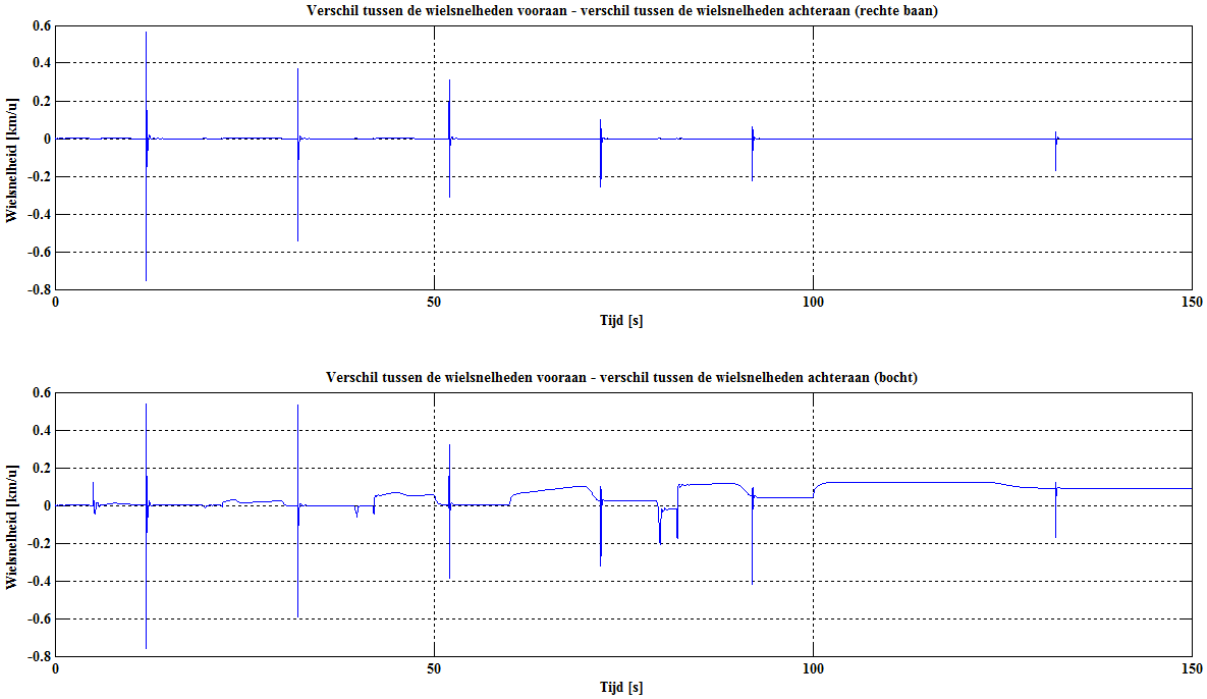
Dit kan als volgt worden verklaard. Indien een voertuig een bocht neemt, veronderstel naar rechts, aan kleine snelheden, dan zal de wielsnelheid vooraan rechts dezelfde zijn als deze achteraan links. Bovendien zal de wielsnelheid vooraan links het hoogst zijn, terwijl deze achteraan rechts het laagst zal zijn. Wanneer dezelfde bocht wordt genomen aan hoge snelheid, dan zullen de wielen aan de binnenzijde sneller draaien dan deze aan de buitenzijde. Deze bevindingen zijn eveneens waarneembaar uit de grafieken in bijlage 'VI Vergelijking van de wielsnelheden' Dit heeft tot gevolg dat het verschil tussen de wielsnelheden links en rechts, nooit nul zal zijn tijdens het nemen van een bocht, zoals te zien is onderaan 'Figuur 3-6'.

Aangezien het verschil tussen de wielsnelheden vooraan en het verschil tussen de wielsnelheden achteraan eenzelfde verloop kent, kan het voorgaande probleem opgelost worden door nogmaals een verschil toe te passen. Ditmaal tussen het resultaat van deze twee bewerkingen. Dit leidt tot 'Figuur 3-7', waar kan worden waargenomen dat voor bochten bij een kleine snelheid het verschil nul geworden is, en voor bochten bij hoge snelheid het verschil geminimaliseerd is. Door de absolute waarde van deze berekening te nemen, zou een statische drempelwaarde gekozen kunnen worden, zodanig dat deze:

- Groter is dan het berekende verschil, welke een kleine waarde heeft bij bochten genomen bij hoge snelheid

- Kleiner is dan de kleinste piek, welke optreedt bij het rijden in een put bij hoge snelheid

Dit leidt tot ernstige beperkingen ter detectie van putten in het wegdek. In dit geval hebben de gemodelleerde putten een diameter van 40 cm, met een diepte van 4 cm. Indien deze dimensies kleiner worden, zullen de pieken eveneens verkleinen, en zodus onder de drempelwaarde komen te liggen in bochten bij hogere snelheden.



Figuur 3-7: Verschil tussen de wielsnelheden vooraan – verschil tussen de wielsnelheden achteraan

Om deze beperking te omzeilen, en zodus de drempelwaarde zo laag mogelijk te houden, is de data van de verticale acceleratie bijzonder nuttig. Deze verticale acceleratie heeft een uitgesproken piek tijdens het rijden in een put, zoals ‘Figuur 3-5’ illustreert, welke gelijk is voor beide trajecten. Bochten hebben met andere woorden geen invloed op het verloop van de verticale acceleratie. Zodus, kan aan deze set van gegevens een drempelwaarde toegekend worden, welke net hoog genoeg is om de kleine verticale acceleratie-pieken tijdens het versnellen of vertragen weg te filteren.

Er is geopteerd om deze kleine verticale acceleratie-pieken weg te filteren met een zogenaamde “small peak filter”, welke is toegevoegd als bijlage ‘VIII Functie “smallPeakFilter”’. Deze functie verwacht als argumenten een set aan datagegevens, een gemiddelde en een drempelwaarde. Na het filteren zal de functie de gefilterde set aan datagegevens teruggegeven. In dit geval is de set aan datagegevens de verticale acceleratie, waarbij het gemiddelde de zwaartekracht van 9.80665 m/s^2 bedraagt. De drempelwaarde wordt ingesteld op 3,5 % van het gemiddelde.

Het uiteindelijke algoritme zal deze gefilterde verticale acceleratie als een parameter beschouwen, ter detectie van een put in het wegdek. Een andere parameter is het verschil uit ‘Figuur 3-7’. Hierdoor wordt het algoritme als volgt omschreven:

(abs(diffFrontRear) >= 0.13) && (zAcceleration(j) ~= 9.80665)

Door deze combinatie kan de drempelwaarde van deze laatste parameter laag worden genomen, waardoor het mogelijk wordt om putten te detecteren met kleinere dimensies. Bovendien maakt de eenvoud van dit algoritme het mogelijk om deze eventueel embedded te programmeren in het voertuig zelf, zonder de noodzaak aan een rekenkundig sterke processor.

3.2.2.6 Stap 6: toepassen van het algoritme

Het toepassen van het algoritme op de volledige set aan simulatiegegevens, wordt uitgevoerd met het toepassen van een sliding window, om zo de praktische uitvoering te benaderen. Om dit proces te vereenvoudigen is hiervoor een eenvoudige functie geschreven welke toegevoegd is als bijlage ‘IX Functie “findPitSW”’.

Deze functie dient als parameters de set aan simulatiegegevens, een drempelwaarde voor de wielsnelheden en een drempelwaarde voor de verticale acceleratie mee te krijgen. Uit analyse is gebleken dat de drempelwaarde voor de wielsnelheden op 0.13 km/u dient ingesteld te worden. De drempelwaarde voor de verticale acceleratie wordt uitgedrukt als een percentage ten opzichte van een gemiddelde waarde, in dit geval de zwaartekracht. Een percentage van 3.5 % blijkt hiervoor een optimale waarde te zijn. Hierdoor kan de eerder aangehaalde functie als volgt worden aangeroepen, waarbij een window wordt toegepast van 100 simulatiegegevens:

[pits headers] = findPitSW(data, 100, 0.13, 0.035);

In onderstaande tabellen is het resultaat te zien voor de uitvoering van het algoritme, met behulp van het sliding window. In ‘Tabel 3-2’ zijn de resultaten indien het traject bestaande uit een rechte baan wordt gevolgd. ‘Tabel 3-3’ toont deze in het geval het traject wordt gevolgd bestaande uit een bocht. De tabellen geven gemiddelde waarden weer, met uitzondering van de laatste gedetecteerde put waarbij de detectie bestaat uit één enkele waarde.

Tijd [s]	*Snelheid [km/u]*	*Verschil [m/s]*	*z-acceleratie [m/s/s]*
12.0533	29.0621	0.1474	9.1932
32.0399	48.8700	-0.1998	10.8360
52.0399	68.7044	-0.3024	11.1831
72.0465	88.2141	-0.2175	10.8701
92.0398	98.4678	-0.1694	10.6209
132.0397	117.8809	-0.1713	10.6826

Tabel 3-2: Gedetecteerde putten bij het volgen van een rechte baan

Tijd [s]	*Snelheid [km/u]*	*Verschil [m/s]*	*z-acceleratie [m/s/s]*
12.0533	29.0610	0.1523	9.1936
32.0466	48.8587	-0.2626	11.1661
52.0399	68.6733	-0.3681	11.3816
72.0398	88.1202	-0.2251	10.9548
92.0398	98.2851	-0.2601	10.7869
132.0397	117.4833	-0.1724	10.8852

Tabel 3-3: Gedetecteerde putten bij het volgen van een bocht

Hoewel de keuze van het window geen invloed heeft ter detectie van de gemodelleerde putten, zal de keuze van de samplefrequentie wel een grote invloed hebben. Het verlagen van de samplefrequentie zal resulteren in een detectie van de putten met een mindere nauwkeurigheid. Uiteindelijk zal bij het verder verlagen van de frequentie, de laatste gemodelleerde put waar het voertuig een snelheid heeft van 120 km/u, niet meer gedetecteerd worden.

De minimale samplefrequentie kan afgeleid worden uit de grafieken afgebeeld in bijlage ‘VI Vergelijking van de wielsnelheden’. Daaruit kan vastgesteld worden dat bij hoge snelheden de afwijkingen slechts gedurende 40 ms voldoende groot zijn, ter detectie van de gemodelleerde putten in het wegdek. Dit komt overeen met een samplefrequentie van 25 Hz. Voor het detecteren van putten bij lagere snelheden, bijvoorbeeld bij 50 km/u, mag de samplefrequentie verlaagd worden tot 18 Hz. Dit doordat de afwijkingen gedurende een langere tijd voldoende groot zijn, namelijk gedurende 55 ms.

3.3 Conclusie

Ter detectie van putten in het wegdek, werd een simulatiemodel gemodelleerd waarbij een voertuig vooraf gemodelleerde trajecten volgt. De gemodelleerde trajecten zijn de twee situaties waarvoor de karakteristieken in de simulatiegegevens behoorlijk verschillen, namelijk bij een rechte baan en bij het nemen van een bocht.

Na analyse van de sensorgegevens, is geopteerd voor een eenvoudig algoritme, welke geldig is in beide situaties. De keuze voor een dergelijk algoritme ligt voor de hand. Het gebrek aan complexiteit maakt het mogelijk om het algoritme eventueel embedded te programmeren, zonder de noodzaak aan rekenkundige processorkracht. Dit is meteen een gedeeltelijk antwoord op open punten 5 en 6, aangehaald bij de doelstellingen op pagina 6.

Het toepassen van dit algoritme op de set van simulatiegegevens leverde gunstige resultaten. Indien een samplefrequentie gelijk aan of groter dan 25 Hz werd gekozen, is aangetoond dat het algoritme de gemodelleerde putten kon detecteren, en dit bij snelheden tot 120 km/u. Hierbij werd gebruik gemaakt van een sliding window over de set aan simulatiegegevens, vergelijkbaar met de komende praktische metingen waar de meetgegevens eveneens stapsgewijs worden verkregen. Deze manier van werken is eveneens een antwoord op open punt 6, waarbij de noodzaak aan opslagcapaciteit in een module zoals de uCAN, beperkt kan gehouden worden.

De dimensies van de gemodelleerde putten in het wegdekoppervlak zijn zo gekozen, dat door middel van een eerste praktische meting met een testvoertuig, het opgestelde algoritme kan geverifieerd worden. Daar de doelstelling, gedeeltelijk opgelegd door het VIM, vereisen dat een put met een diameter van 20 cm en een diepte van 4 cm gedetecteerd moet kunnen worden, werd als afsluitend onderdeel het wegdekprofiel gemodelleerd met dergelijke putten. Hieruit werd vastgesteld dat het algoritme eveneens in staat is om deze putten te detecteren, en dit voor beide trajecten. Daaruit mag voorzichtig geconcludeerd worden dat, indien de verificatie met het voertuig positief is in het geval van een put met een diameter van 40 cm en een diepte van 4 cm, dit eveneens het geval zal zijn voor de door het VIM opgelegde dimensies.

4 PRAKTISCHE UITVOERING

Door het gebruik van simulatie werd in het vorige hoofdstuk een algoritme gevonden, welke het mogelijk moet maken om putten in het wegdek te detecteren. Het doel van dit praktisch georiënteerde deel is om in de eerste plaats de conclusies uit de simulatie te bevestigen. Meer bepaald de ondergrens van de samplefrequentie, en de werking van het gevonden algoritme dienen bevestigd te worden.

Eenmaal de simulatie bevestigd is, en zodus het algoritme in de praktijk functioneert, zullen meerdere testritten op de openbare weg plaatsvinden. Hiervoor zal een vast traject gevolgd worden, waarvan de putten in het wegdek geïnventariseerd zijn. Tijdens de testrit worden de gegevens gelogd, om deze later te analyseren in Matlab. Dit biedt de mogelijkheid tot verdere optimalisatie van het algoritme, en het onderzoeken van alternatieve algoritmen.

4.1 Meetopstelling

Dit praktisch gedeelte wordt uitgevoerd met een Ford Focus, met als bouwjaar 2007. De enige uitzondering op deze regel is de test waarbij de minimale samplefrequentie wordt geverifieerd.

Aangezien de uCAN-module niet ter beschikking kon gesteld worden tijdens de praktische testen, diende een alternatieve oplossing gevonden te worden om van een doordeweeks voertuig, de gegevens op de CAN-bus te loggen. Aangezien het testvoertuig een vrij recent model is, beschikt het over een OBD-II connector. Zoals reeds aangehaald in paragraaf ‘2.1 Onderzoeken van CAN’, is deze connector verbonden met de in het voertuig aanwezige CAN-bus. Na onderzoek is gebleken dat het bij dit testvoertuig mogelijk is om de gegevens op de CAN-bus te monitoren, door een interface aan te sluiten op de OBD-II connector, zoals voorgesteld in ‘Figuur 4-1’.

Figuur 4-1: Kvaser Leaf SemiPro LS

In het licht van deze thesis is het echter wenselijk om de CAN-bus te loggen gedurende een testrit, en deze vervolgens te exporteren naar een CSV-bestand. Dit maakt het mogelijk om na afloop van de testrit, de gegevens te analyseren in Matlab en (alternatieve) algoritmen uit te testen. Om dit mogelijk te maken heeft dr. Wouter Hendrickx de API van de interface aangewend, om in de programmeertaal C# een softwaretool te ontwikkelen die dit mogelijk maakt. In een laatste fase kan het algoritme in deze softwaretool geïmplementeerd worden, om zodoende tijdens de rit, real-time putten in het wegdek te detecteren.

Daar de CAN-identifiers van het testvoertuig echter niet bekend zijn, dient eerst reverse engineering toegepast te worden om de nodige identifiers op te sporen. Hiervoor kan een eenvoudig proces gevolgd worden, welke als volgt kan worden omschreven:

Bij het lokaliseren van een parameter op de CAN-bus, dienen er zo veel mogelijk andere parameters uitgesloten te worden, zodat over de CAN-bus een minimum aan gegevens verstuurd wordt. Indien dit niet mogelijk is, dient een testscenario gekozen te worden waarbij het resultaat van de te lokaliseren parameter gekend is.

Tijdens dit proces worden de CAN-gegevens gelogd, waarna deze kunnen geanalyseerd worden. Het resultaat zou de gezochte parameter moeten opleveren.

Voor het opsporen van de wielsnelheden bijvoorbeeld, kan dit gedaan worden door alle gegevens van de CAN-bus te loggen tijdens een rit. Gedurende de rit wordt op zo veel mogelijk tijdstippen bijgehouden wat de voertuigsnelheid is. Door alle databytes per identifier te plotten in grafieken, of in het geval van dit testvoertuig te plotten per paar databytes, kan aan de hand van de grafieken bepaald worden onder welke identifier de wielsnelheden op de CAN-bus geplaatst worden. Een herschaling, en het toepassen van een offset, maakt het mogelijk om de wielsnelheden uit te drukken in km/u.

De ontbrekende factor zijn de gegevens van de accelerometer, waarvan de verticale acceleratie van belang bleek te zijn voor een werkend algoritme. Het gebrek aan deze gegevens dient gezocht te worden in het ontbreken van ESP op het testvoertuig, zodat gegevens afkomstig van de accelerometer, niet aanwezig kunnen zijn op de CAN-bus. Bovendien wil de aanwezigheid van een ESP niet impliceren dat de gegevens van de accelerometer aanwezig zijn op de CAN-bus, zoals reeds geconcludeerd werd in paragraaf ‘2.2 Onderzoeken van voertuigsystemen en hun bijhorende sensoren’. Om deze reden is ervoor gekozen om een smartphone in de meetopstelling te integreren. De gegevens van de ingebouwde driedimensionale accelerometer worden opgevraagd en eveneens gelogd in de, door dr. Wouter Hendrickx, ontwikkelde softwaretool.

4.2 Verificatie van het simulatiemodel

4.2.1 Samplefrequentie

Tijdens de simulaties werd er vastgesteld dat de samplefrequentie niet lager mag zijn dan 25 Hz. Indien dit wel het geval zou zijn, is de kans reëel dat (sommige) putten in het wegdek niet zullen gedetecteerd worden.

Om deze theorie in de praktijk te testen, dient een testrit uitgevoerd te worden met een voertuig waarvan de CAN-berichten, en in het bijzonder deze met betrekking tot de wielsnelheden, met een lage frequentie op de bus worden geplaatst. Een Volvo V70, met als bouwjaar 2001, is hiervoor bijzonder geschikt. Daar dit voertuig tot het wagenpark behoort

van de opleiding “Autotechnologie” op Campus Hoboken, dienen de CAN-identifiers niet reverse-engineered te worden.

Tijdens een testrit worden de volgende identifiers gelogd, zodat verdere analyse mogelijk is:

- 0x00300036, DB 6 en DB 7: voertuigsnelheid
- 0x01400002, DB 4 t.e.m. DB 7: wielsnelheden vooraan rechts, vooraan links, achteraan rechts en achteraan links

Het gevolgde traject van deze testrit kan als volgt worden beschreven:

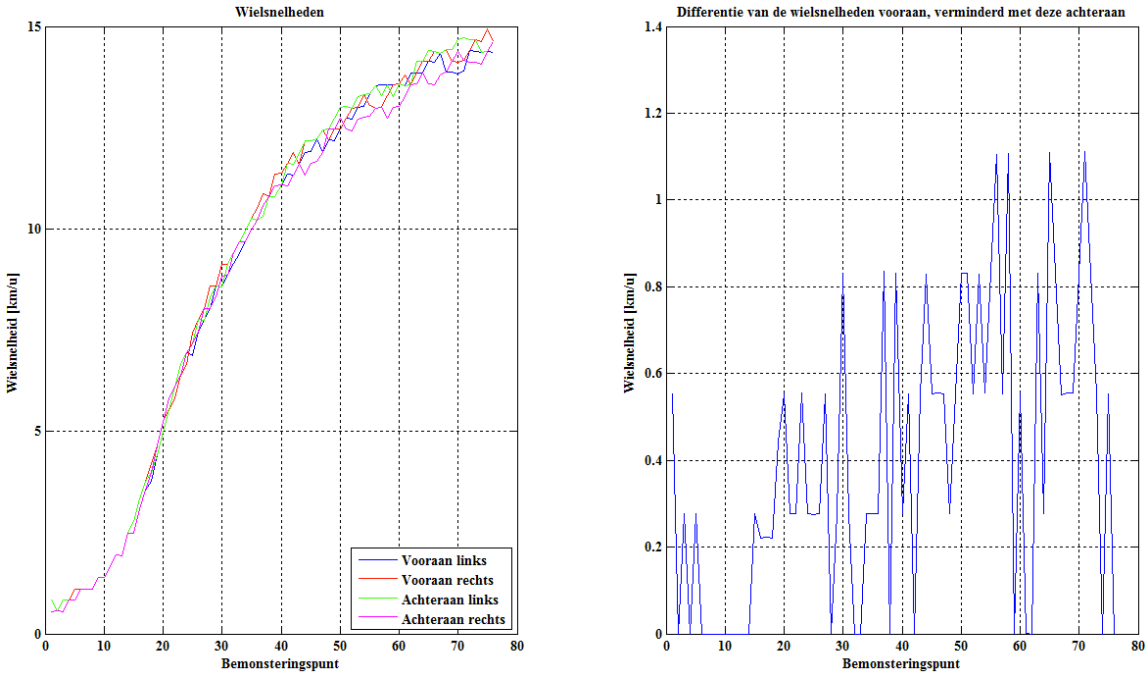
Het voertuig volgt een kort stuk weg met een lengte van circa 30 m. Het rechter voorwiel rijdt hierbij in een put, met een diameter van 40 cm en een diepte van 4 cm, en dit bij een snelheid van 15 km/u. De meting wordt gestopt kort na het rijden in de put.

Het resultaat van de meting is toegevoegd als bijlage ‘X Meting Volvo V70’. Deze bijlage geeft de wielsnelheden weer, welke onder identifier 0x01400002 op de CAN-bus worden geplaatst. De hexadecimale waarden zijn hierbij geconverteerd naar decimale waarden om de leesbaarheid en bewerkingen te bevorderen.

Uit het resultaat van de meting kan geconcludeerd worden dat de wielsnelheden als een (oplopende) teller, met een waarde tussen 0 en 255, op de CAN-bus worden geplaatst. In paragraaf ‘2.2.1 Anti-Lock Braking System’ werd toegelicht dat een wielsnelheid opgemeten wordt door een wielsnelheidssensor te plaatsen aan een tandenkrans, welke gemonteerd is op het draaiende wiel. De waarden van de oplopende teller in de desbetreffende tabel, zijn dan ook een rechtstreeks gevolg van deze implementatie. Per tijdsinterval, wordt de huidige waarde van de sensor op de bus geplaatst.

Om de wielsnelheden grafisch uit te zetten, dient het verschil tussen twee waarden berekend te worden, waarna de afgeleide naar de tijd dient genomen te worden. Dit omwille van het feit dat het tijdsinterval niet constant is. De linkse grafiek in ‘Figuur 4-2’ is hiervan het resultaat, indien een sliding window wordt gehanteerd van 5 waarden.

De rechtse grafiek in ‘Figuur 4-2’, is het verschil van de wielsnelheden vooraan en van deze achteraan, waarna nogmaals het verschil van beide bewerkingen wordt genomen. Dit in overeenstemming met de simulaties, waar deze berekening een parameter was van het uiteindelijke algoritme. Indien de drempelwaarde, welke was vastgelegd op 0.13 km/u, buiten beschouwing wordt gelaten, dan is het bij deze meting nog onmogelijk om de put in het wegdek te lokaliseren aan de hand van deze gegevens. Dit doordat de waarde van de teller weinig nauwkeurig is, en het tijdsinterval waarmee deze op de CAN-bus wordt geplaatst veel te groot is. Dit tijdsinterval is gemiddeld genomen 115 ms, wat overeenkomt met 8.7 Hz. Veel te laag zo blijkt uit ‘Figuur 4-2’. Hiermee is aangetoond dat er wel degelijk een ondergrens bestaat voor de samplefrequentie, welke het mogelijk maakt om putten in het wegdek te detecteren.



Figuur 4-2: Grafische uitzetting van de meting met een Volvo V50 (2001)

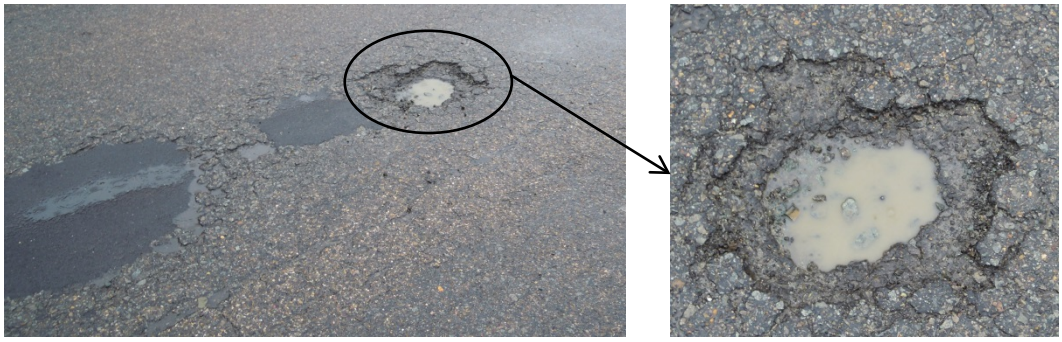
Een definitieve bevestiging wordt verkregen in het volgende onderdeel van de verificatie, wanneer het verkregen algoritme in de praktijk wordt toegepast. Daar zal blijken dat, wanneer de wielsnelheden met een voldoende hoge frequentie op de CAN-bus worden geplaatst, conclusies kunnen getrokken worden betreffende putten in het wegdek

De gegevens van de accelerometer, afkomstig van de smartphone, zijn in deze meting nog niet geïntegreerd. Verder in deze uiteenzetting zal vastgesteld kunnen worden dat de frequentie waarmee deze gegevens beschikbaar zijn, gemiddeld 25 Hz bedraagt. Dit is precies de ondergrens welke uit simulatie werd bepaald, en zal net voldoende blijken te zijn.

4.2.2 Algoritme

Om het algoritme op zijn praktische werking na te gaan, dient een testscenario gekozen te worden welke nauw aanleunt bij deze van de simulatie. Dit wil zeggen dat in het ideale scenario het testvoertuig aan een snelheid van bijvoorbeeld 50 km/u, in een put met een diameter van 40 cm en een diepte van 4 cm zou rijden. Daar de testritten gebeuren op de openbare weg, is dit niet zo evident te noemen.

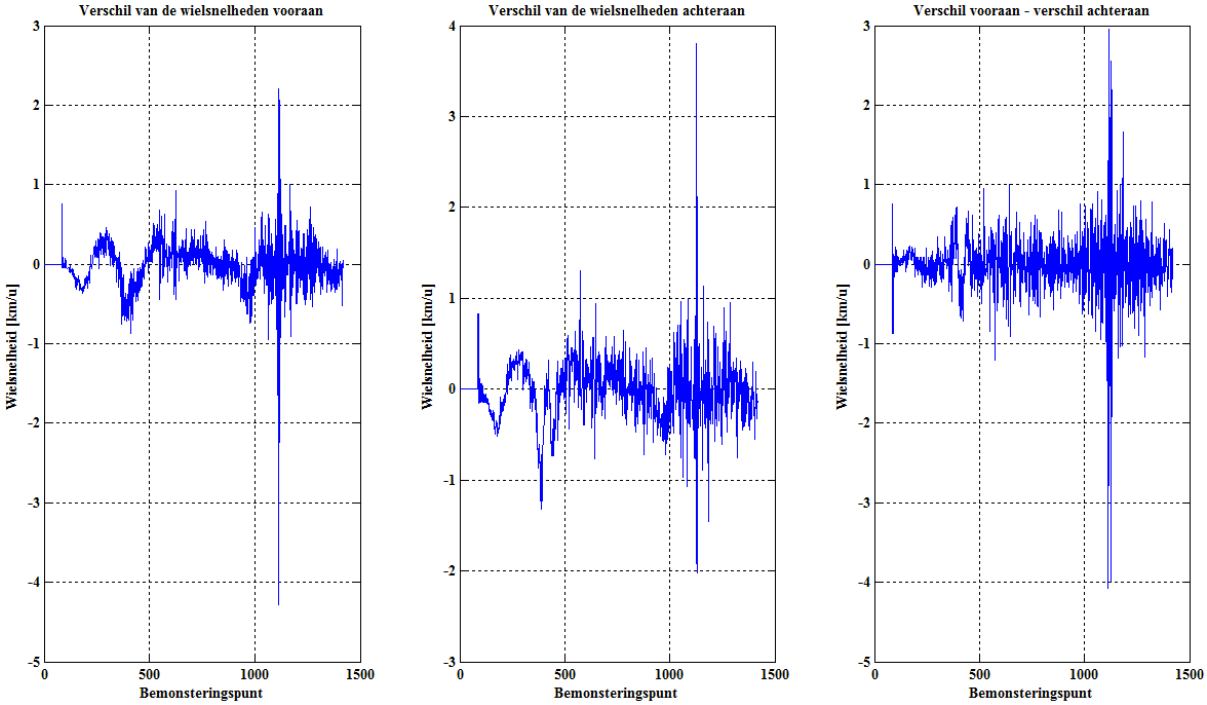
Desalniettemin is er toch een deel openbare weg gevonden welke voldoet aan de criteria van de simulatie. Het weggedeelte in kwestie, de Oudebaan te Wilrijk, bevat een rechte baan bestaande uit asfalt, met in elke rijrichting enkele geïsoleerde putten. Een van deze putten, welke voorgesteld is in ‘Figuur 4-3’, heeft een diameter van 40 cm en een snel oplopende diepte tot 4 cm. De op het dashboard afgelezen snelheid waarmee het voertuig in de put rijdt, en dit met het rechtse voor- en achterwiel, bedraagt ongeveer 45 km/u, wat in de buurt komt van de gewenste 50 km/u. Dit is eveneens af te leiden uit de grafieken in bijlage ‘XI Wielsnelheden Ford Focus (2007)’, welke de wielsnelheden voorstellen tijdens deze specifieke meting. Daaruit blijkt dat de wielsnelheid op dat moment ongeveer 43 km/u bedraagt.



Figuur 4-3: Put in het wegdek

4.2.2.1 Verificatie van de wielsnelheden

Uit de grafieken in de bijlage is af te leiden dat rond het bemonsteringspunt 1100, het voertuig in de bewuste put rijdt. Dit resulteert, zoals verwacht uit de simulatie, in een plotse piekwaarde bij de wielsnelheden van het rechtse voor- en achterwiel. Na het uitvoeren van de noodzakelijke wiskundige bewerkingen, worden de grafieken verkregen zoals voorgesteld in 'Figuur 4-4'. Hierbij is de rechtse grafiek van belang, daar deze een eerste parameter is van het algoritme ter detectie van putten in het wegdek.



Figuur 4-4: Wiskundige bewerkingen met de wielsnelheden

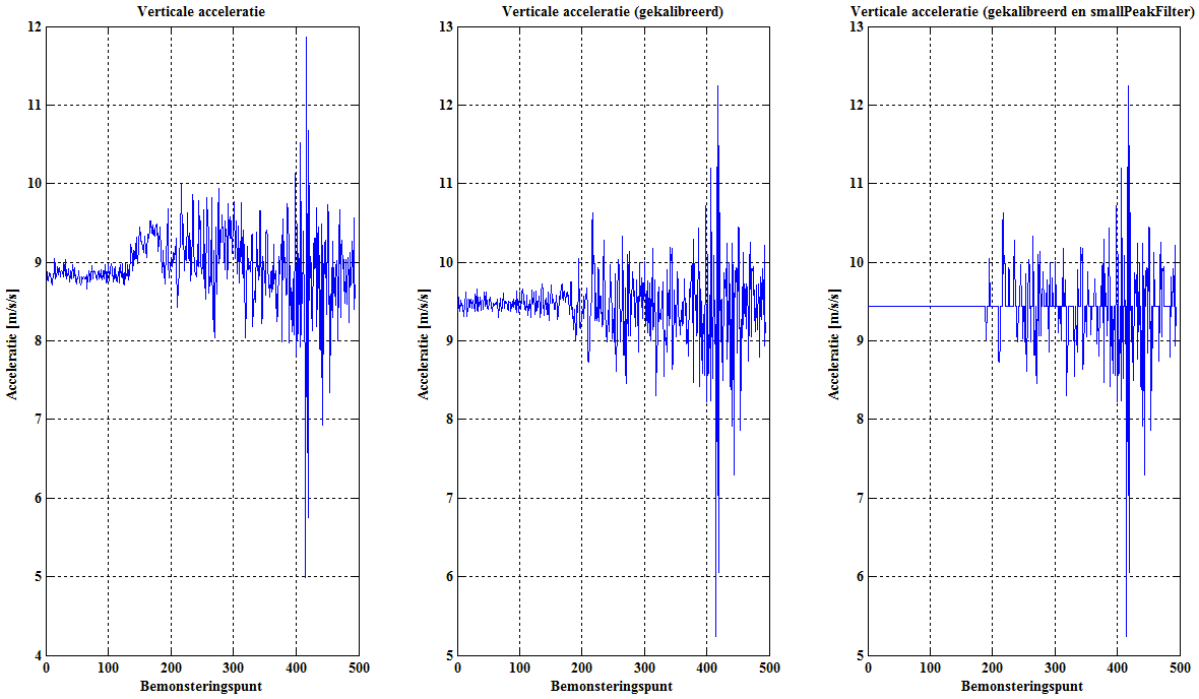
Hoewel het resultaat gunstig is, daar een duidelijke piek aanduidt dat er zich op dat moment een put in het wegdek voordoet, zijn er toch enkele opmerkelijke verschillen in vergelijking met het gesimuleerde resultaat van 'Figuur 3-7'. Het meest in het oog springende verschil is de ruis welke aanwezig is. Door de hoge frequentie waarmee de gegevens op de CAN-bus worden geplaatst, namelijk 75 Hz, is deze ruis het gevolg van oneffenheden in het

wegoppervlak, zoals te zien is in ‘Figuur 4-3’. Dit impliceert dat het ruisniveau afhankelijk zal zijn van de wegverharding (asfalt, beton, …), en de algemene staat van het wegdek (verzakkingen, herstellingen, …). Een tweede verschil is de amplitude van de piekwaarde. Deze blijkt een factor 10 keer groter te zijn in vergelijking met deze uit de simulaties.

Het gevolg is dan ook dat de, uit simulatie vastgelegde, drempelwaarde van 0.13 km/u dient herzien te worden. Daar de piekwaarde een factor 10 keer groter blijkt te zijn, wordt geopteerd om de drempelwaarde vast te leggen op een waarde welke eveneens 10 keer groter is, wat resulteert in 1.3 km/u. Bijgevolg zal de ruis ten gevolge van de wegverharding en/of de staat van het wegdek, niet gedetecteerd worden als een mogelijke put in het wegdek.

4.2.2.2 Verificatie van de verticale acceleratie

Zoals eerder al is aangegeven zijn de gegevens van de driedimensionale accelerometer afkomstig van een smartphone, en dit aan een samplefrequentie van 25 Hz. In ‘Figuur 4-5’ stelt de linkse grafiek de ruwe data voor, welke afkomstig is van deze accelerometer.



Figuur 4-5: Verticale acceleratie

De smartphone kan echter in eender welke positie in het voertuig aanwezig zijn, zodat de verticale as van de smartphone niet loodrecht op het voertuig staat. Daar dit een vereiste is om putten in het wegdek te detecteren, zodat de verticale acceleratie van het voertuig wordt gemeten, dienen de gegevens afkomstig van de accelerometer gekalibreerd te worden [25], [26]. Deze kalibratie kan gebeuren door gebruik te maken van de hoeken van Euler. Hierbij wordt gebruik gemaakt van twee vrijheidsgraden, namelijk de hoek rond de x-as (roll) en deze rond de y-as (pitch). De wiskundige uitwerking voor deze kalibratie is beschreven in bijlage ‘XII functie “accCalibration”‘.

De middelste grafiek in ‘Figuur 4-5’ is het resultaat van deze kalibratie, waarbij de verticale acceleratie in rust ongeveer 9.45 m/s^2 bedraagt, wat dicht aanleunt bij de gravitatiekracht van

9.80665 m/s^2. Meerdere metingen hebben uitgewezen dat deze kalibratie zeer betrouwbaar functioneert. Het is dus niet noodzakelijk om de smartphone zodanig te plaatsen in het voertuig, dat de verticale as loodrecht staat op het voertuig.

Verder kan in de middelste grafiek een witte ruis worden opgemerkt. Deze is afkomstig van het draaien van de motor welke trillingen in de wagen teweeg brengt, het veranderen van versnelling, … Met de eerder besproken functie in bijlage 'VIII Functie "smallPeakFilter"', is het mogelijk om deze ruis weg te filteren. Daarvoor wordt bij het begin van de meting, een gemiddelde genomen van de verticale acceleratie. De tijd waarover dit gemiddelde wordt genomen, is de tijd dat het voertuig stilstaat. Mits het toepassen van een bijkomende tolerantie van 3.5 % op dit gemiddelde, in overeenstemming met de simulaties, wordt de rechtse grafiek bekomen waarbij de ruis weg gefilterd is.

Na deze bewerkingen wordt een resultaat bekomen welke dient vergeleken te worden met de verticale acceleratie bekomen uit simulatie, voorgesteld in 'Figuur 3-5'. Door interpolatie toe te passen in 'Figuur 3-5', kan vastgesteld worden dat de verticale acceleratie bij een snelheid van 45 km/u, 12.45 m/s^2 bedraagt als bovengrens en 7.30 m/s^2 als ondergrens. Vergelijken we dit met de opgemeten verticale acceleratie van 'Figuur 4-5', dan mag besloten worden dat de bovengrens alvast een gelijkaardige waarde oplevert. De ondergrens ligt weliswaar 2 m/s^2 lager dan deze bekomen uit simulatie. Dit is echter geen probleem, daar dit een grotere (negatieve) piekwaarde oplevert dan verwacht uit simulatie, zodat de tolerantie van 3.5 % voldoende laag gekozen is. Het is zelfs zo dat deze hoger mag gekozen worden ter detectie van putten met een diameter van 40 cm en een diepte van 4 cm. Daar het ultieme doel van het VIM is om zoveel mogelijk defecten in het wegdek te detecteren, met kleinere dimensies, is ervoor gekozen om de vooropgestelde tolerantie te behouden.

4.2.2.3 Verificatie van het detecteren van putten

Na deze analyse en de bijkomende praktische aanpassingen, kan het algoritme in de praktijk worden getest. Hiervoor is een nieuwe functie geschreven, welke als bijlage 'XIII Functie "findAnomalies"' is toegevoegd. Als argumenten dienen een set aan CAN-datagegevens, een struct met de gekende CAN-identifiers, een venstergrootte, een drempelwaarde voor de wielsnelheden en een toegestane tolerantie voor de verticale acceleratie meegegeven te worden. De drempelwaarde, voor de parameter met betrekking tot de wielsnelheden, en de toegestane tolerantie, voor de parameter met betrekking tot de verticale acceleratie, zijn reeds gekend. De eerder besproken bewerkingen zijn immers mee opgenomen in deze functie. De set aan ruwe CAN-datagegevens wordt verkregen door alle beschikbare CAN-data te loggen naar een CSV-bestand. Dit met behulp van de speciaal hiervoor ontworpen software. Door het deels reverse engineeren van het testvoertuig, zijn de voor deze thesis belangrijkste CAN-identifiers gekend, en geformuleerd in een struct. Uit de set van ruwe CAN-datagegevens, is het daardoor mogelijk om deze gekende berichten te filteren. Het toepassen van een venstergrootte kan op zijn beurt het met mondjesmaat verkrijgen van CAN-data simuleren. Het algoritme zal immers geïmplementeerd worden in de ontworpen software. Dit met de bedoeling om, in real-time, putten in het wegdek te detecteren. Het resultaat van deze functie, de gedetecteerde putten, worden teruggegeven naar de oproepende functie.

Het resultaat van het uitvoeren van deze functie voor deze specifieke testrit, waarbij in de bewuste put van 'Figuur 4-3' wordt gereden, is te zien in 'Tabel 4-1'.

Wielsnelheid [m/s]				
Vooraan links	*Vooraan rechts*	*Achteraan links*	*Achteraan rechts*	*Verticale acceleratie [m/s/s]*
42.897	43.133	42.454	42.719	11.465

Tabel 4-1: Resultaat van de verificatie van het algoritme

Er dient echter vermeld te worden dat de GPS-data bij deze testrit niet beschikbaar is, ten gevolge van een softwareprobleem. Eveneens is de tijdstempel van het moment waarop een bericht wordt gelogd door de software, niet correct. Dit heeft tot gevolg dat het tijdvenster waarin wordt gekeken of de verticale acceleratie een piek vertoont, en dit op het moment dat de parameter met betrekking tot de wielsnelheden zijn drempelwaarde heeft overschreden, niet kan worden toegepast. Om deze reden is van het resultaat een gemiddelde genomen van de wielsnelheden en de verticale acceleratie.

Desalniettemin wordt van de functie gemiddelde waarden teruggekregen, welke overeenkomen met de visuele waarnemingen. Het voertuig reed op het moment van de bewuste put met een afgelezen snelheid van 45 km/u. Visueel was uit de grafieken af te leiden dat dit echter 43 km/u bedroeg. Het algoritme geeft dit ook als resultaat terug, samen met een aannemelijke gemiddelde waarde van de verticale acceleratie. Er mag dan ook besloten worden dat het algoritme welke uit simulatie verkregen werd, ook in praktijk met succes functioneert. Uiteraard mits enkele aanpassingen wat betreft de drempelwaarden.

4.3 Testritten op een geselecteerd traject

Nadat het algoritme op zijn praktische werking is geverifieerd, kan het algoritme zijn mogelijkheden verder onderzocht worden. Daarom zal deze paragraaf niet enkel en alleen het detecteren van putten in het wegdek onderzoeken, maar eerder algemeen het detecteren van defecten in het wegdek. Het testvoertuig betreft nog steeds een Ford Focus, met als bouwjaar 2007.

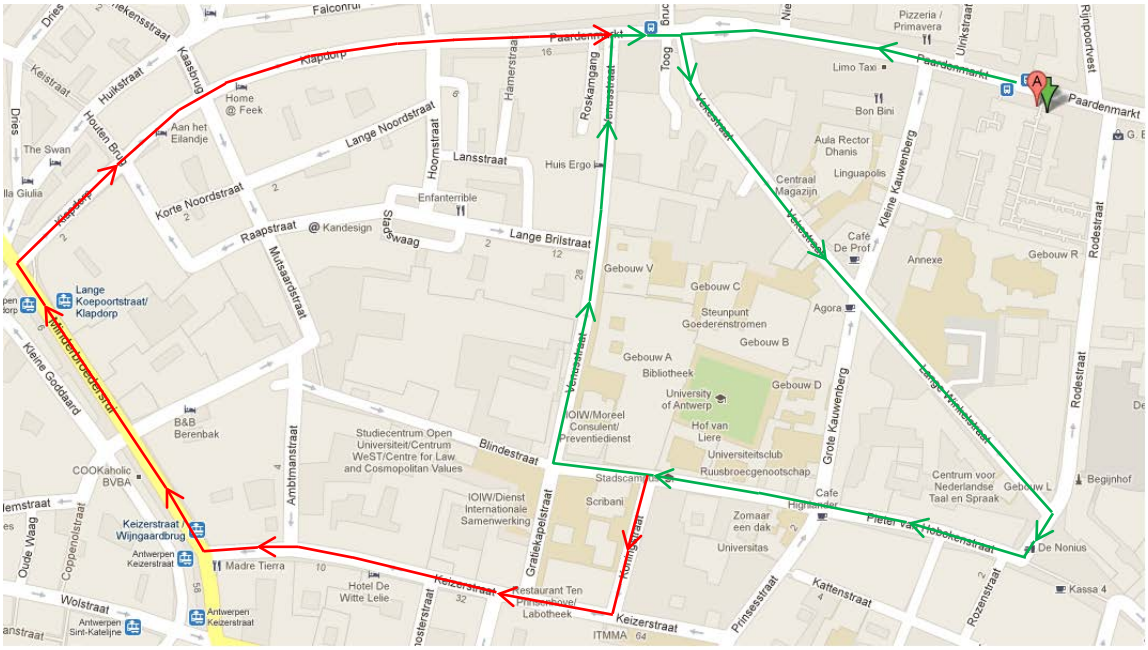
Om dit te verwezenlijken is ervoor gekozen om in de buurt van Campus Paardenmarkt twee testtrajecten te selecteren. De, voor de weggebruiker, meest storende defecten in het wegdek van de geselecteerde testtrajecten, worden geïnventariseerd.

In ‘Figuur 4-6’ worden de geselecteerde testtrajecten aangeduid op een kaart. Bij het groene testtraject is er vertrokken vanaf de campus, waarna er driemaal het traject Vekestraat – Lange Winkelstraat – Pieter van Hobokenstraat – Venusstraat wordt doorlopen. Er wordt hierbij getracht om bij elke rondrit door dezelfde defecten te rijden. Dit heeft tot doel om twee onbekende variabelen te onderzoeken, namelijk:

- Is de GPS-data afkomstig van de smartphone accuraat genoeg?
- Volstaat één ronde om alle defecten in het wegdek te detecteren, of zijn er meerdere rondritten noodzakelijk? (‘Open punt 4’ uit de doelstellingen)

Het rode testtraject is een uitbreiding van het groene traject, waarbij driemaal het traject Vekestraat – Lange Winkelstraat – Pieter van Hobokenstraat – Koningstraat – Keizerstraat –

Minderbroedersrui – Klapdorp – Paardenmarkt wordt doorlopen. De meting van dit traject wordt echter gestart bij het begin van de Vekestraat.



Figuur 4-6: Testtrajecten[6]

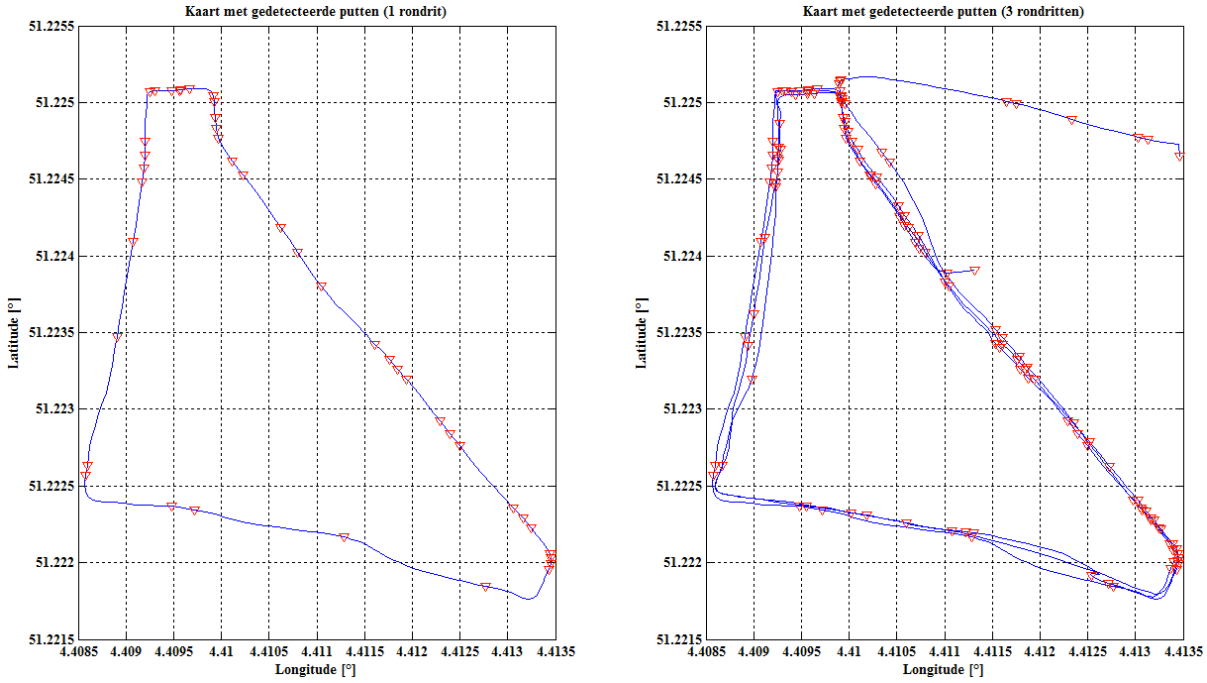
Daar tijdens deze testritten de softwareproblemen verholpen zijn, wordt tijdens deze ritten de functie uitgevoerd zoals beschreven in bijlage ‘XIII Functie “findAnomalies”’. Hierbij wordt opnieuw gekozen voor een drempelwaarde van 1.3 km/u, en een tolerantie van 3.5 %. Opnieuw is gekozen om een sliding window toe te passen van 100 ontvangen CAN-berichten, wat overeenkomt met een tijdvenster van 115 ms. De grootte van dit venster is echter van geen enkel belang. Het wordt enkel gebruikt om het met mondjesmaat verkrijgen van CAN-berichten te simuleren. Zodoende wordt aangetoond dat het algoritme ook in real-time zal functioneren.

Merk op dat door het verschil in samplefrequentie tussen de CAN-gegevens (75 Hz) en de gegevens afkomstig van de accelerometer (25 Hz), er binnen een goed gekozen tijdsvenster bepaald wordt of de drempelwaarde van de verticale acceleratie wordt overschreden.

Het resultaat na het volgen van het groene traject is voorgesteld in ‘Figuur 4-7’. De blauwe lijnen op beide grafieken zijn het resultaat van het plotten van de latitude en longitude, afkomstig van de, in de smartphone, ingebouwde GPS. De door het algoritme gedetecteerde defecten in het wegdek, worden als rode driehoeken weergegeven. Een samenvatting van welke soort defecten er gedetecteerd worden, is toegevoegd als bijlage ‘XIV Traject met selectieve inventaris’, waarbij een selectie is gemaakt uit de opgestelde inventaris. Na vergelijking van de inventaris met de door het algoritme aangeduide defecten, is gebleken dat er niet enkel putten met de vooropgestelde dimensies kunnen gedetecteerd worden. Rioolddeksel welke sterk verzonken zijn, verzakkingen in het wegdek en slecht aangebrachte of hervallen herstellingen kunnen eveneens door het algoritme worden gedetecteerd.

[6] https://maps.google.com/

Om ‘Open punt 4’ uit de doelstellingen te beantwoorden, dienen de linkse en de rechtse grafiek van ‘Figuur 4-7’ met elkaar te worden vergeleken. De linkse grafiek stelt de defecten in het wegdek voor welke in één enkele rit worden gedetecteerd. De rechtse grafiek stelt deze voor na driemaal hetzelfde traject te hebben afgelegd. Wanneer de inventaris van het testtraject wordt vergeleken met het resultaat in de linkse grafiek, dan kan er worden geconcludeerd dat één enkele rondrit de meest kritische defecten detecteert.



Figuur 4-7: Groene testtraject met gedetecteerde putten

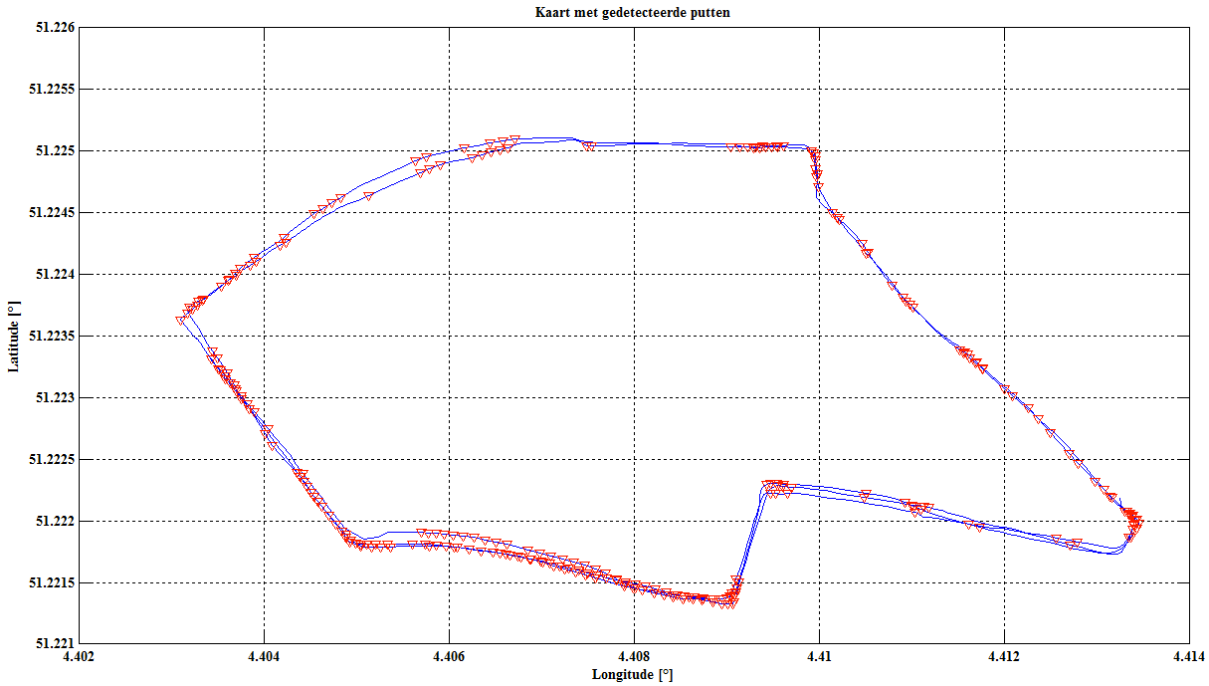
Echter, de defecten welke zich bevinden op de grens van de mogelijkheden van het algoritme, kunnen mogelijk niet door één enkele rit als dusdanig worden aangeduid. Een voorbeeld van een dergelijk defect welke zich op de grens van het mogelijke bevindt, is voorgesteld in ‘Figuur 4-8’. Het betreft hier een herstelling, waarbij het omringende asfalt begint te verzakken. De verzakking is echter plaatselijk een centimeter diep, eveneens weinig storend voor de weggebruiker, en moeilijk te detecteren door het algoritme.

Figuur 4-8: Defect welke mogelijk niet wordt gedetecteerd

Het verlagen van de drempelwaarde met betrekking tot de wielsnelheden zou een oplossing bieden. De vraag is echter of het wenselijk is om dergelijke defecten te lokaliseren.

Het stuk weg tussen de Lange Winkelstraat en de Pieter van Hobokenstraat is een kasseistrook welke in slechte staat verkeerd. In bijlage ‘XIV Traject met selectieve inventaris’ wordt hiervan een inventarisfoto afgebeeld. Na onderzoek, en zoals ‘Figuur 4-7’ aantoont, is gebleken dat het algoritme deze defecten detecteert, en dit zonder de gehele kasseistrook als een opeenvolging van defecten te detecteren.

De resultaten van het volgen van het rode traject, waarbij de Keizerstraat en de Minderbroedersrui bestaan uit kasseistenen, zijn tegenstrijdig met deze laatste bevindingen. Zoals ‘Figuur 4-9’ aantoont worden deze kasseistroken gedetecteerd als een opeenvolging van defecten.

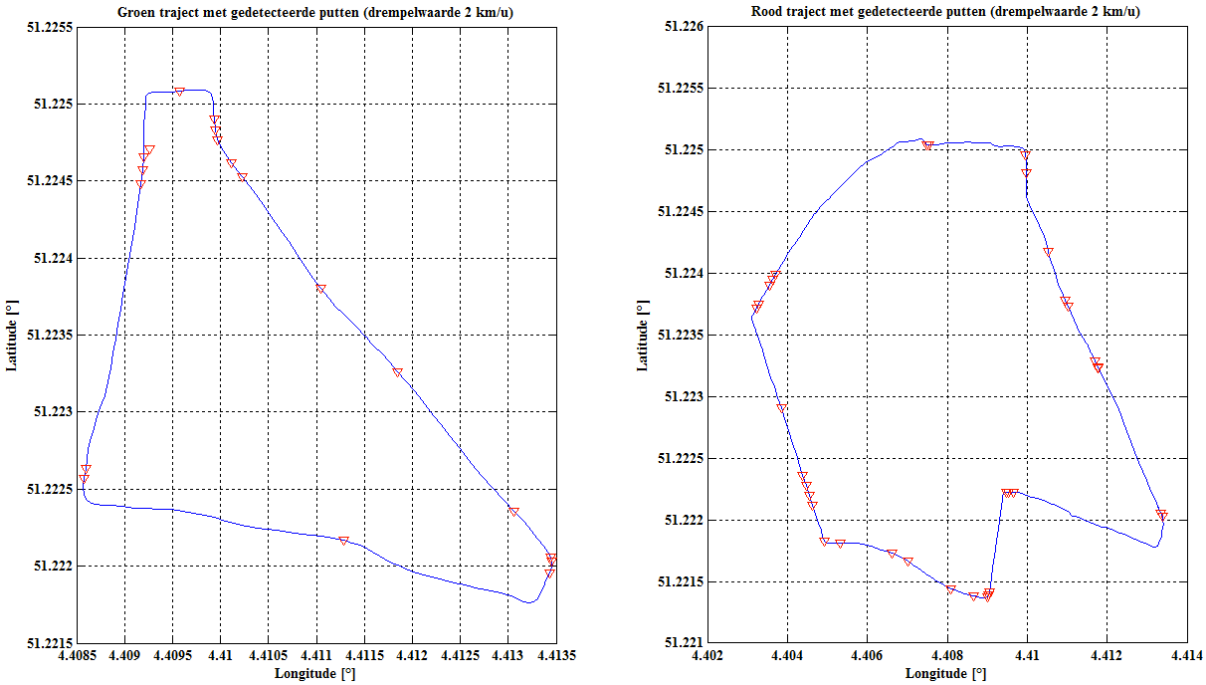


Figuur 4-9: Rode testtraject met gedetecteerde putten

Analyse heeft uitgewezen dat een verhoging van de drempelwaarde tot 2 km/u, een oplossing biedt voor dit probleem. Keerzijde van deze oplossing is dat kleinere defecten, welke al dan niet storend kunnen zijn voor de weggebruiker, niet zullen worden gedetecteerd. Het gevolg van deze verhoogde drempelwaarde is voorgesteld in ‘Figuur 4-10’, en dit zowel voor het groene als voor het rode traject. Indien in dit geval de inventaris vergeleken wordt met de gedetecteerde defecten, is vast te stellen dat één enkele rit nog steeds volstaat om de meeste van de ernstige defecten te detecteren.

Wat nog niet besproken is, is de accuraatheid van de GPS. Zoals te zien in de grafische uitzettingen in ‘Figuur 4-7’ en ‘Figuur 4-9’, is op deze data enige afwijking waar te nemen. Zo kan voor één enkele put, de positie enkele meters verschillen per detectie. Dit geeft de indruk dat er meer putten gedetecteerd worden bij meerdere ritten over hetzelfde traject, terwijl dit te wijten is aan de precisie van de GPS-data. Dit kan verholpen worden door een GPS met een hogere samplefrequentie te gebruiken, of door op een server waar de binnenkomende data verwerkt wordt, een algoritme te ontwikkelen welke meerdere detectiepunten binnen eenzelfde straal als één detectiepunt aanziet.

Voor het VIM is de precisie van de GPS-data echter weinig relevant. Dit omdat het VIM niet geïnteresseerd is individuele defecten, maar eerder naar wegsegmenten welke in slechte staat verkeren. Dit maakt het financieel en technisch interessanter om herstellingen uit te voeren.



Figuur 4-10: Effect van het verhogen van de drempelwaarde

4.4 Conclusie

De verificatie van het simulatiemodel heeft uitgewezen dat een algoritme verkregen uit simulatie, in de praktijk kan toegepast worden. Er dient evenwel rekening gehouden te worden dat sommige drempelwaarden aangepast moeten worden, in overeenstemming met de praktische meetgegevens. Het is daarom aan te raden om bij een toekomstige praktische implementatie van een algoritme, welke ontwikkeld is met behulp van simulatie, de afzonderlijke parameters in de praktijk te analyseren en deze eventueel aan te passen. Dit naar analogie met de aanpassing van de parameter, welke betrekking heeft tot de wielsnelheden, in paragraaf ‘4.2.2.1 Verificatie van de wielsnelheden’.

Er is ook aangetoond dat er voor de samplefrequentie wel degelijk een ondergrens bestaat. Een samplefrequentie van 8.7 Hz bleek onvoldoende te zijn om defecten in het wegdek te detecteren, terwijl dit bij 75 Hz geen problemen opleverde.

Een uitgebreide praktische test op twee geselecteerde testtrajecten heeft uitgewezen dat putten met een diameter van 20 cm en een diepte van 4 cm, zoals vooropgesteld in de doelstellingen, zonder enig probleem kunnen gedetecteerd worden. Het is zelfs zo dat het algoritme in staat is om andere defecten in het wegdek, welke als storend kunnen bevonden worden door de weggebruiker, te detecteren.

Er dient echter afgewogen te worden welke graad van defecten er door het algoritme wensen gedetecteerd te worden. Dit kan door de parameter met betrekking tot de wielsnelheden aan te passen. Wordt deze bijvoorbeeld laag gekozen, is de gevoeligheid van het algoritme

zodanig dat enerzijds kleine defecten in het wegdek gedetecteerd worden, maar dat anderzijds kasseistroken eveneens als defecten worden aanzien. De parameter met betrekking tot de tolerantie op de verticale acceleratie lijkt weinig invloed te hebben, en wordt best zo gekozen dat deze de witte ruis net kan weg filteren.

Deze praktische testen werden afgenomen aan snelheden van maximaal 50 km/u. Daar deze overeenkomen met de simulaties, mag worden aangenomen dat ook voor hogere snelheden dit algoritme naar wens zal functioneren.

5 EINDCONCLUSIE

Uit deze thesis kan geconcludeerd worden dat het detecteren van putten, en meer algemeen defecten, in het wegdek mogelijk is door gebruik te maken van een eenvoudig algoritme bestaande uit gegevens afkomstig van de CAN-bus. Indien gebruik wordt gemaakt van een krachtig simulatiepakket, zoals AMESim, kan op een eenvoudige en efficiënte manier een theoretisch werkend algoritme worden bepaald, welke mits kleine praktische aanpassingen in de praktijk kan geïmplementeerd worden.

Een belangrijke voorwaarde is de samplefrequentie van de parameters waaruit het algoritme is samengesteld. Uit simulatie is geconcludeerd dat deze minstens 25 Hz dient te zijn, wanneer defecten dienen gedetecteerd te worden aan snelheden tot 120 km/u. Bij lagere snelheden mag de samplefrequentie lager gekozen worden, tot een theoretische 18 Hz. De theorie werd later bevestigd in de praktijk. Bij de praktische verificatie van de samplefrequentie, aan een voertuigsnelheid van 45 km/u, is gebleken dat een samplefrequentie van 8.7 Hz onvoldoende is om putten met een diameter van 40 cm en een diepte 4 cm te detecteren. Een samplefrequentie van 75 Hz leverde geen problemen op met betrekking tot het detecteren van defecten. Daarmee is aangetoond is dat er wel degelijk een ondergrens bestaat voor de samplefrequentie, in overeenstemming met de simulaties.

Indien aan de voorwaarde voldaan wordt, is vastgesteld dat één enkele rit voldoende is om ernstige defecten in het wegdek te detecteren. Kleinere defecten, welke als weinig storend worden ondervonden door de wegebruiker, leunen aan tegen de grens van de mogelijkheden van het algoritme, waardoor meerdere ritten noodzakelijk zijn ter detectie.

Door de eenvoud van het algoritme, is het mogelijk om dit op termijn embedded te implementeren in het voertuig, zonder de noodzaak aan een omvangrijke rekencapaciteit. Aangezien het algoritme getest is op zijn werking in het geval dat de CAN-gegevens met mondjesmaat beschikbaar komen, is er eveneens geen behoefte aan een omvangrijke opslagcapaciteit in het geval van een embedded implementatie. Het algoritme is echter niet universeel, en de drempelwaarde van de parameter met betrekking tot wielsnelheden zal waarschijnlijk moeten ingesteld worden per voertuigmodel.

6 TOEKOMSTIG WERK

Een optimalisatie van het simulatiemodel behoort zeker tot een toekomstvisie. Door het simulatiemodel nog dichter bij de werkelijkheid te laten aanleunen, kunnen verdere analyses versneld worden zonder de noodzaak aan praktische testritten. Hiervoor is echter een samenwerking wenselijk met personen met een mechanische achtergrond. Bijkomend kan dan onderzocht worden welke simulatieparameters invloed hebben op het algoritme.

De grootste beperking aan het huidige algoritme, is de keuze welke wordt afgedwongen bij de wegdekbeheerder. Of er wordt gekozen om kleine defecten, welke minder storend zijn voor de weggebruiker, alsnog te detecteren, met als nadeel dat een kasseistrook als een opeenvolging van defecten kan worden aanzien. Ofwel wordt dit nadeel weggewerkt, met als gevolg dat kleine defecten niet worden gedetecteerd.

Verder is het onderzoek naar een universeel algoritme, waarbij de drempelwaarde met betrekking tot de wielsnelheden automatisch kan worden ingesteld, zeer interessant. Er wordt hierbij gedacht aan het gebruik van wiskundige functies zoals Kalman filtering of reeksontwikkelingen zoals de Taylorreeksen.

BIBLIOGRAFIE

[1] Benbow, E. et al., Deliverable 3.1 Demonstration of Methods for the Measurement of Condition using Probe Vehicles, 2008 (INTRO project number 012344)

[2] Hermans, T., Discovering the meaning of unstructured data on a CAN motor management network, Hoboken, KdG dep. IWT, 2009 (eindwerk KdG master in de Industriële Wetenschappen Elektronica-ICT afstudeerrichting Automotive Engineering)

[3] LMS, LMS Imagine.Lab AMESim, [Online februari 2013], *http://www.lmsintl.com/LMS-Imagine-Lab-AMESim*

[4] uCAN, uCan Automotive Solutions, [Online februari 2013], *http://www.ucan-solutions.com/nl/oplossingen/ucan-automotive/*

[5] NXP, Telematics On Board, [Online februari 2013], *http://www.nxp.com/products/automotive/telematics_on_board_unit/*

[6] TASS TSS, uCAN: autorijden in kaart gebracht, [Online februari 2013], *http://www.tass.nl/nl-nl/opdrachtgevers/cases/ucan/*

[7] Beijer, Beijer Automotive, [Online februari 2013], *http://www.beijer.com/CANSolutions.aspx?LanguageID=41*

[8] LaQuSo, LaQuSo, [Online februari 2013], *http://www.laquso.com/*

[9] Bosch, CAN Specification 2.0, 1991, *http://esd.cs.ucr.edu/webres/can20.pdf*

[10] ISO, Controller Area Network, [Online februari 2013], *http://www.iso.org/iso/home/search.htm?qt=Controller+Area+Network&sort=rel&type=simple&published=on*

[11] Matheus, K., Ethernet in cars: an idea whose time has come, 2012, [Online februari 2013], *http://www.sae.org/mags/aei/11142*

[12] Paret, D., Multiplexed Networks for Embedded Systems; CAN, LIN, FlexRay, Safe-by-Wire, ... Warrendale, SAE International, 2007

[13] Nice, K., How Anti-Lock Brakes Work, 2000, [Online februari 2013], *http://auto.howstuffworks.com/auto-parts/brakes/brake-types/anti-lock-brake.htm*

[14] Burton, D. et al., Effectiveness of ABS and Vehicle Stability, p. 3–16, 2004

[15] the Editors of Publications International, Traction Control Explained, 2005, [Online februari 2013], *http://auto.howstuffworks.com/28000-traction-control-explained.html*

[16] Hall-Geisler, K., How Electronic Stability Control Works, 2009, [Online februari 2013], *http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/electronic-stability-control.htm*

[17] IPG, CarMaker, [Online februari 2013], *http://www.ipg.de/index.php?id=carmaker*

[18] uCAN, uCAN, [Online februari 2013], *http://www.ucan-solutions.com/ucan/*

[19] NXP, Take pole position in vehicle Telematics solutions, p. 1–2, 2010

[20] NXP, A standard approach to eCalling, p. 1–2, 2008

[21] Verkeerskunde, Brabant in–car II: Wat zijn de resultaten, 2013, [Online april 2013], *http://www.verkeerskunde.nl/Uploads/2013/3/Eindrapport-GC.pdf*

[22] VIM, Sensovo, [Online maart 2013], *http://www.vim.be/projects/sensovo*

[23] MathWorks, Simulink, [Online april 2013], *http://www.mathworks.nl/products/simulink/*

[24] MathWorks, Matlab, [Online april 2013], *http://www.mathworks.nl/products/matlab/*

[25] Pedley, M., Tilt Sensing Using a Three-Axis Accelerometer, p. 5–12, 2013

[26] Astarita, V. et al., A mobile application for road surface quality control : UNIquALroad, 2012

I SIMULATIEMODEL

Op de volgende pagina is het model weergegeven welke gebruikt werd tijdens de simulaties in AMESim.

Dit model is eveneens terug te vinden op de CD-ROM onder de bestandsnaam “simulatiemodel”, en dit in de folder ‘Simulatie\AMESim’.

run stats
Simulation statistics
Steering wheel angle
Velocity profil
Braking system
ESP
ABS/ASR
Manual gearbox - 5 forward gears
Engine torque
Clutch
cte torque =
cte pedal position
Clutch
primary shaft
secondary shaft
Power axle
gear 1
gear 2
gear 3
gear 4
gear 5
Powertrain system
Vehicle Dynamics Post processing
X
Y
Z
Position of Carbody center of gravity
Acceleration of Carbody fixed point
Velocity of Carbody center of gravity
Acceleration of Carbody center of gravity
Roll angle
Pitch angle
Yaw angle
Roll velocity
Pitch velocity
Yaw velocity
Over steering at wheel
Front axle system sideslip (corrected with mean kin. steer. angle)
Mean kinematic steering angle
Assistance law
pinion-rack
Mechanical Power Steering System
Wind velocity
Aerodynamics
Chassis
Calipers (Rear Axles)
Calipers (Front Axles)
Suspension System (rear axles)
End stop (lower)
End stop (higher)
Damper
Spring
Antiroll bar
Spring
Damper
End stop (higher)
End stop (lower)
Suspension System (front axles)
End stop (lower)
End stop (higher)
Damper
Spring
Antiroll bar
Spring
Damper
End stop (higher)
End stop (lower)
REAR RIGHT WHEEL [22]
REAR LEFT WHEEL [21]
FRONT RIGHT WHEEL [12]
FRONT LEFT WHEEL [11]
Elastokinematics (Rear Axle)
Elastokinematics (Front Axle)
Ground normal expressed in ground frame
Height of the ground expressed in ground frame
Rate of climb of the ground expressed in ground frame
Absolute velocity of wheel center A2
Absolute location of wheel center A2

II FUNCTIE "GETAMERESULTS"

Om wiskundige bewerkingen op de gesimuleerde data mogelijk te maken, is ervoor gekozen om de gesimuleerde data te importeren in Mathworks Matlab.

De functie "getAmeResults" kan door het meegeven van 3 argumenten, een simulatie starten in AMESim, waarna de parameters geïmporteerd worden in de Matlab-omgeving. Een voorbeeld van het aanroepen van deze functie is:

[data headers] = getAmeResults('simulationModel', 150, 50);

Hiervoor dient het simulatiemodel wel geopend te zijn in AMESim, en dient het gemodelleerde simulatiemodel reeds eenmaal gesimuleerd zijn binnen de AMESim-omgeving.

Deze code is eveneens terug te vinden op de CD-ROM onder de bestandsnaam "getAmeResults", en dit in de folder 'Simulatie\MATLAB'.

```
function [ data headers ] = getAmeResults( projectname, simulationTime, sampleTime )
%set names
projectname = strcat(projectname, '_');       %filename: 'filename_'
pname = strcat(projectname,'.param');
sname = strcat(projectname,'.sim');
vname = strcat(projectname,'.var');

%get variable names from unique identifiers
rollAngle_varname = char(amegetvarnamefromui(pname,'xEulerAngle@RelativeAngleSensor'));
pitchAngle_varname = char(amegetvarnamefromui(pname,'yEulerAngle@RelativeAngleSensor'));
yawAngle_varname = char(amegetvarnamefromui(pname,'zEulerAngle@RelativeAngleSensor'));

rollVelocity_varname = char(amegetvarnamefromui(pname,'xEulervel@RelativeEulerVelocitySensor'));
pitchVelocity_varname = char(amegetvarnamefromui(pname,'yEulervel@RelativeEulerVelocitySensor'));
yawVelocity_varname = char(amegetvarnamefromui(pname,'zEulervel@RelativeEulerVelocitySensor'));     %ESPYaw

%longitudinal acceleration
xAcceleration_varname = char(amegetvarnamefromui(pname,'aPxRexpressionframe@AbsoluteAccelerationSensor'));
%lateral acceleration
yAcceleration_varname = char(amegetvarnamefromui(pname,'aPyRexpressionframe@AbsoluteAccelerationSensor'));
%vertical acceleration
zAcceleration_varname = char(amegetvarnamefromui(pname,'aPzRexpressionframe@AbsoluteAccelerationSensor'));

velocity_varname = char(amegetvarnamefromui(pname,'vPxRexpressionframe@AbsoluteVelocitySensor'));   %ESPVelocity

clutch_varname = char(amegetvarnamefromui(pname,'fofx@fofx'));
RPM_varname = char(amegetvarnamefromui(pname,'w@rotaryload2ports_2_2_2'));

steeringRackDis_varname = char(amegetvarnamefromui(pname,'xdup@displacementsensor'));    %ESPSteeringRackPos
steeringRackAngle_varname = char(amegetvarnamefromui(pname,'theta1@vdrack')); %sensor attached to steering column
steeringRackSpeed_varname = char(amegetvarnamefromui(pname,'v2@vdrack'));     %sensor attached to steering column

velocityFL11_revMin_varname = char(amegetvarnamefromui(pname,'wdup@rotaryspeedsensor'));            %front left
velocityFR12_revMin_varname = char(amegetvarnamefromui(pname,'wdup@rotaryspeedsensor_1'));          %front right
velocityRL21_revMin_varname = char(amegetvarnamefromui(pname,'wdup@rotaryspeedsensor_3'));          %rear left
velocityRR22_revMin_varname = char(amegetvarnamefromui(pname,'wdup@rotaryspeedsensor_2'));          %rear right
```

```
% run simulation
[Results,Varnames,sname,retval,msg] = amerun(sname, 0.0, simulationTime, sampleTime);

%get results
time = amegetvar(Results,Varnames,'time [s]');

rollAngle = amegetvar(Results, Varnames, rollAngle_varname);                    %[ deg ]
pitchAngle = amegetvar(Results, Varnames, pitchAngle_varname);                  %[ deg ]
yawAngle = amegetvar(Results, Varnames, yawAngle_varname);                      %[ deg ]

rollVelocity = amegetvar(Results, Varnames, rollVelocity_varname);              %[ deg/s ]
pitchVelocity = amegetvar(Results, Varnames, pitchVelocity_varname);            %[ deg/s ]
yawVelocity = amegetvar(Results, Varnames, yawVelocity_varname);                %[ deg/s ]

xAcceleration = amegetvar(Results, Varnames, xAcceleration_varname);            %[ m/s/s ]
yAcceleration = amegetvar(Results, Varnames, yAcceleration_varname);            %[ m/s/s ]
zAcceleration = amegetvar(Results, Varnames, zAcceleration_varname);            %[ m/s/s ]

velocity = amegetvar(Results, Varnames, velocity_varname);                      %[ m/s ]

clutch = amegetvar(Results, Varnames, clutch_varname);                          %[ none ]
RPM = amegetvar(Results, Varnames, RPM_varname);                                %[ rev/min ]

steeringRackDis = amegetvar(Results, Varnames, steeringRackDis_varname);        %[ meter ]
steeringRackAngle = amegetvar(Results, Varnames, steeringRackAngle_varname);    %[ degree ]
steeringRackSpeed = amegetvar(Results, Varnames, steeringRackSpeed_varname);    %[ m/s ]

velocityFL11_revMin = amegetvar(Results, Varnames, velocityFL11_revMin_varname);  %[ rev/min ]
velocityFR12_revMin = amegetvar(Results, Varnames, velocityFR12_revMin_varname);  %[ rev/min ]
velocityRL21_revMin = amegetvar(Results, Varnames, velocityRL21_revMin_varname);  %[ rev/min ]
velocityRR22_revMin = amegetvar(Results, Varnames, velocityRR22_revMin_varname);  %[ rev/min ]

%Reshape the results
[data headers] = reorganizeAmeResults(time, rollAngle, pitchAngle, yawAngle, rollVelocity, pitchVelocity,
                  yawVelocity, xAcceleration, yAcceleration, zAcceleration, velocity, clutch, RPM, steeringRackDis,
                  steeringRackAngle, steeringRackSpeed, velocityFL11_revMin, velocityFR12_revMin,
                  velocityRL21_revMin, velocityRR22_revMin);
end
```

III FUNCTIE "REORGANIZEAMERESULTS"

Deze functie wordt automatisch opgeroepen door de functie "getAmeResults". Het doel van deze functie is om de, uit AMESim geïmporteerde, simulatievariabelen te converteren naar kolomvectoren en deze samen te voegen in één grote matrix. Dit maakt het eenvoudiger en overzichtelijker om bewerkingen in Matlab uit te voeren.

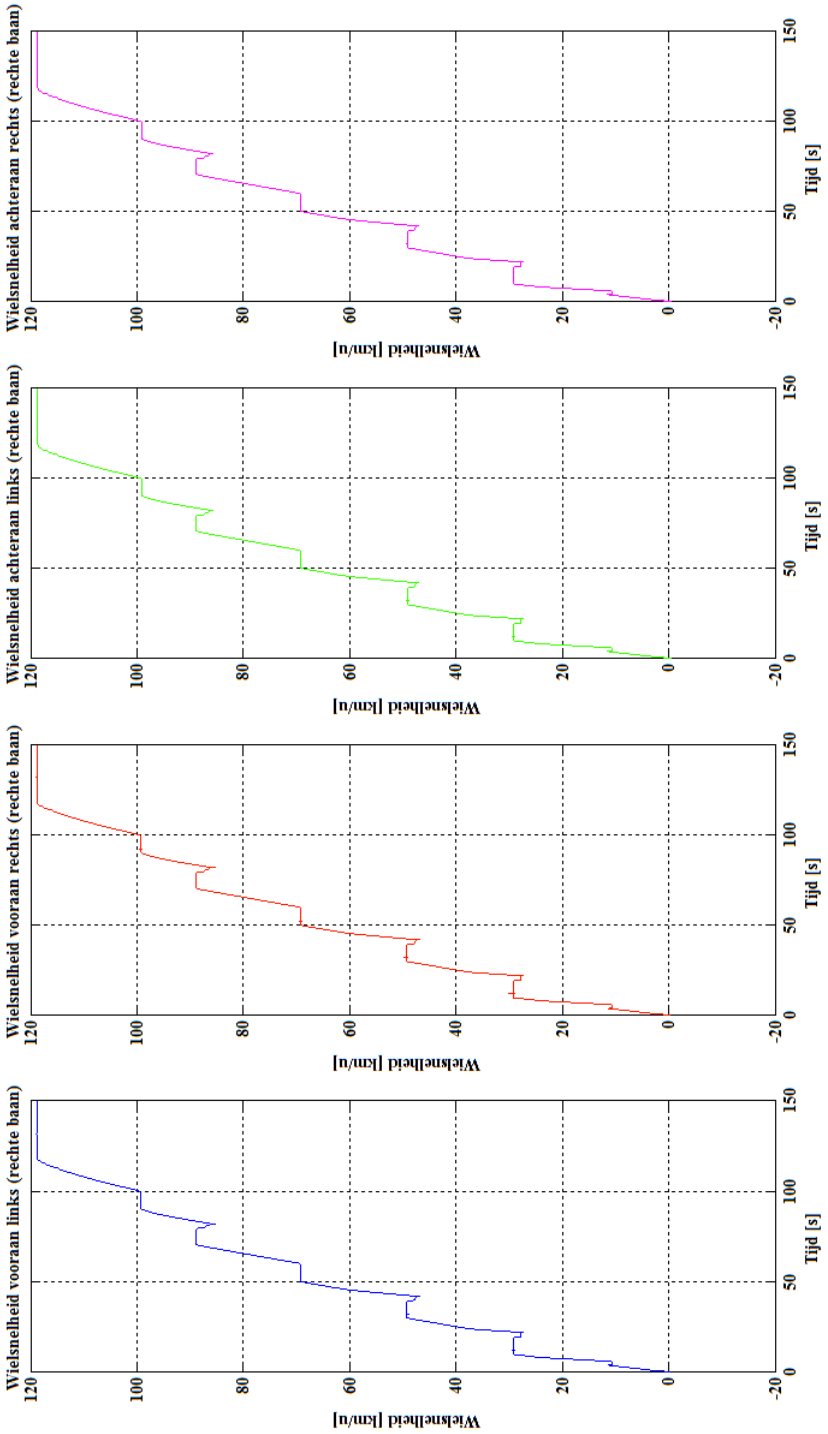
Deze code is eveneens terug te vinden op de CD-ROM onder de bestandsnaam "reorganizeAmeResults", en dit in de folder 'Simulatie\MATLAB'.

```
function [ matrix headers ] = reorganizeAmeResults ( time, rollAngle, pitchAngle, yawAngle, rollVelocity,
          pitchVelocity, yawVelocity, xAcceleration, yAcceleration, zAcceleration, velocity, clutch, RPM,
          steeringRackDis, steeringRackAngle, steeringRackSpeed, velocityFL11_revMin, velocityFR12_revMin,
          velocityRL21_revMin, velocityRR22_revMin )

%convert to row matrix
time = reshape(time,[],1);

rollAngle = reshape(rollAngle,[],1);
pitchAngle = reshape(pitchAngle,[],1);
yawAngle = reshape(yawAngle,[],1);

rollVelocity = reshape(rollVelocity,[],1);
pitchVelocity = reshape(pitchVelocity,[],1);
yawVelocity = reshape(yawVelocity,[],1);

xAcceleration = reshape(xAcceleration,[],1);
yAcceleration = reshape(yAcceleration,[],1);
zAcceleration = reshape(zAcceleration,[],1);

velocity = reshape(velocity,[],1);

clutch = reshape(clutch,[],1);
RPM = reshape(RPM,[],1);

steeringRackDis = reshape(steeringRackDis,[],1);
steeringRackAngle = reshape(steeringRackAngle,[],1);
steeringRackSpeed = reshape(steeringRackSpeed,[],1);

velocityFL11_revMin = reshape(velocityFL11_revMin,[],1);
velocityFR12_revMin = reshape(velocityFR12_revMin,[],1);
velocityRL21_revMin = reshape(velocityRL21_revMin,[],1);
velocityRR22_revMin = reshape(velocityRR22_revMin,[],1);
```

```
%make one matrix
matrix = horzcat( time, velocity, clutch, RPM, velocityFL11_revMin, velocityFR12_revMin, velocityRL21_revMin,
          velocityRR22_revMin, xAcceleration, yAcceleration, zAcceleration, yawVelocity );
headers = { 'time', 'velocity', 'clutch', 'RPM', 'velocityFL11_revMin', 'velocityFR12_revMin',
           'velocityRL21_revMin', 'velocityRR22_revMin', 'xAcceleration', 'yAcceleration', 'zAcceleration',
           'yawVelocity' };

end
```

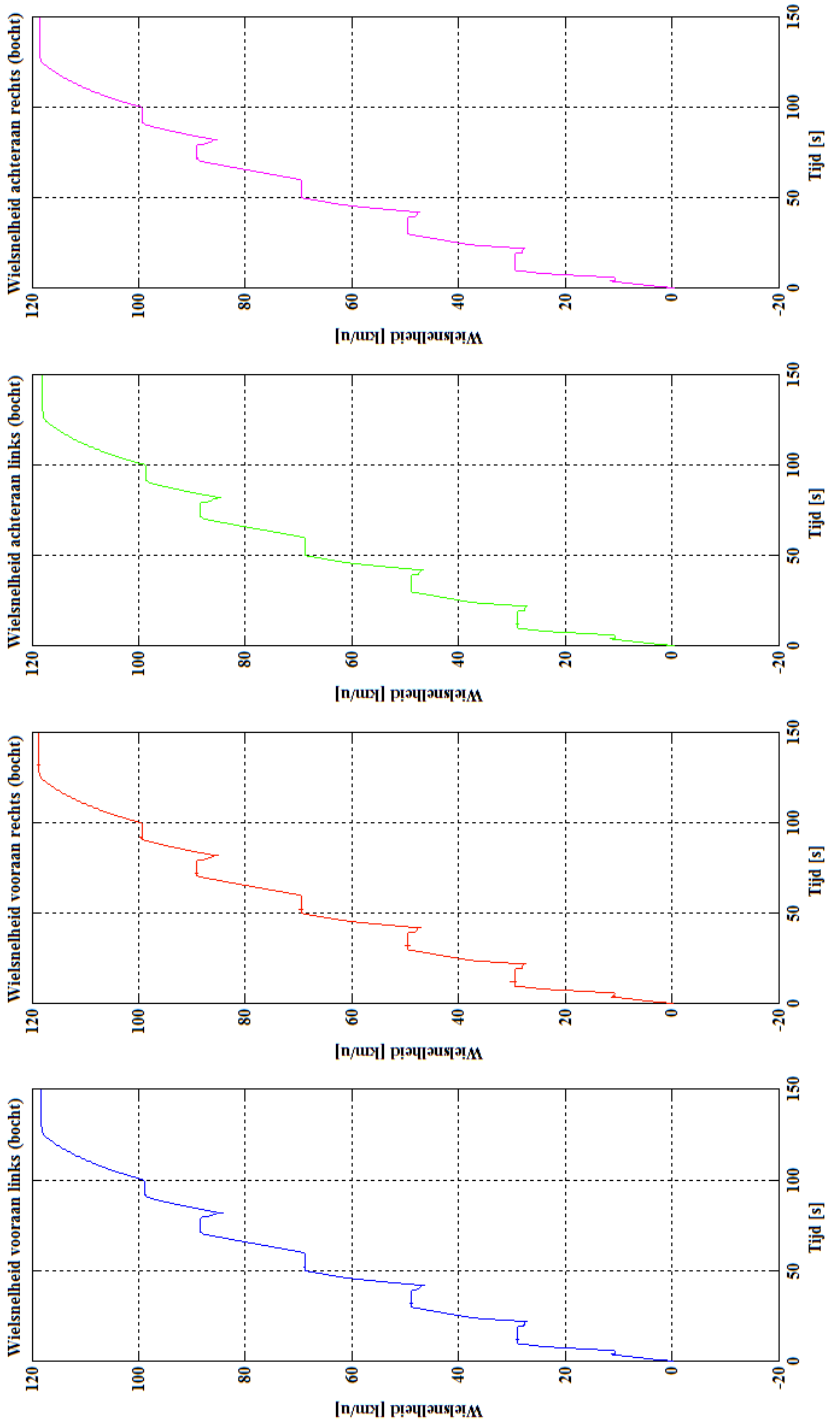
IV WIELSNELHEDEN (RECHTE BAAN)

Onderstaande figuur geeft de wielsnelheden weer, en dit bij het volgen van een rechte baan. Het betreft hier om de wielsnelheden verkregen na een simulatie van het simulatiemodel.



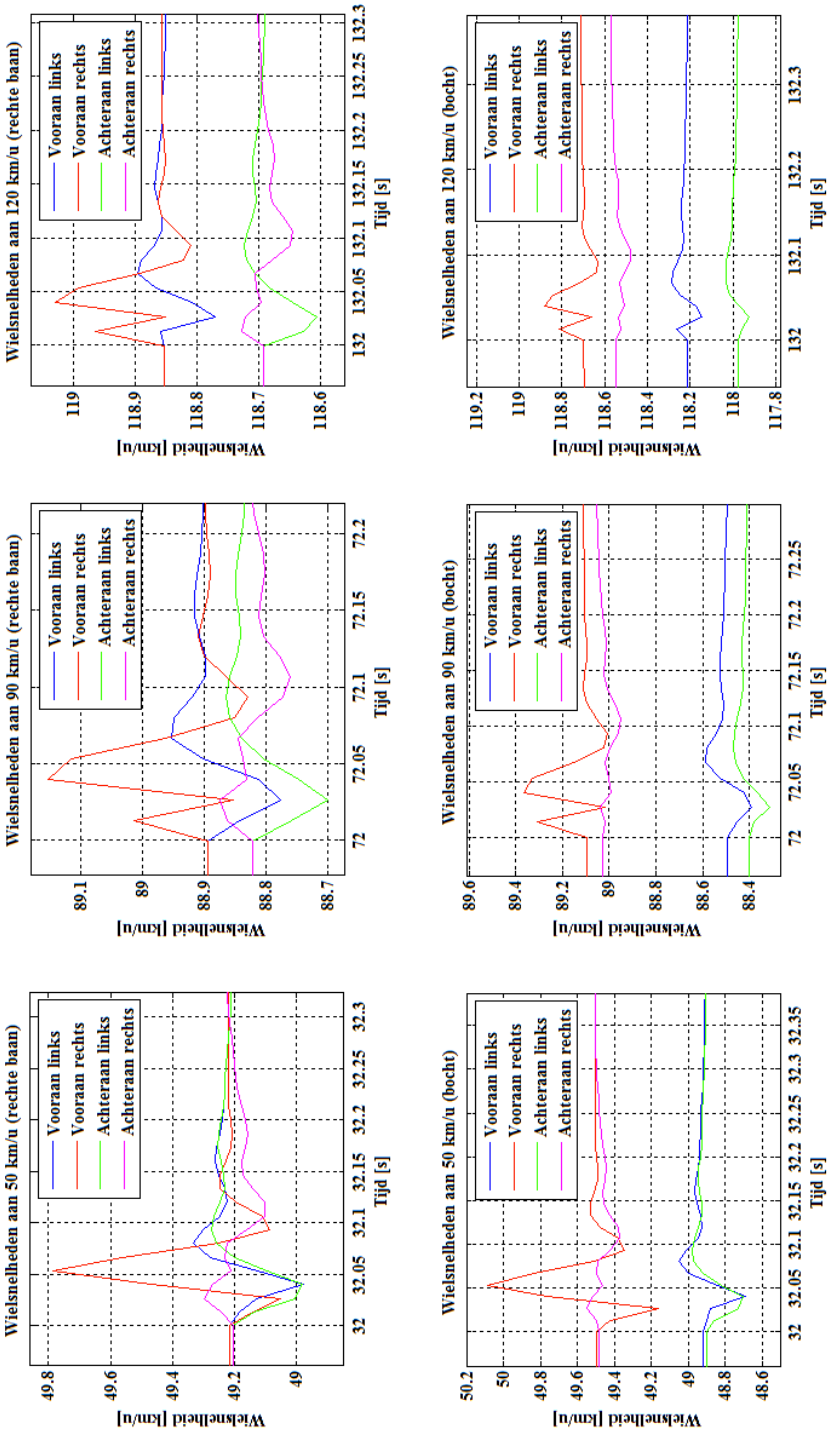
V WIELSNELHEDEN (BOCHT)

Onderstaande figuur geeft de wielsnelheden weer, en dit bij het volgen van een rechte baan. Het betreft hier om de wielsnelheden verkregen na een simulatie van het simulatiemodel.



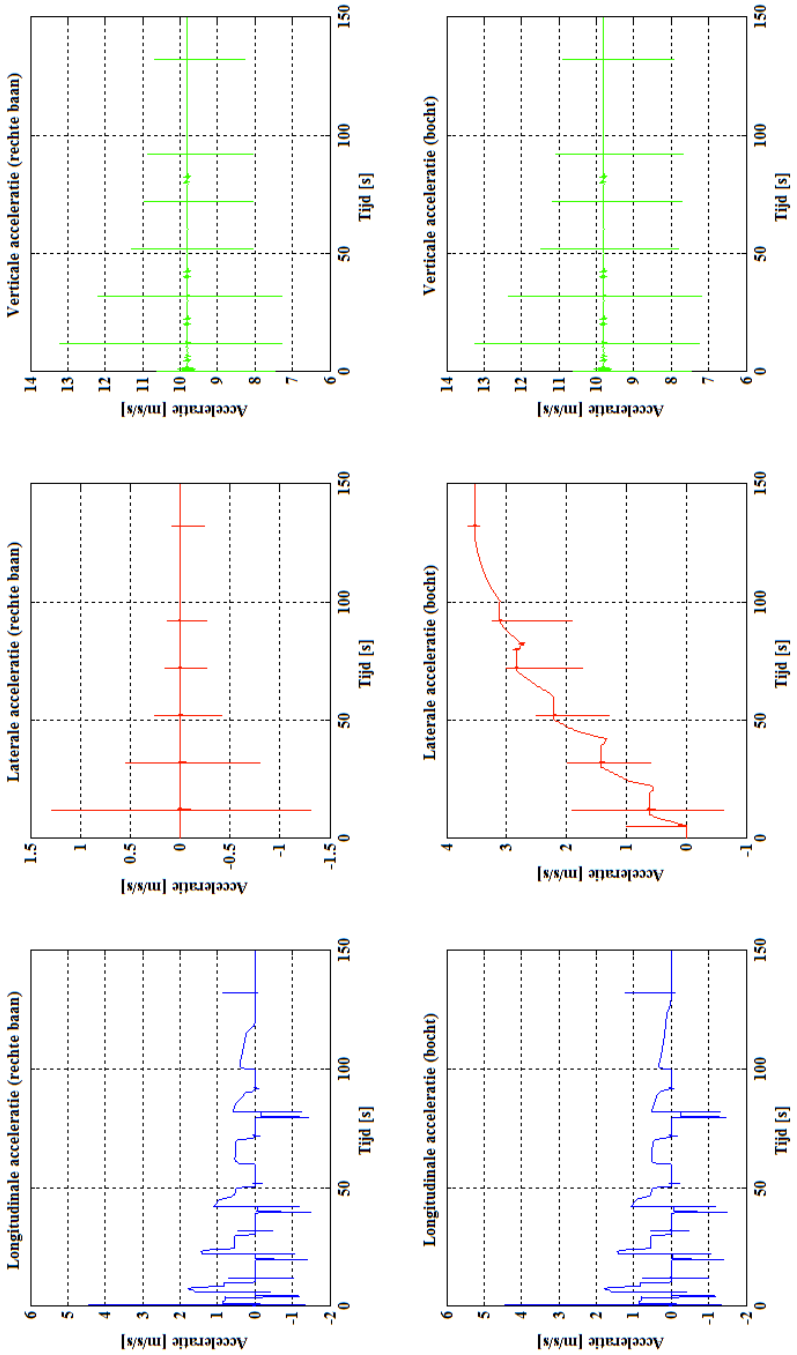
VI VERGELIJKING VAN DE WIELSNELHEDEN

In onderstaande figuur wordt ingezoomd bij enkele snelheden, op het moment dat zich een put in het wegdek voordoet, en dit bij het simuleren van het simulatiemodel.



VII GEGEVENS ACCELEROMETER

Onderstaande figuur geeft de gegevens van accelerometer weer tijdens een simulatie, en dit zowel bij het volgen van een rechte baan, als bij het nemen van een bocht.



VIII FUNCTIE "SMALLPEAKFILTER"

Met dit filter kan op een eenvoudige wijze ruis weg gefilterd worden. Als argumenten dient aan deze functie een set aan datagegevens, een gemiddelde en een drempelwaarde mee gegeven te worden. De gefilterde data wordt tot slot teruggekeerd naar het programma welke deze functie heeft opgeroepen.

Deze code is eveneens terug te vinden op de CD-ROM onder de bestandsnaam "smallPeakFilter", en dit in de folders 'Simulatie\MATLAB' en 'Praktisch\MATLAB'.

```
function returnData = smallPeakFilter(data, mean, threshold)
%Small Peak Filter
    for i=1:length(data)
        if ( ( data(i) <= ( mean - threshold ) ) || ( data(i) >= ( mean +
            threshold ) ) )
            data(i) = data(i);
        else
            data(i) = mean;
        end
    end

    returnData = data;
end
```

IX FUNCTIE "FINDPITSW"

De functie "findPitSW" kan door het meegeven van 3 argumenten, een algoritme testen op een set aan simulatiedata. De teruggegeven parameters bevatten de door het algoritme gedetecteerde putten met enkele berekende parameters, overeenkomstig met de teruggegeven headers. Een voorbeeld van het aanroepen van deze functie is:

[pits headers] = findPitSW(data, 100, 0.13, 0.035);

Deze code is eveneens terug te vinden op de CD-ROM onder de bestandsnaam "findPitSW" , en dit in de folder 'Simulatie\MATLAB'.

```
function [ detectedPits headers ] = findPitSW(data, windowSize, thresholdWheelSpeed, thresholdAcc)

pits= [];
gravity = 9.80665;          %[ m/s/s ]
tireRadius = 0.292;         %[ m ]

for i=1:windowSize:size(data,1)-windowSize,

    %copy data from the matrix 'data', with the specific window size
    selection = data([i:(i+windowSize)],:);

    time = selection(:,1);
    velocity = selection(:,2) * 3.6;                                        %convert [ m/s ] to [ km/h ]
    frontLeft = selection(:,5) * (1/60) * tireRadius * 2 * pi * 3.6;    %convert [ rev/min ] to [ km/h ]
    frontRight = selection(:,6) * (1/60) * tireRadius * 2 * pi * 3.6;   %convert [ rev/min ] to [ km/h ]
    rearLeft = selection(:,7) * (1/60) * tireRadius * 2 * pi * 3.6;     %convert [ rev/min ] to [ km/h ]
    rearRight = selection(:,8) * (1/60) * tireRadius * 2 * pi * 3.6;    %convert [ rev/min ] to [ km/h ]
    zAcceleration = selection(:,11);

    %suppressing the noise caused by switching gear
    zAcceleration = smallPeakFilter(zAcceleration, gravity, gravity * thresholdAcc);

    for j=1:length(frontLeft)

        diffFront = frontLeft(j) - frontRight(j);
        diffRear = rearLeft(j) - rearRight(j);
        diffFrontRear = diffFront - diffRear;

        if ( ( abs(diffFrontRear) >= thresholdWheelSpeed ) && ( zAcceleration(j) ~= gravity ) )
            pits = [ pits; time(j) velocity(j) frontLeft(j) frontRight(j) rearLeft(j) rearRight(j)
                    diffFront diffRear diffFrontRear zAcceleration(j) ];
        end
    end
end

detectedPits = pits;
headers = { 'time', 'velocity', 'frontLeft', 'frontRight', 'rearLeft', 'rearRight', 'diffVelocityFront',
          'diffVelocityRear', 'diffVelocityFrontRear', 'zAcceleration'};
end
```

X METING VOLVO V70

Volgende tabel geeft het resultaat weer van een logging van de CAN-bus, uitgevoerd op een Volvo V70 met als bouwjaar 2001.

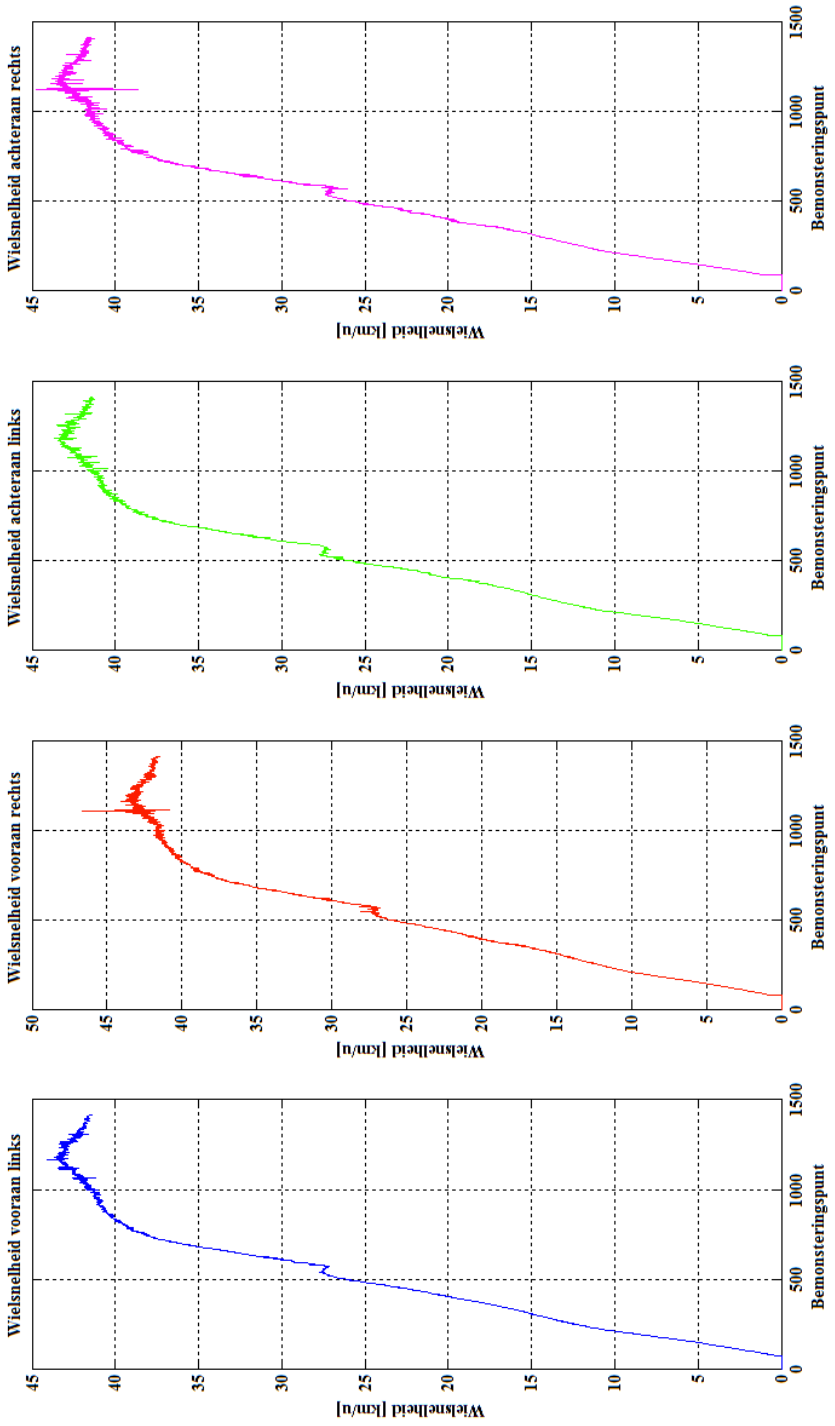
Tijd [µs]	*Tijdsinterval [µs]*	*DB 4*	*DB 5*	*DB 6*	*DB 7*
285,768,220		234	205	74	7
285,884,700	116,480	234	206	75	7
285,995,480	110,780	235	206	75	8
286,106,840	111,360	235	207	76	8
286,219,740	112,900	236	207	77	9
286,332,630	112,890	236	208	77	9
286,447,320	114,690	237	209	78	10
286,558,420	111,100	238	210	79	11
286,671,320	112,900	239	211	80	12
286,785,690	114,370	240	212	81	13
286,897,110	111,420	241	213	82	14
287,011,800	114,690	242	214	83	15
287,122,900	111,100	244	216	85	17
287,235,860	112,960	245	217	86	18
287,348,700	112,840	247	219	88	20
287,462,680	113,980	249	221	90	22
287,575,700	113,020	251	223	92	24
287,687,320	111,620	254	226	95	27
287,800,790	113,470	1	229	98	29
288,027,220	226,430	8	236	105	36
288,138,900	111,680	12	240	109	40
288,251,800	112,900	16	245	113	45
288,365,210	113,410	21	250	118	50
288,479,260	114,050	26	255	123	55
288,590,420	111,160	32	5	129	61
288,703,900	113,480	38	11	135	67
288,816,220	112,320	44	18	142	73
288,929,690	113,470	51	24	148	80
289,044,760	115,070	57	32	155	87
289,154,900	110,140	65	39	163	94
289,267,800	112,900	72	47	170	102
289,381,270	113,470	80	55	178	109
289,495,770	114,500	88	63	186	117
289,607,450	111,680	96	72	194	126

289,719,900	112,450	104	80	203	134
289,832,280	112,380	113	89	212	143
289,947,930	115,650	122	98	221	152
290,059,740	111,810	131	108	230	161
290,170,900	111,160	140	117	240	170
290,284,700	113,800	150	127	249	180
290,396,700	112,000	160	137	3	190
290,511,260	114,560	170	147	14	200
290,622,490	111,230	180	158	24	210
290,735,380	112,890	190	168	34	220
290,848,280	112,900	201	179	45	230
290,963,670	115,390	211	190	56	241
291,075,220	111,550	222	200	67	252
291,187,350	112,130	233	212	78	6
291,299,860	112,510	244	223	89	17
291,413,270	113,410	255	234	100	28
291,527,510	114,240	10	245	112	40
291,638,550	111,040	22	1	123	51
291,751,380	112,830	33	13	135	62
291,864,860	113,480	45	24	147	74
291,978,840	113,980	56	36	159	85
292,091,220	112,380	68	48	170	96
292,203,480	112,260	80	60	183	108
292,315,860	112,380	92	72	195	120
292,428,760	112,900	104	83	207	131
292,543,830	115,070	117	95	219	143
292,655,130	111,300	129	107	231	155
292,767,380	112,250	141	120	244	166
292,880,860	113,480	153	132	0	178
292,994,900	114,040	166	144	13	190
293,107,670	112,770	178	157	25	203
293,218,970	111,300	191	169	38	215
293,331,860	112,890	203	182	51	227
293,445,910	114,050	216	195	64	240
293,558,740	112,830	229	208	77	252
293,670,550	111,810	242	221	90	9
293,785,110	114,560	255	234	103	22
293,896,340	111,230	11	247	116	35
294,009,240	112,900	24	4	129	48

294,122,390	113,150	37	17	143	61
294,234,970	112,580	50	30	156	73
294,347,860	112,890	63	44	169	86
294,460,890	113,030	76	57	182	99
294,575,380	114,490	89	70	195	112
294,686,620	111,240	102	84	208	125
294,800,600	113,980	115	97	222	139
295,027,540	226,940	141	124	248	165

XI WIELSNELHEDEN FORD FOCUS (2007)

Onderstaande figuur geeft de wielsnelheden weer van het testvoertuig waarmee de praktische metingen zijn uitgevoerd. In dit specifieke geval wordt er bij een snelheid van 43 km/u, met het rechtervoorwiel in een put gereden. Deze put heeft een diameter van 40 cm, en een snel oplopende diepte tot 4 cm.



XII FUNCTIE "ACCCALIBRATION"

Om ervoor te zorgen dat de verticale as van een smartphone steeds loodrecht op het voertuig staat, zodat de verticale acceleratie consequent is, kan volgende functie aangewend worden. Door het meegegeven van de ruwe gegevens, afkomstig van de smartphone zijn driedimensionale accelerometer, wordt met behulp van twee vrijheidsgraden een kalibratie doorgevoerd van de verticale acceleratie. Het resultaat wordt teruggegeven naar de oproepende functie.

Deze code is eveneens terug te vinden op de CD-ROM onder de bestandsnaam "accCalibration", en dit in de folder 'Praktisch\MATLAB'.

```
function [ zAcceleration ] = accCalibration( xAcc, yAcc, zAcc )
%reorientation of XYZ-Acceleration parameters

alpha = atan( yAcc ./ zAcc );                                    %roll angle
beta = atan( -xAcc ./ ( sqrt( yAcc.^2 + zAcc.^2 ) ) );           %pitch angle

%use angle to reorientate XYZ-Acceleration
xAccReor = ( cos(beta) .* xAcc ) + ( sin(beta) .* sin(alpha) .* yAcc ) + (
           cos(alpha) .* sin(beta) .* zAcc );
yAccReor = ( cos(alpha) .* yAcc ) - ( sin(alpha) .* zAcc );
zAccReor = ( -sin(beta) .* xAcc ) + ( cos(beta) .* sin(alpha) .* yAcc ) + (
           cos(beta) .* cos(alpha) .* zAcc );

zAcceleration = zAccReor;

end
```

XIII FUNCTIE "FINDANOMALIES"

Deze functie is een praktische implementatie van het, met behulp van simulatie, gevonden algoritme. Als argumenten dienen een set aan CAN-datagegevens, een struct met de gekende CAN-identifiers, een venstergrootte, een drempelwaarde voor de wielsnelheden en een drempelpercentage voor de verticale acceleratie meegegeven te worden.

De functie bestaat uit vier stappen, zijnde:

- Stap 1: er wordt gewacht op de gegevens van de accelerometer, afkomstig van de smartphone.

- Stap 2: zolang het voertuig stilstaat wordt de verticale acceleratie bijgehouden. Van zodra het voertuig in beweging komt, wordt van de bijgehouden gegevens het gemiddelde berekend. Dit gemiddelde is van belang voor de functie "smallPeakFilter".

- Stap 3: wanneer het voertuig in beweging is, wordt het algoritme ter detectie van putten in het wegdek uitgevoerd.

- Stap 4: van zodra het voertuig terug stilstaat, wordt teruggekeerd naar stap 2. Dit is van belang om op die manier de gegevens van de accelerometer opnieuw te kalibreren. De smartphone kan immers door de bestuurder verplaatst zijn.

Het resultaat van deze functie, de gedetecteerde putten, worden teruggegeven naar de oproepende functie.

Deze code is eveneens terug te vinden op de CD-ROM onder de bestandsnaam "findAnomalies", en dit in de folder 'Praktisch\MATLAB'. Een bijhorende struct met de gekende CAN-identifiers voor een Ford Focus met als bouwjaar 2007, is eveneens terug te vinden op de CD-ROM. Dit onder de bestandsnaam "focus", in de folder 'Praktisch\MATLAB'.

```
function [ pits ] = findAnomalies( data, specs, windowSize, thresholdWheelSpeed, toleranceAcc );

meanZAcc = 9.80665;         %in case the initialization step isn't carried out
temp = [];
pitsDef = [];

timeCAN = [];
frontLeft = [];
frontRight = [];
rearLeft = [];
rearRight = [];
detectionWheels = [];

timeGPS = [];
latitude = [];
longitude = [];
xAcceleration = [];
yAcceleration = [];
zAcceleration = [];

for i=1:windowSize:size(data,1)-windowSize

    subData = data([i:(i+windowSize)],:);   %copy from the matrix 'data' with the applied window

    for j=1:length(specs)

        spec = specs(j);

        selection = subData(subData(:,2) == spec.canId, :);
        if (isempty(selection))
            continue
        end
        valueIndices = spec.data;

        values = zeros(size(selection,1),1);
        for k=1:length(valueIndices)
            values = values * 256 + selection(:,valueIndices(k) + 2);
        end
```

```
    values = values * spec.scale + spec.offset;

    name = spec.name;

    if ( strcmp(name, 'Wheel speed (front-left)') )
        frontLeft = values;
        timeCAN = selection(:,1);
    elseif ( strcmp(name, 'Wheel speed (front-right)') )
        frontRight = values;
    elseif ( strcmp(name, 'Wheel speed (rear-left)') )
        rearLeft = values;
    elseif ( strcmp(name, 'Wheel speed (rear-right)') )
        rearRight = values;
    elseif ( strcmp(name, 'Latitude') )
        latitude = values;
        timeGPS = selection(:,1);
    elseif ( strcmp(name, 'Longitude') )
        longitude = values;
    elseif ( strcmp(name, 'xAcceleration') )
        xAcceleration = values;
    elseif ( strcmp(name, 'yAcceleration') )
        yAcceleration = values;
    elseif ( strcmp(name, 'zAcceleration') )
        zAcceleration = values;
    end
end

for j=1:length(frontLeft)

    if ( isempty(zAcceleration) || zAcceleration(end) == 0 )    %step 1: wait for acceleration-data

        init = 1;
        continue
```

```
elseif ( init == 1 )  %step 2: collect acceleration-data during standstill => calibration

     k = 1;

     if (frontLeft(j) == 0 && frontRight(j) == 0 && rearLeft(j) == 0 && rearRight(j) == 0)
         while ( k <= length(latitude) && ( timeGPS(k) >= timeCAN(j) ) )
             temp = [temp ; timeGPS(k) latitude(k) longitude(k) xAcceleration(k) yAcceleration(k)
                    zAcceleration(k)];
             k = k+1;
         end
     else
            if(isempty(temp) == 0)
                %calibrate vertical acceleration
                [ temp(:,6) ] = accCalibration( temp(:,4), temp(:,5), temp(:,6) );
                meanZAcc = mean( temp(:,6) );   %necessary for smallPeakFilter
            end
            temp = [];                          %clear buffer
            init = 0;                           %stop initialization-stage
     end

%Step 4: if vehicle stands still => reinitialize
elseif ( frontLeft(j) == 0 && frontRight(j) == 0 && rearLeft(j) == 0 && rearRight(j) == 0 )

    init = 1;

else                                                    %step 3: if the vehicle moves...

    diffFront = frontLeft(j) - frontRight(j);
    diffRear = rearLeft(j) - rearRight(j);
    diffFrontRear = diffFront - diffRear;

     %calibrate vertical acceleration
    [ zAcceleration ] = accCalibration( xAcceleration, yAcceleration, zAcceleration );
     %filter with the provided threshold-percentage and use an offset
    zAcceleration = smallPeakFilter( zAcceleration, meanZAcc, meanZAcc * toleranceAcc );
```

```
            if ( abs(diffFrontRear) >= thresholdWheelSpeed )          %...detect pits (wheelspeeds)
                detectionWheels = [detectionWheels; timeCAN(j) frontLeft(j) frontRight(j) rearLeft(j)
                                   rearRight(j)];
            end

            if ( isempty(detectionWheels) == 0 )
                for l=1:size(detectionWheels,1)
                    k = 1;
                    detectionTimeCAN = detectionWheels(l,1);
                          while ( k <= length(latitude) )           %...detect pits (vertical acceleration)
                        if ( ( (detectionTimeCAN - 0.2) <= timeGPS(k) ) && ( (detectionTimeCAN + 0.2) >=
                           timeGPS(k) ) && ( zAcceleration(k) ~= meanZAcc ) )
                            pitsDef = [pitsDef; timeGPS(k) latitude(k) longitude(k) xAcceleration(k)
                                       yAcceleration(k) zAcceleration(k)];
                        end
                        k = k+1;
                    end
                end
            end
        end
    end
end

pitsDef = unique(pitsDef,'rows');

pits = pitsDef;

end
```

XIV TRAJECT MET SELECTIEVE INVENTARIS

In deze bijlage worden twee testtraject in kaart gebracht, met een selectie uit de inventaris van de defecten in het wegdek. Deze selectie is een voorbeeld van wat er met het toegepast algoritme aan defecten kan gedetecteerd worden. Dit indien een drempelwaarde van 1.3 km/u wordt gekozen voor de eerste parameter van het algoritme, en een tolerantie van 3.5 % voor de tweede parameter. Hierbij wordt het groene testtraject met succes doorlopen, terwijl een uitbreiding van dit traject, welke aangeduid is in het rood, enkele problemen aantoont met het gebruikte algoritme.

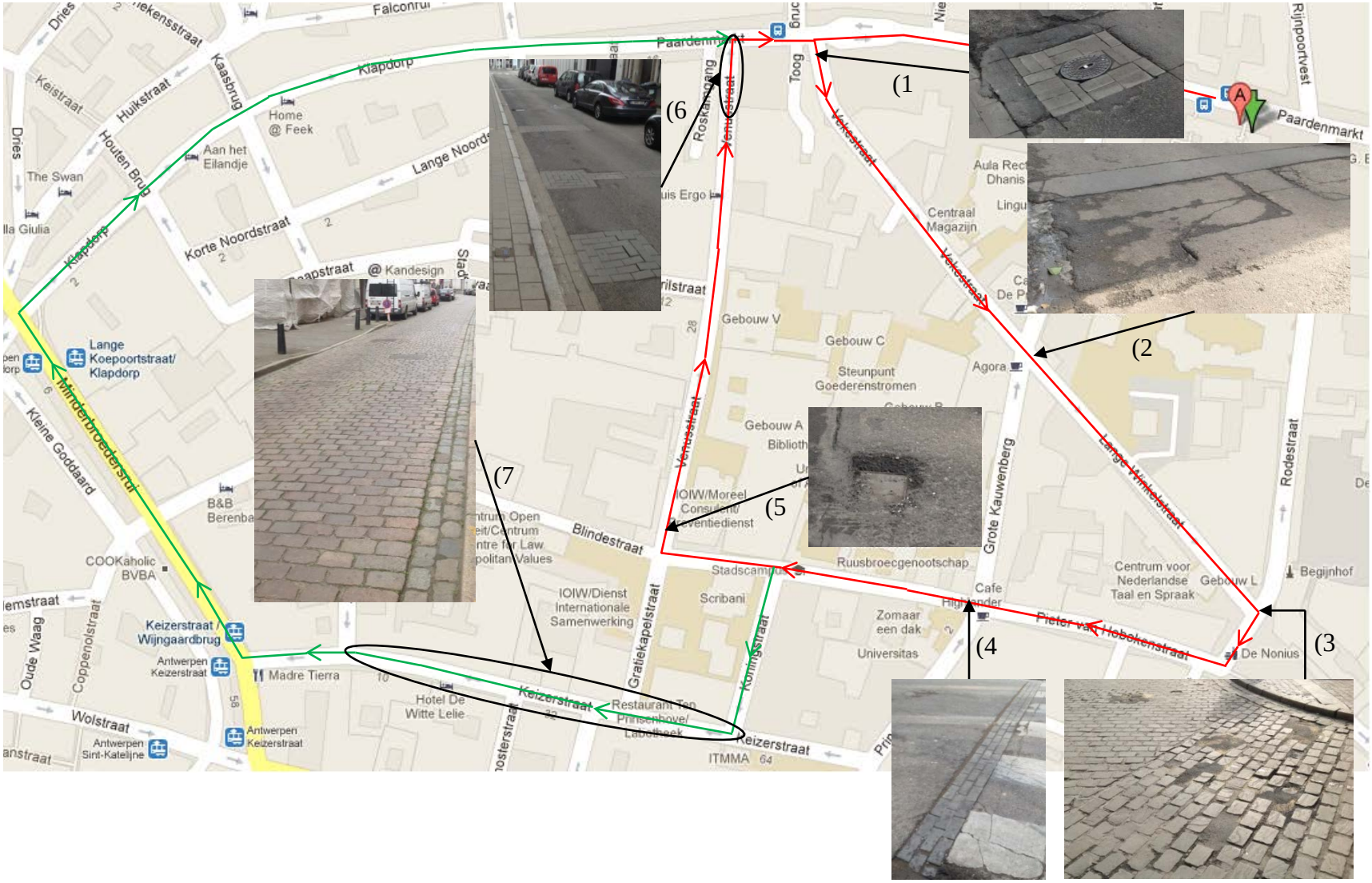
Locaties (1) tot (6) tonen defecten aan het wegdek, welke door het algoritme als dusdanig worden herkend. Zo worden sterk verzonken riooldeksel (1) als een mogelijk defect herkend. Het is echter wel zo dat de verzinking voldoende merkbaar moet zijn voor de inzittenden, en dus als storend moet worden ondervonden, zodat niet ieder deksel gedetecteerd wordt als een defect in het wegdek. Door de wegdekbeheerder aangebrachte herstellingen kunnen na verloop van tijd hun effect mislopen (2), (4), (6), wat eveneens de nodige ongemakken met zich meebrengt, en zodus dienen te worden aangeduid door het algoritme. Het algoritme is eveneens in staat om gebrekkige punten in een kasseistrook te onderscheiden van punten waar er zich geen defecten voordoen (3), althans in het geval van het groene testtraject.

Wanneer echter het uitgebreide testtraject wordt gevolgd, aangeduid in het groen, detecteert het algoritme de gehele kasseistrook van de Keizerstraat (7) als een opeenvolging van putten. Dit is tegenstrijdig met de eerdere vaststellingen bij het groene testtraject waar dit probleem zich niet voordeed.

In de doelstellingen is vooropgesteld dat het algoritme in staat moet zijn om putten, met een diameter van 20 cm en een diepte van 4 cm, te detecteren. Het testvoertuig heeft tijdens het volgen van het groene traject minstens 1 maal in een vergelijkbare put gereden (5), waarvan de diameter 22 cm en de diepte 3.5 cm bedraagt.

De data van de CAN-bus is eveneens terug te vinden op de CD-ROM, en dit voor beide testtrajecten, onder de respectievelijke folders ‘Praktisch\Groene traject’ en ‘Praktisch\Rode traject’. Bijhorend zijn de nodige functies toegevoegd, welke nodig zijn om de defecten in het wegdek te detecteren. Een voorbeeld van het oproepen van de functie ter detectie van defecten in het wegdek is:

[pits] = findAnomalies(data, focus(), 100, 1.3, 0.035);


(1
(2
(3
(4
(5
(6
(7
Paardenmarkt
Verversstraat
Grote Kauwenberg
Lange Winkelstraat
Rodestraat
Pieter van Hobokenstraat
Blindestraat
Gratiekapelstraat
Keizerstraat
Wolstraat
Korte Noordstraat
Lange Koepoortstraat/ Klapdorp
Kleine Goddaard
Windertbroedersrui
Gebouw V
Gebouw C
Gebouw A
Agora
Steunpunt Goederenstromen
Ruusbroecgenootschap
Centrum voor Nederlandse Taal en Spraak
Begijnhof
De Nonius
Universitas
Zomaar een dak
Scribani
IOIW/Dienst Internationale Samenwerking
IOIW/Moreel Consulent/ Eventiedienst
Centraal Magazijn
Madre Tierra
Hotel De Witte Lelie
ITMMA
Keizerstraat / Wijngaardbrug
Antwerpen Keizerstraat
Antwerpen Sint-Katelijne
COOKaholic BVBA
B&B Berenba
Home @ Feek
Aan het Eilandje
The Swan
@ Kandesign